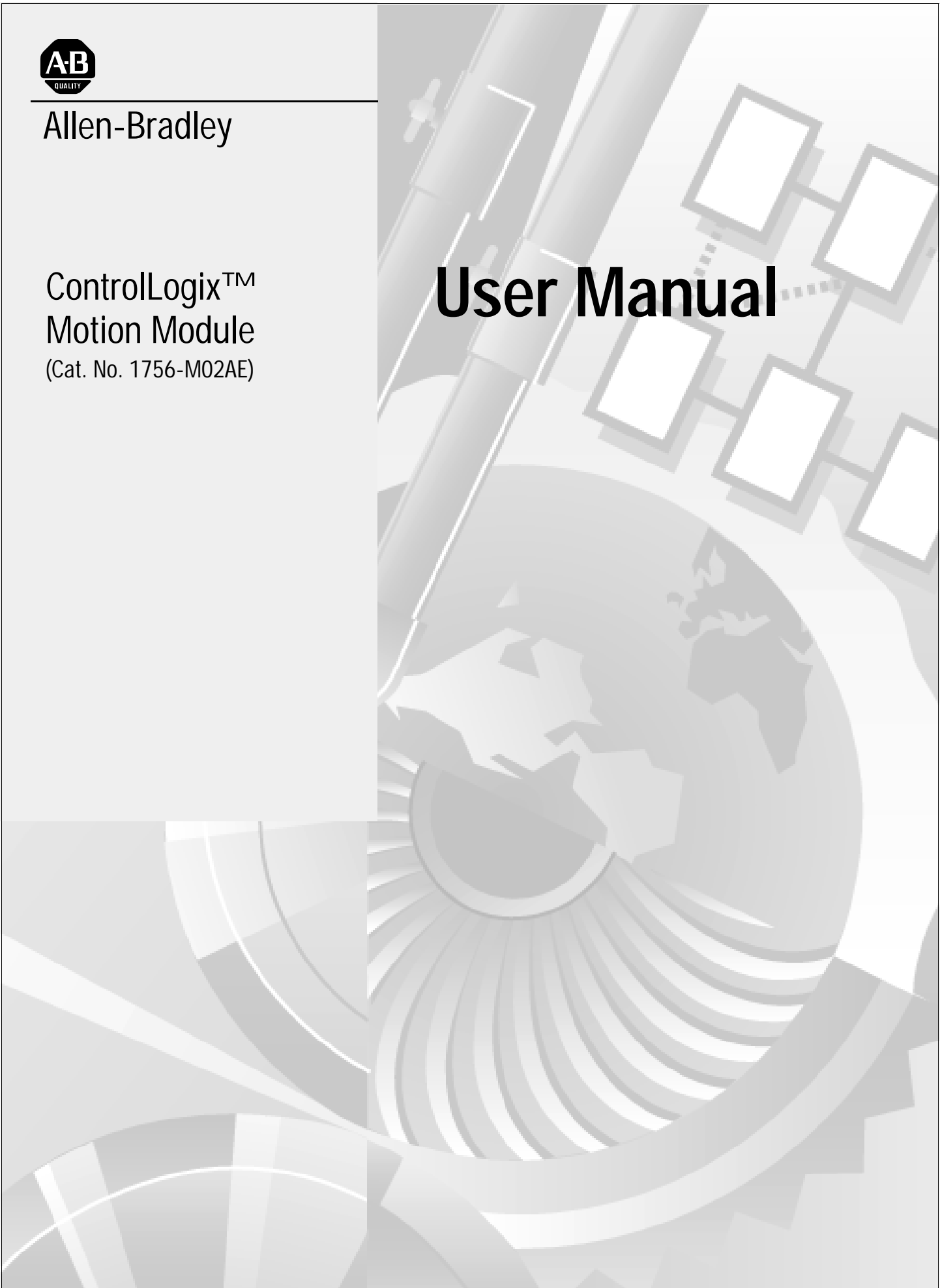




Allen-Bradley

ControlLogix™
Motion Module
(Cat. No. 1756-M02AE)

User Manual



Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs, and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley publication SGI-1.1, Safety Guidelines for the Application, Installation and Maintenance of Solid-State Control (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

Reproduction of the contents of this copyrighted publication, in whole or part, without written permission of Allen-Bradley Company, Inc., is prohibited.

Throughout this manual we use conventions to make you aware of safety considerations:



ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss.

Attention statements help you to:

- identify a hazard
- avoid a hazard
- recognize the consequences

Important: Identifies information that is critical for successful application and understanding of the product.

Using This Manual

Preface P-1

Who Should Use This Manual P-1

The Purpose of This Manual P-2

Conventions Used in This Manual P-3

Related Documentation P-3

Rockwell Automation Support P-4

 Local Product Support P-4

 Technical Product Assistance P-4

**Understanding the ControlLogix
Motion Control System**

Chapter 1 1-1

Understanding ControlLogix Motion Control 1-1

Identifying the Components of the ControlLogix Motion System 1-2

 The Logix5550 Controller 1-2

 The Analog/Encoder Servo Module (1756-MO2AE) 1-2

 RSLogix 5000 Programming Software 1-3

Developing a Motion Control Application Program 1-4

 Understanding Application Program Development 1-4

 Understanding the MOTION_INSTRUCTION Tag 1-5

 Understanding Motion Status and Configuration Parameters 1-6

 Modifying Motion Configuration Parameters 1-6

 Handling Motion Faults 1-6

Installing Your Motion Module

Chapter 2 2-1

Identifying the Module Components 2-2

 Identifying the Motion Module 2-2

 Identifying the Removable Terminal Block and Housing 2-3

Determining the Power Requirements 2-3

Preventing Electrostatic Discharge 2-4

Removing and Inserting Under Power (RIUP) 2-4

Understanding Compliance with the European Union Directive 2-5

 EMC Directive 2-5

 Low Voltage Directive 2-5

Installing the Module 2-6

Keying the Removable Terminal Block 2-8

Wiring a Removable Terminal Block 2-10

 Wiring a Spring Clamp RTB 2-10

 Wiring a Cage Clamp RTB 2-11

Assembling the Removable Terminal Block and the Housing 2-12

Installing the Removable Terminal Block onto the Module 2-12

Checking the LED Indicators 2-14

Removing the Removable Terminal Block from the Module 2-15

Removing the Module from the Chassis 2-16

Getting Started With Your Motion Module

Chapter 3 3-1

Understanding the Getting Started Tutorial 3-2

Setting the Master Coordinated System Time 3-3

Adding the 1756-M02AE Module 3-6

Naming an Axis 3-8

Configuring a Motion Axis 3-9

Running Hookup Diagnostics and Auto Tuning 3-19

Entering a Ladder Logic Program 3-29

 Creating Additional Tags Using the Tag Editor 3-32

 Completing Your Application Program 3-34

Adding and Configuring Your Motion Module

Chapter 4 4-1

Understanding Application Program Development 4-1

Selecting the Master Coordinated System Time 4-2

Adding the 1756-M02AE Module 4-5

Naming an Axis 4-7

Configuring a Motion Axis 4-8

Assigning Additional Motion Modules and Axes 4-27

Running Hookup Diagnostics and Auto Tuning 4-28

Developing a Motion Application Program 4-38

Understanding a Programming Example 4-39

| | | |
|--|---|------------|
| Understanding Motion Instructions | Chapter 5 | 5-1 |
| | Understanding Motion State Instructions | 5-2 |
| | Understanding Motion Move Instructions | 5-3 |
| | Understanding Motion Group Instructions | 5-4 |
| | Understanding Motion Event Instructions | 5-5 |
| | Understanding Motion Configuration Instructions | 5-6 |
| | | |
| Troubleshooting | Chapter 6 | 6-1 |
| | Understanding Module Status Using the OK Indicator | 6-1 |
| | Understanding Module Status Using the FDBK Indicator | 6-2 |
| | Understanding Module Status Using the DRIVE Indicator | 6-3 |
| | | |
| Specifications and Performance | Appendix A | A-1 |
| | Understanding Motion Module Specifications | A-1 |
| | Understanding Coarse Update Rate Calculations | A-4 |
| | Defining the Baseline Task Time (Table 1) | A-4 |
| | Understanding Action Timing (Table 2) | A-5 |
| | Using the Sample Calculations Worksheet | A-6 |
| | Understanding Sample Calculation 1 | A-7 |
| | Understanding Sample Calculation 2 | A-9 |
| | | |
| Loop and Interconnect Diagrams | Appendix B | B-1 |
| | Understanding Block Diagrams | B-2 |
| | Using a 1756-M02AE Module With a Torque Servo Drive | B-3 |
| | Using a 1756-M02AE Module With a Velocity Servo Drive | B-4 |
| | Understanding Wiring Diagrams | B-5 |
| | Wiring to a Servo Module RTB | B-5 |
| | Wiring to an Ultra 100 Series Drive | B-6 |
| | Wiring to an Ultra 200 Series Drive | B-7 |
| | Wiring to a 1394 Servo Drive | B-8 |
| | Wiring the 1394-SA15 Cable | B-9 |
| | Wiring Registration Sensors | B-10 |
| | Wiring the Home Limit Switch Input | B-11 |
| | Wiring the OK Contacts | B-12 |

| | |
|--------------------------------------|---|
| The Motion Control Structures | Appendix C C-1 |
| | Understanding the AXIS Structure C-2 |
| | Understanding the MOTION_GROUP Structure C-8 |
| | Understanding the MOTION_INSTRUCTION Structure C-11 |
| | Understanding Error Codes (.ERR) C-12 |
| | Understanding Message Status (.STATUS) C-13 |
| | Understanding Execution Status (.STATE) C-13 |
| | |
| The Motion Attributes | Appendix D D-1 |
| | Motion Instance Variables D-1 |
| | |
| Instruction Timing | Appendix E E-1 |
| | Understanding Immediate Type Instructions E-1 |
| | Understanding Message Type Instructions E-3 |
| | Understanding Process Type Instructions E-5 |
| | |
| Fault Handling | Appendix F F-1 |
| | Handling Motion Faults F-1 |
| | Understanding Errors F-1 |
| | Understanding Minor/Major Faults F-2 |

Using This Manual

This preface describes how to use this manual. The following table describes what the preface contains:

| For information about | See page |
|---------------------------------|-----------------|
| Who Should Use This Manual | P-1 |
| The Purpose of This Manual | P-2 |
| Conventions Used in This Manual | P-3 |
| Related Documentation | P-3 |
| Rockwell Automation Support | P-4 |

Who Should Use This Manual

To use this manual, you should be able to program and operate the Allen-Bradley Logix5550™ controller to efficiently use you motion control modules.

If you need more information about programming and operating the Logix5550 controller, refer to the Logix5550 Controller User Manual, publication number 1756-6.5.12.

The Purpose of This Manual

This manual describes how to install, configure, and troubleshoot your ControlLogix motion module.

The following table shows the contents of each section in this manual:

| Section | Contains |
|---|---|
| Chapter 1 Understanding the ControlLogix Motion Control System | Information about the ControlLogix motion control system. |
| Chapter 2 Installing Your Motion Module | Information about installing and wiring the motion module. |
| Chapter 3 Getting Started With Your Motion Module | A tutorial for configuring and using your 1756-M02AE motion module. |
| Chapter 4 Adding and Configuring Your Motion Module | A step-by-step procedure for configuring your motion module using the RSLogix™ 5000 programming software. |
| Chapter 5 Understanding Motion Instructions | Information about the 27 motion instructions provided in the RSLogix 5000 programming software. |
| Chapter 6 Troubleshooting | Information about troubleshooting your ControlLogix motion control system. |
| Appendix A Specifications and Performance | Specifications and performance guidelines for the motion module. |
| Appendix B Loop and Interconnect Diagrams | Loop diagrams and wiring diagrams for your ControlLogix motion control system. |
| Appendix C The Motion Control Structures | An explanation of the motion control structures. |
| Appendix D The Motion Attributes | Information about the motion attributes. |
| Appendix E Instruction Timing | Information about types of timing for motion instructions. |
| Appendix F Fault Handling | Information about motion control faults. |

Conventions Used in This Manual

This manual uses the following conventions for using windows and dialog boxes.

| Convention | Example |
|---|---|
| Names of fields in windows and dialog boxes are <i>italicized</i> . | In the <i>Name</i> field, type the name of your axis. |
| Input that you type exactly is bold . | In the <i>Name</i> field, type Module_1 . |

Note: Some windows and dialog boxes may contain greyed-out (unavailable) fields because of configuration options you have chosen. If a field is greyed-out, it means the field does not apply to your configuration and is not required.

Related Documentation

The following table lists related ControlLogix documentation:

| Publication Number | Publication | Description |
|--------------------|--|---|
| 1756-5.47 | Analog Encoder (AE) Servo Module Installation Instructions | Provides instructions for installing, wiring, and troubleshooting your 1756-M02AE servo module. |
| 1756-5.72 | ControlLogix Motion Module Application Guide | Provides in-depth descriptions of motion concepts and instructions. |
| 1756-10.1 | Logix5550 Controller Quick Start | Provides instructions for installing the Logix5550 controller and its components. |
| 1756-6.5.11 | Logix5550 Controller Instruction Set Quick Reference | Provides a brief description of the RSLogix 5000 programming software instructions. |
| 1756-6.5.12 | Logix5550 Controller User Manual | Provides information for using your Logix5550 controller and its components. |
| 1756-6.4.1 | Logix5550 Controller Instruction Set Reference Manual | Provides descriptions of all the instructions supported by the RSLogix 5000 programming software. |
| 1756-5.33 | Logix5550 Memory Board Installation Instructions | Provides instructions for installing the Logix5550 memory board. |

For more information on the documentation, refer to the Allen-Bradley Publication Index, publication number SD499.

Rockwell Automation Support

Rockwell Automation offers support services worldwide, with over 75 sales/support offices, 512 authorized distributors, and 260 authorized systems integrators located throughout the United States. In addition, Rockwell Automation representatives are located in every major country in the world.

Local Product Support

Contact your local Rockwell Automation representative for:

- sales and order support
- product technical training
- warranty support
- support service agreements

Technical Product Assistance

If you need to contact Rockwell Automation for technical assistance, please review Chapter 6 - *Troubleshooting* in this manual. If the problem persists, call your local Rockwell Automation representative.

Understanding the ControlLogix Motion Control System

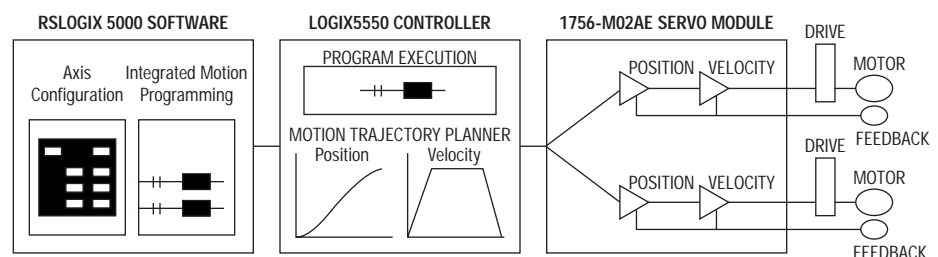
This chapter describes the ControlLogix motion control system. The following table shows the contents of this chapter:

| For information about | See page |
|--|----------|
| Understanding ControlLogix Motion Control | 1-1 |
| Identifying the Components of the ControlLogix Motion System | 1-2 |
| Developing a Motion Control Application Program | 1-4 |

Understanding ControlLogix Motion Control

The Logix5550 controller, 1756-M02AE servo module, and RSLogix 5000 programming software provide integrated motion control support.

- The Logix5550 controller contains a high-speed motion task, which executes ladder motion commands and generates position and velocity profile information. The controller sends this profile information to one or more 1756-M02AE servo modules. You can use several Logix5550 controllers in each chassis. Each controller can control up to 16 1756-M02AE servo modules.
- The 1756-M02AE servo module connects to a servo drive and closes a high-speed position and velocity loop. Each Logix5550 controller can support up to 16 1756-M02AE servo modules. Each 1756-M02AE module can control up to two axes.
- RSLogix 5000 programming software provides complete axis configuration and motion programming support.



Identifying the Components of the ControlLogix Motion System

The Logix5550 Controller

The Logix5550 controller is the main component in the ControlLogix system. It supports sequential and motion functions, and it performs all of the motion command execution and motion trajectory planner functions. You can use one or more Logix5550 controllers in each chassis, and each controller can control up to 16 motion modules.

The Logix5550 controller provides the following motion support:

- Twenty-seven motion instructions
- A high-speed motion task, which manages motion functions and generates move profiles
- The ability to control up to 16 Analog/Encoder servo modules for a total of 32 axes

The Analog/Encoder Servo Module (1756-M02AE)

The Analog/Encoder servo module provides an analog/quadrature encoder servo drive interface. The servo module receives configuration and move information from the Logix5550 controller and manages motor position and velocity.

The servo module supports the following:

- Connection capability for up to two drives
 - $\pm 10\text{V}$ analog outputs
 - Quadrature encoder inputs
 - Home limit switch inputs
 - Drive fault inputs
 - Drive enable outputs
 - 5V or 24V registration inputs
- 200 μs position and velocity loop updates

RSLogix 5000 Programming Software

The RSLogix 5000 programming software provides complete programming and commissioning support for the ControlLogix system. RSLogix 5000 is the only programming software needed to fully configure and program ControlLogix motion control systems.

RSLogix 5000 software provides the following motion support:

- Wizards for servo axis configuration including drive hookup diagnostics and auto tuning
- Ladder-based application programming including support for 27 motion commands

Developing a Motion Control Application Program

This section provides an introduction to concepts used in developing application programs for motion control. These concepts include:

- Understanding application program development
- Understanding the MOTION_INSTRUCTION tag
- Understanding motion status and configuration parameters
- Modifying motion configuration parameters
- Handling motion faults

Understanding Application Program Development

Developing a motion control application program involves the following:

| Task | Description |
|---|---|
| Select the master coordinated system time | Sets one controller as the master controller. Once you complete this step, you can synchronize all the motion modules and Logix5550 controllers in your chassis |
| Add a motion module | Adds a motion module to your application program |
| Name an axis | Adds an axis to your application program |
| Configure an axis | Configures each axis for motion control |
| Assign additional servo modules and axes | Adds additional modules and axes to your application program |
| Run hookup diagnostics and auto tuning | Completes hookup diagnostics and auto tuning for each axis |
| Develop a motion application program | Create a program for your motion control application |

For more information about completing these tasks, refer to *Chapter 4 - Adding and Configuring Your Motion Module*.

Understanding the MOTION_INSTRUCTION Tag

The controller uses the MOTION_INSTRUCTION tag (structure) to store status information during the execution of motion instructions. Every motion instruction has a motion control parameter that requires a MOTION_INSTRUCTION tag to store status information.



ATTENTION: Tags used for the motion control parameter of instructions should only be used once. Re-use of the motion control parameter in other instructions can cause unintended operation of the control variables.

For more information about the MOTION_INSTRUCTION tag, refer to Appendix C - *The Motion Control Structures*.

Understanding Motion Status and Configuration Parameters

You can read motion status and configuration parameters in your ladder logic program using two methods.

| Method | Example | For more information |
|---|--|--|
| Directly accessing the AXIS and MOTION_GROUP structures | <ul style="list-style-type: none"> Axis faults Motion status Servo status | Refer to Appendix C - <i>The Motion Control Structures</i> |
| Using the GSV instruction | <ul style="list-style-type: none"> Actual position Command position Actual velocity | Refer to the <i>Input/Output Instructions</i> chapter of the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1 |

Modifying Motion Configuration Parameters

In your ladder logic program, you can modify motion configuration parameters using the SSV instruction. For example, you can change position loop gain, velocity loop gain, and current limits within your program.

For more information about the SSV instruction, refer to the *Input/Output Instructions* chapter of the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

Handling Motion Faults

Two types of motion faults exist.

| Type | Description | Example |
|-------------|--|--|
| Errors | <ul style="list-style-type: none"> Do not impact controller operation Should be corrected to optimize execution time and ensure program accuracy | A Motion Axis Move (MAM) instruction with a parameter out of range |
| Minor/Major | <ul style="list-style-type: none"> Caused by a problem with the servo loop Can shutdown the controller if you do not correct the fault condition | The application exceeded the PositionErrorTolerance value |

For more information about handling faults, see *Handling Controller Faults* in the Logix5550 Controller User Manual, publication 1756-6.5.12.

Installing Your Motion Module

This chapter describes how to install you motion module. The following table shows the contents of this chapter:

| For information about | See page |
|--|----------|
| Identifying the Module Components | 2-2 |
| Determining the Power Requirements | 2-3 |
| Preventing Electrostatic Discharge | 2-4 |
| Removing and Inserting Under Power (RIUP) | 2-4 |
| Understanding Compliance with the European Union Directive | 2-5 |
| Installing the Module | 2-6 |
| Keying the Removable Terminal Block | 2-8 |
| Wiring a Removable Terminal Block | 2-10 |
| Assembling the Removable Terminal Block and the Housing | 2-12 |
| Installing the Removable Terminal Block onto the Module | 2-12 |
| Checking the LED Indicators | 2-14 |
| Removing the Removable Terminal Block from the Module | 2-15 |
| Removing the Module from the Chassis | 2-16 |

The Analog Encoder (AE) Servo module mounts in a ControlLogix chassis and uses a removable terminal block (RTB) to connect all field-side wiring.

Before you install your module you should have:

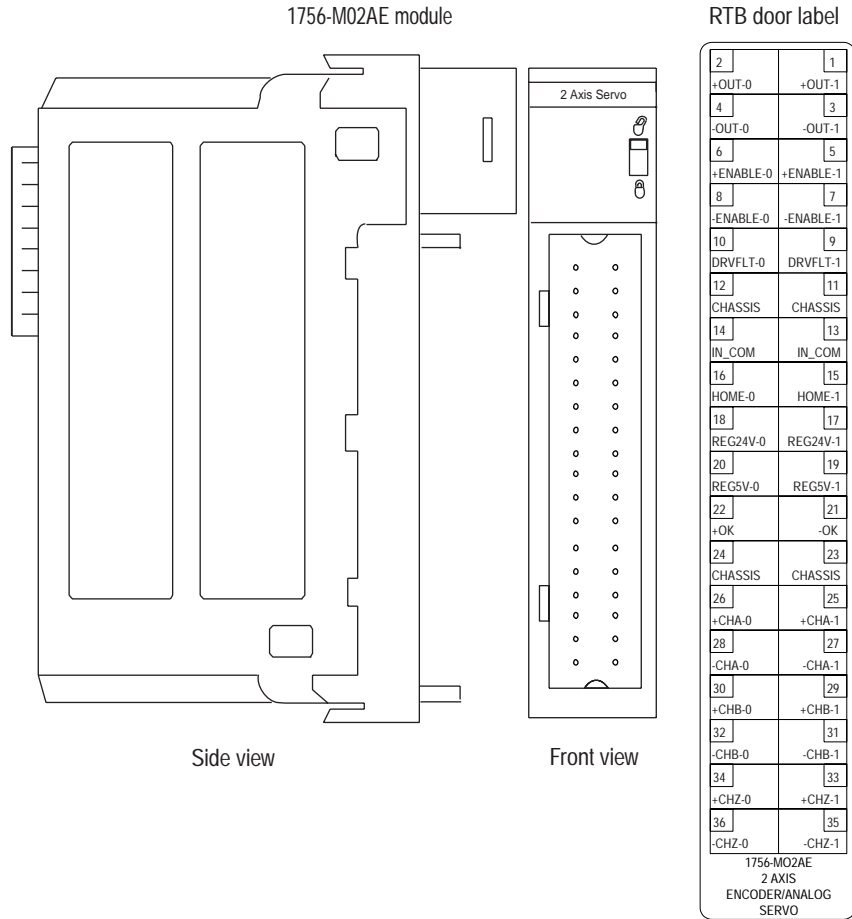
- installed and grounded a 1756 chassis and power supply.
- ordered and received an RTB and its components for your application.

Identifying the Module Components

Identifying the Motion Module

You received two components with your order:

- 1756-M02AE module
- RTB door label



If you did not receive these components, contact your local Allen-Bradley representative.

Identifying the Removable Terminal Block and Housing

A separately-ordered RTB connects field-side wiring to the module. You cannot use your module without an RTB and its components.

Use one of the following RTBs with your module:

- 1756-TBCH 36-position cage clamp RTB
- 1756-TBS6H 36-position spring clamp RTB

You received the following components with your RTB:

- 1756-TBH standard-depth RTB housing
- Wedge-shaped keying tabs and U-shaped keying bands
- RTB door label

Determining the Power Requirements

This module receives power from the 1756 chassis power supply and requires two sources of power: 700 mA at 5V and 2.5 mA at 24V from the backplane. Add this current to the requirements of the other modules in the chassis to prevent overloading the backplane power supply.

Preventing Electrostatic Discharge



ATTENTION: Electrostatic discharge can damage the servo board if you touch the circuitry or connector pins without taking precautions. Follow these guidelines when you handle the servo board:

- Touch a grounded object to discharge potential static.
 - Wear an approved grounding wriststrap.
 - Do not touch the connector or connector pins on the servo board.
 - Do not touch circuit components inside the servo board.
 - If available, use a static-safe work station.
-

Removing and Inserting Under Power (RIUP)



ATTENTION: This module is designed so you can remove and insert it under backplane power and field-side power. When you remove or insert a module while field-side power is applied, you can cause an electrical arc. An electrical arc can cause personal injury or property damage because it can:

- Send an erroneous signal to your system field devices causing unintended machine motion or loss of process control.
- Cause an explosion in a hazardous environment.

Repeated electrical arcing causes excessive wear to contacts on both the module and its mating connector. Worn contacts may create electrical resistance. For additional information on RIUP, please contact your local Allen-Bradley sales representative

Understanding Compliance with the European Union Directive

If this product bears the CE marking, it is approved for installation within the European Union and EEA regions. It has been designed and tested to meet the following directives.

EMC Directive

This product is tested to meet Council Directive 89/336/EEC Electromagnetic Compatibility (EMC) and the following standards, in whole or in part, documented in a technical construction file:

- EN 50081-2EMC - Generic Emission Standard, Part 2 - Industrial Environment
- EN 50082-2EMC - Generic Immunity Standard, Part 2 - Industrial Environment

This product is intended for use in an industrial environment.

Low Voltage Directive

This product is tested to meet Council Directive 73/23/EEC Low Voltage, by applying the safety requirements of EN 61131-2 Programmable Controllers, Part 2 - Equipment Requirements and Tests.

For specific information required by EN 61131-2, see the appropriate sections in this publication, as well as the following Allen-Bradley publications:

- Industrial Automation Wiring and Grounding Guidelines For Noise Immunity, publication 1770-4.1
- Automation Systems Catalog, publication B111

This equipment is classified as open equipment and must be installed (mounted) in an enclosure during operation as a means of providing safety protection.

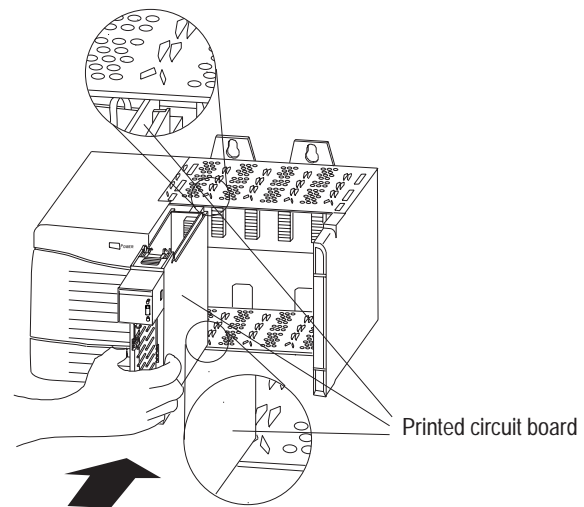
Installing the Module



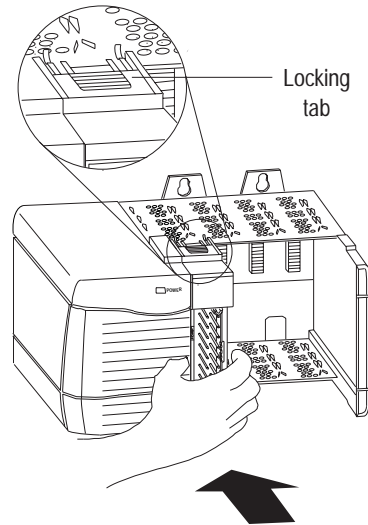
ATTENTION: When you remove or insert an RTB with field-side power applied, unintended machine motion or loss of process control can occur. Exercise extreme caution when power is applied. Failure to observe this caution can cause personal injury.

To install the AE module:

1. Align the module circuit board with the top and bottom chassis guides.



2. Push evenly and firmly to seat the module in the chassis. It is seated when the top and bottom locking tabs have snapped into place.



Note: The 1756 chassis provides grounding for your module.

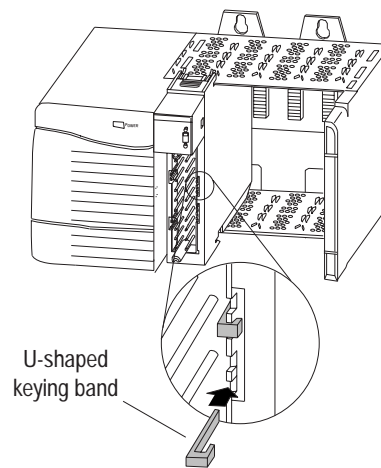
Keying the Removable Terminal Block

To identify the RTB that belongs with each module, you can use a module keying pattern. First, you can create a unique keying pattern for your module using the U-shaped keying bands that you received with your RTB. Then you can use the keying tabs to key the RTB with the same pattern as the module.

To prevent confusion, use a unique keying pattern for each module.

To key the module:

1. Insert the U-shaped keying band with the longer side near the terminals.

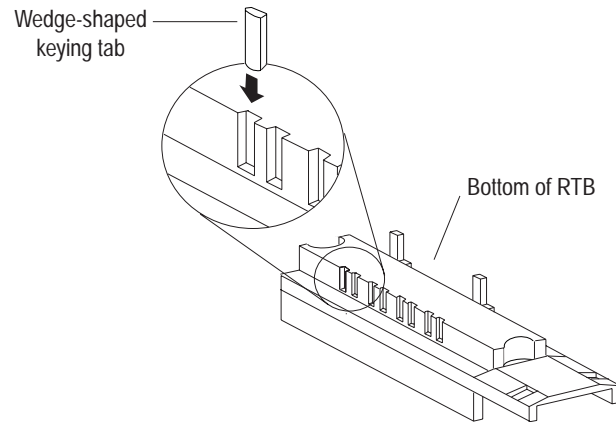


2. Push the keying band onto the module until it snaps into place.

To key your removable terminal block:

1. With the rounded edge first, insert the wedge-shaped keying tab on the RTB.

Note: Insert the wedge-shaped keying tabs in positions that correspond to unkeyed positions on the module.



2. Push the keying tab onto the RTB until it stops.

Note: To use the RTB in future module applications, you can reposition the keying tabs on the RTB.

Wiring a Removable Terminal Block

There are two types of RTBs:

- spring clamp
- cage clamp

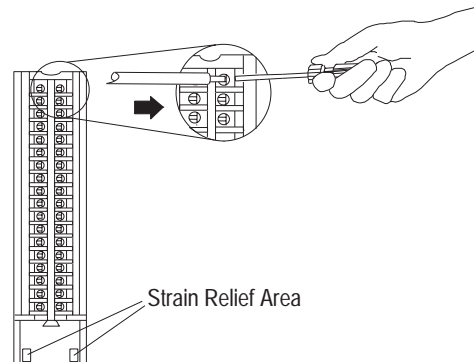
This section describes how to wire each type of RTB. For wiring diagrams, refer to Appendix B - *Interconnect Diagrams*.

Wire the RTB before installing it onto the module. Use a 1/8 inch (3.2mm) maximum flat-bladed screwdriver.

Wiring a Spring Clamp RTB

To wire a spring clamp RTB:

1. Strip a maximum of 7/16 in. (11mm) of insulation from the end of your wire.
2. Insert the screwdriver into the outer hole of the RTB.
3. Insert the wire into the open terminal and remove the screwdriver.

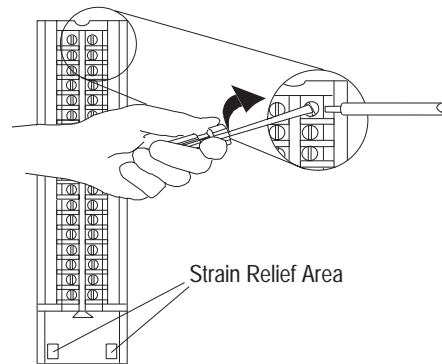


4. After you complete field-side wiring, secure the wires in the strain relief area with a cable-tie.

Wiring a Cage Clamp RTB

To wire a cage clamp RTB:

1. Strip 5/16-3/8 in. (8-9.5mm) of insulation from the end of your wire.
2. Insert the wire into the open terminal.
3. Turn the screw clockwise to close the terminal on the wire. Use 5 lb-in. (0.5 Nm) maximum torque.

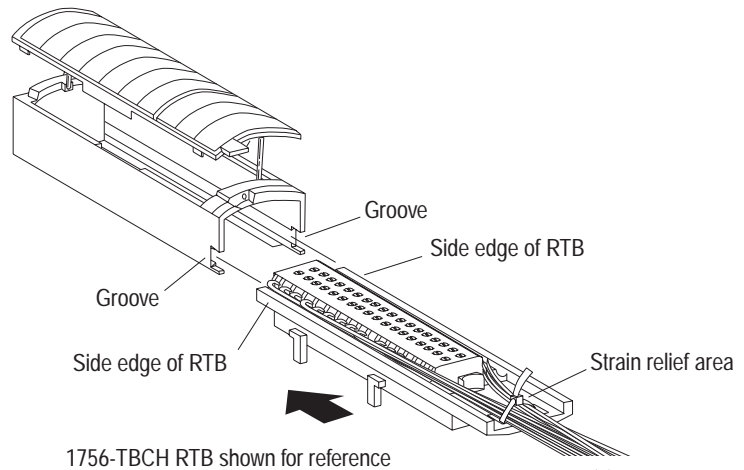


4. After you complete field-side wiring, secure the wires in the strain relief area with a cable-tie.

Assembling the Removable Terminal Block and the Housing

To assemble the removable terminal block and housing:

1. Align the grooves at the bottom of each side of the housing with the side edges of the RTB.
2. Slide the RTB into the housing until it snaps into place.



Installing the Removable Terminal Block onto the Module



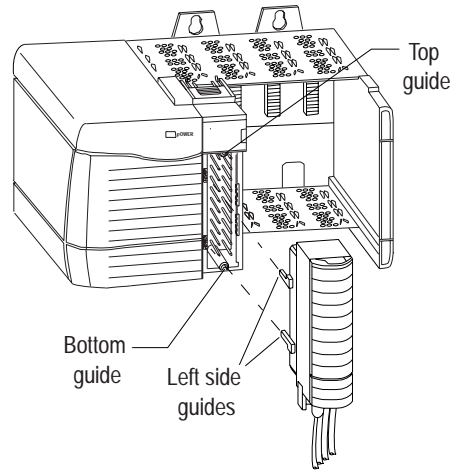
ATTENTION: A shock hazard exists. If the RTB is installed onto the module while the field-side power is applied, the RTB is electrically live. Do not touch the RTB terminals. Failure to observe this caution can cause personal injury.

Before installing the RTB, make certain:

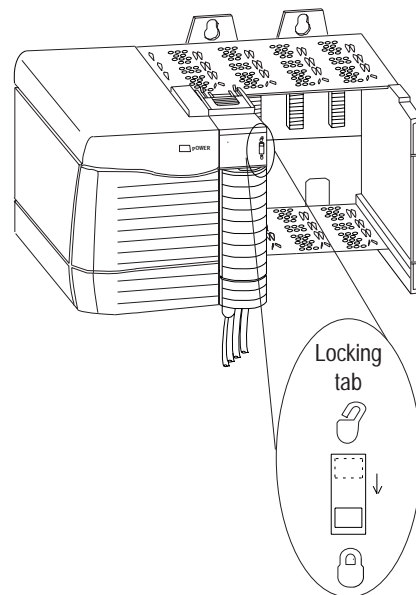
- field-side wiring of the RTB has been completed.
- the RTB housing is snapped into place on the RTB.
- the RTB housing door is closed.
- the locking tab at the top of the module is unlocked.

To install the removable terminal block onto the module:

1. Align the top, bottom, and left side guides of the RTB with the guides on the module.

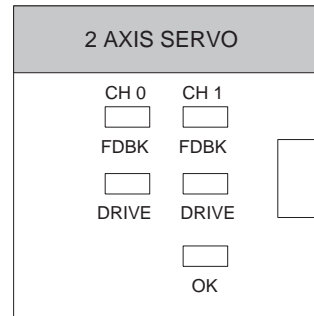


2. Press quickly and evenly to seat the RTB on the module until the latches snap into place.
3. Slide the locking tab down to lock the RTB onto the module.



Checking the LED Indicators

The module provides bi-colored LED indicators to show individual drive and feedback status for both axes and a single bi-colored LED for module OK.



During power up, the module completes an indicator test. The OK indicator turns red for 1 second and then turns to flashing green if the module passes all its self-tests.

For more information about the LED indicators, refer to Chapter 6 - *Troubleshooting*.

This completes installation of the module.

Removing the Removable Terminal Block from the Module

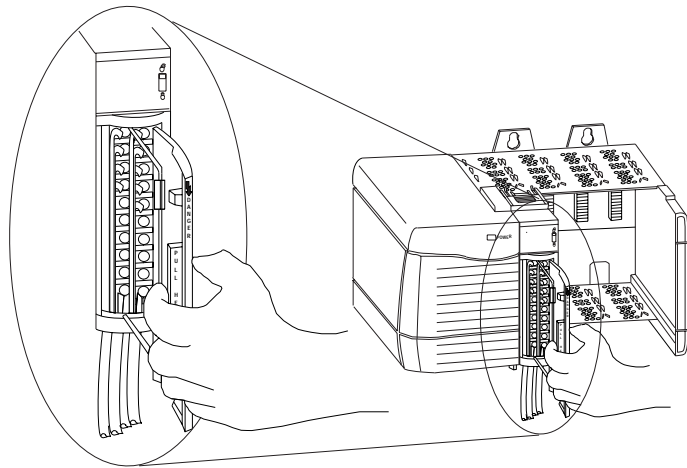


ATTENTION: A shock hazard exists. If the RTB is removed from the module while the field-side power is applied, the module is electrically live. Do not touch the RTB's terminals. Failure to observe this caution can cause personal injury.

You must remove the RTB before you can remove the module.

To remove the RTB from the module:

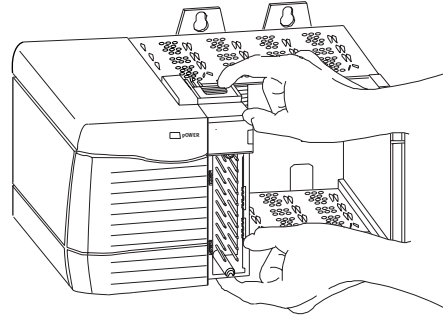
1. Unlock the locking tab at the top of the module.
2. Open the RTB door using the bottom tab.
3. Hold the spot marked PULL HERE and pull the RTB toward you and off the module.



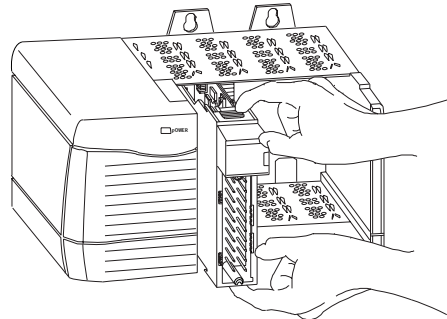
Removing the Module from the Chassis

To remove the module from the chassis:

1. If the RTB is on the module, unlock the RTB and remove it. (Refer to *Removing the Removable Terminal Block from the Module*.)
2. Push in and hold the top and bottom locking tabs on the module.



3. Pull the module out of the chassis.



Getting Started With Your Motion Module

This chapter provides a step-by-step procedure for configuring a motion axis and developing a simple application program using one axis.

Note: Before beginning this chapter, complete the *Getting Started* chapter in the Logix5550 Controller User Manual, publication 1756-6.5.12.

The following table shows the contents of this chapter:

| For information about | See page |
|--|----------|
| Understanding the Getting Started Tutorial | 3-2 |
| Setting the Master Coordinated System Time | 3-3 |
| Adding the 1756-M02AE Module | 3-6 |
| Naming an Axis | 3-8 |
| Configuring a Motion Axis | 3-9 |
| Running Hookup Diagnostics and Auto Tuning | 3-19 |
| Entering a Ladder Logic Program | 3-29 |

Before using this tutorial, you should:

- Install your Logix5550 controller (For more information, refer to the Logix5550 Controller User Manual, publication 1756-6.5.12.)
- Install your 1756-M02AE motion module (For more information, refer to Chapter 2 - *Installing Your Motion Module*.)
- Complete the *Getting Started* chapter in the Logix5550 Controller User Manual, publication 1756-6.5.12
- Ensure your application is offline. (If your application is online, select **Go Offline** from the Communication menu.)

Understanding the Getting Started Tutorial

This tutorial guides you through all the steps in developing a simple motion control application with one axis. For this tutorial, you will use the following control system components:

- One 4-slot chassis and power supply
- One 1756-M02AE servo module (installed in slot 0)
- One 1756-IB16 input module (installed in slot 1)
- One 1756-OB16E output module (installed in slot 2)
- One Logix5550 controller (installed in slot 3)
- RSLogix 5000 programming software

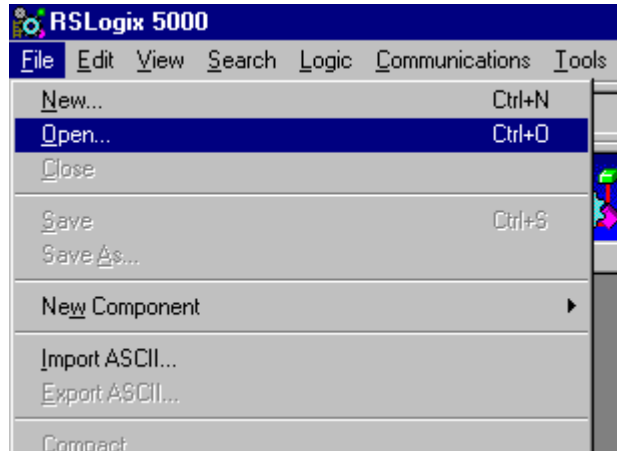
During this tutorial, you will create a motion application program by completing the following tasks:

| Task | Description |
|---|---|
| Select the master coordinated system time | Sets one controller as the master controller. Once you complete this step, you can synchronize all the motion modules and Logix5550 controllers in your chassis |
| Add a motion module | Adds a motion module to your application program |
| Name an axis | Adds an axis to your application program |
| Configure an axis | Configures each axis for motion control |
| Run hookup diagnostics and auto tuning | Completes hookup diagnostics and auto tuning for each axis |
| Develop a motion application program | Create a program for your motion control application |

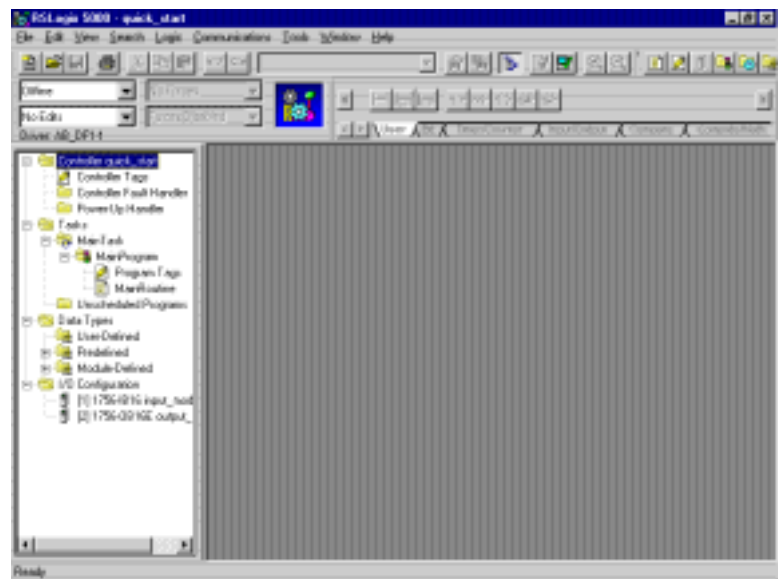
Setting the Master Coordinated System Time

To select the master coordinated system time:

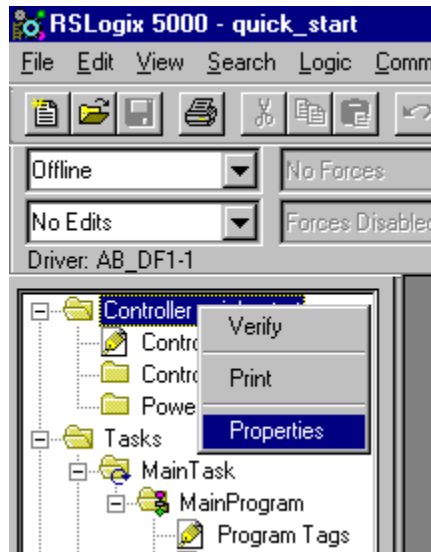
1. In the File menu of the RSLogix 5000 programming software, select **Open**.



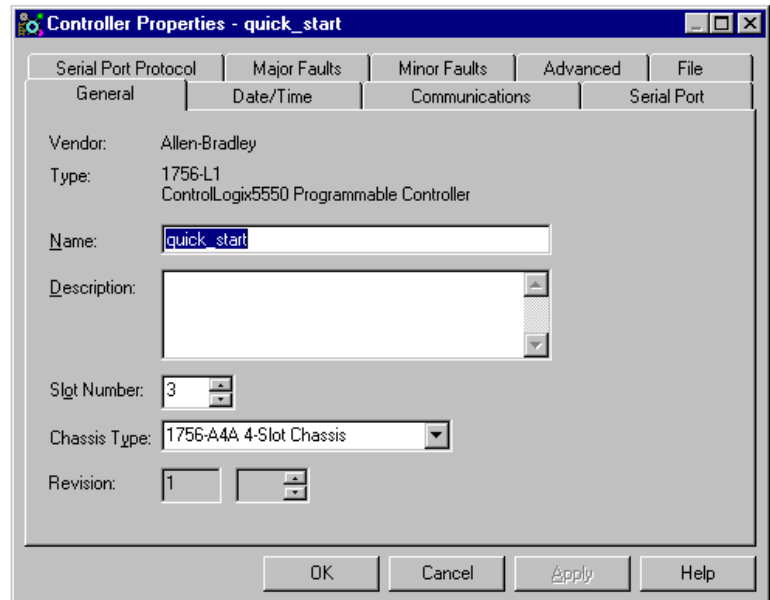
2. Select **quick_start**, which is the project you created when you completed the *Getting Started* chapter in the Logix5550 Controller User Manual, publication 1756-6.5.12. The following window appears.



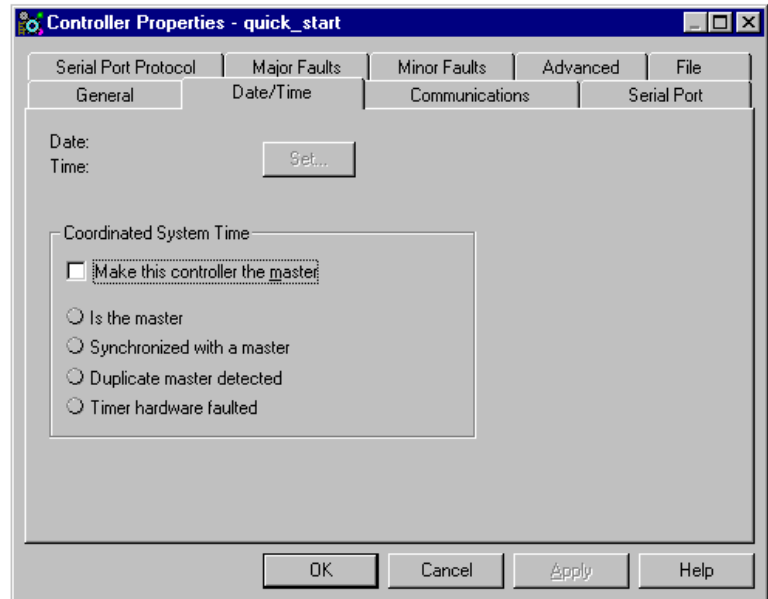
3. Right-click the Controller folder.



4. Select **Properties**. The Controller Properties window appears.



5. Select the Date/Time tab. The following window appears.

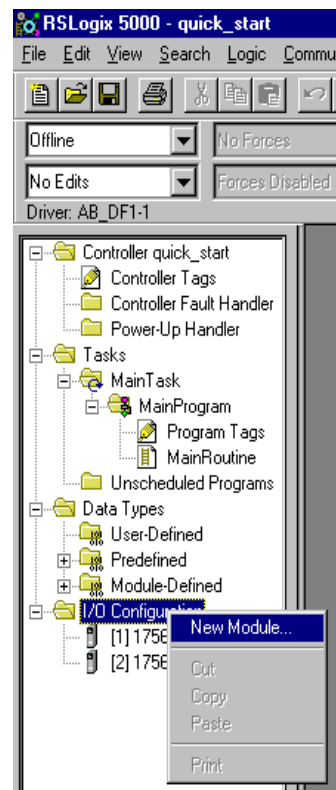


6. Select **Make this controller the master**.
7. Select **OK**.

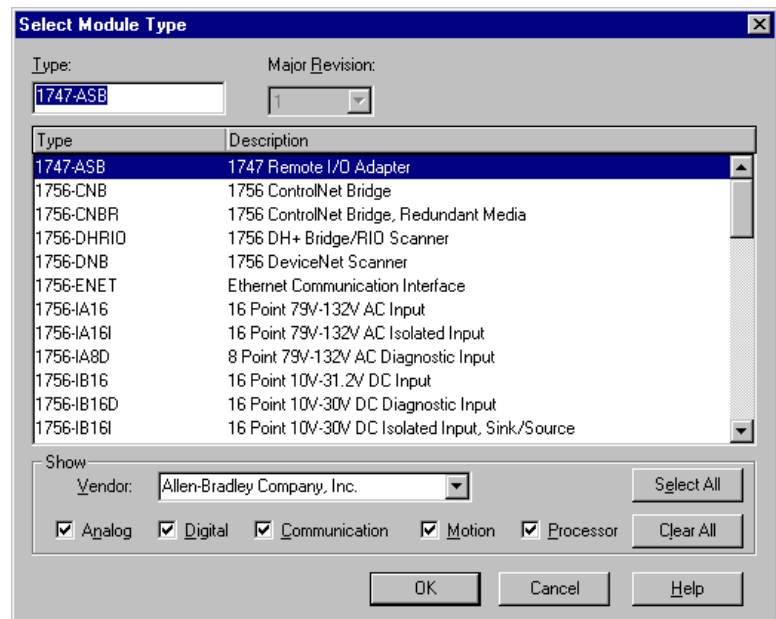
Adding the 1756-M02AE Module

To add a motion module:

1. Right-click the I/O Configuration folder.



2. Select **New Module**. The Select Module Type window appears.



3. In the *Type* field, select **1756-M02AE 2 Axis Analog/Encoder Servo**.
4. Select **OK**. The New Module window appears.

Module Properties - Local (1756-M02AE 1.1)

Type: 1756-M02AE 2 Axis Analog/Encoder Servo
 Vendor: Allen-Bradley Company, Inc.
 Parent: Local
 Name: Slot: 0
 Description:
 Associated Axes:
 Channel 0:
 Channel 1:
 Revision: Electronic Keying:

5. Make entries in the following fields.

| Field | Entry |
|-------------------|----------------|
| Name | Servocard |
| Slot | 0 |
| Electronic keying | Disable keying |

Naming an Axis

To name an axis:

1. In the New Module window (shown in step 4 of the *Adding the 1756-M02AE Module* section), select **New Axis**. The New Tag window appears.

2. Make an entry in the following field.

| Field | Entry |
|-------|--------|
| Name | Axis_X |

Configuring a Motion Axis

To configure your new axis:

1. In the New Tag window (shown in step 1 of the *Naming an Axis* section), select **Configure**. The Axis Wizard-General window appears.

Axis Wizard Axis_X - General

Module: <none> Channel:

Type: Servo

Positioning Mode: Linear

Help Cancel < Back Next > Finish

2. Make entries in the following fields.

| Field | Entry |
|------------------|--------|
| Type | Servo |
| Positioning Mode | Linear |

3. Select **Next**. The Axis Wizard-Group window appears.

Axis Wizard Axis_X - Group

Assigned Motion Group: [] New Group

Axes Assigned: []

Coarse Rate: 1 ms

Servo Update Period: [] us

General Fault Type: []

Help Cancel < Back Next > Finish

4. Select **New Group**. The New Tag window appears.

5. Make an entry in the following field.

| Field | Entry |
|-------|--------------|
| Name | Motion_Group |

6. Select **Configure**. The Axis Wizard-Axis Assignment window appears.

7. From the *Unassigned* field, select **Axis_X**.
8. Select **Add**.

9. Select **Next**. The Axis Wizard-Update Rates window appears.

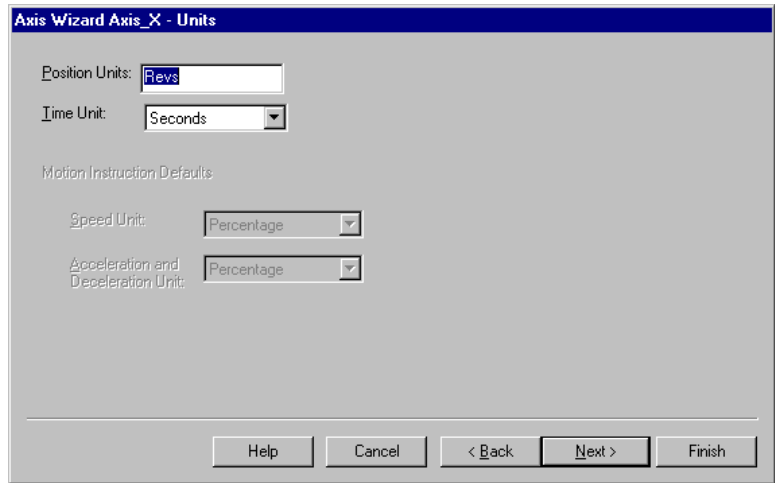
10. Make entries in the following fields.

| Field | Entry |
|---------------------|-----------------|
| Coarse rate | 5 |
| Servo update period | 200 |
| General fault type | Non major fault |

11. Select **Finish**. The Axis Wizard-Group window appears.

12. In the *Assigned Motion Group* field, select **Motion_Group**.

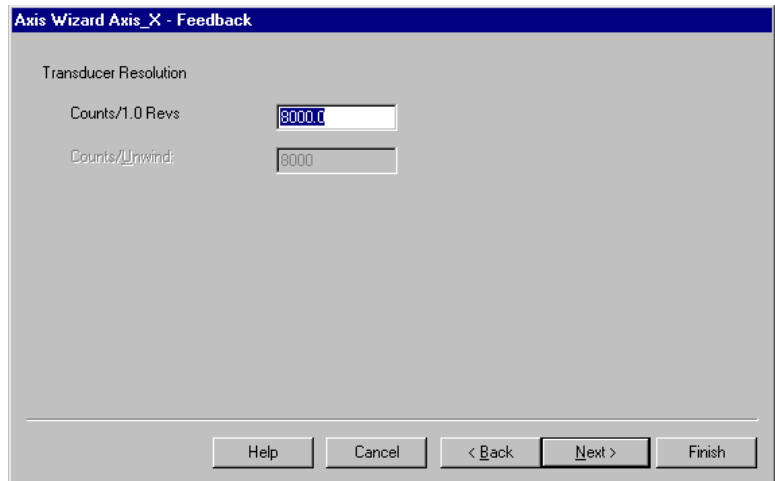
13. Select **Next**. The Axis Wizard-Units window appears.



14. Make entries in the following fields.

| Field | Entry |
|----------------|---------|
| Position units | Revs |
| Time unit | Seconds |

15. Select **Next**. The Axis Wizard-Feedback window appears.



16. Make an entry in the following field.

| Field | Entry |
|-----------------|--------|
| Counts/1.0 revs | 8000.0 |

17. Select **Next**. The Axis Wizard-Positioning window appears.

18. Make entries in the following fields.

| Field | Entry |
|---------------------------|-------|
| Lock tolerance | 0.025 |
| Average velocity timebase | 0.005 |

19. Select **Next**. The Axis Wizard-Homing window appears.

20. Make entries in the following fields.

| Field | Entry |
|-------------------|---------------------|
| Home position | 0.0 |
| Mode | Active |
| Sequence | Home to marker only |
| Homing direction | Negative |
| Homing speed | 1.25 |
| Home return speed | 0.625 |

21. Select **Next**. The Axis Wizard-Overtravels window appears.

Axis Wizard Axis_X - Overtravels

Soft Travel Limits

Overtravels:

Maximum Positive: 0.0 Revs

Maximum Negative: 0.0 Revs

Help Cancel < Back **Next >** Finish

22. Do not make any entries in this window.

23. Select **Next**. The Axis Wizard-Servo window appears.

Axis Wizard Axis_X - Servo

Drive Type: Velocity

Enable Drive Fault Input

Drive Fault Input: Normally Closed

Output Limit: 10.0 V

Enable Servo Update:

Position Error Velocity Error Velocity Command Servo Output Level

Position |Error Velocity |Error Velocity Feedback

Help Cancel < Back **Next >** Finish

24. Make entries in the following fields.

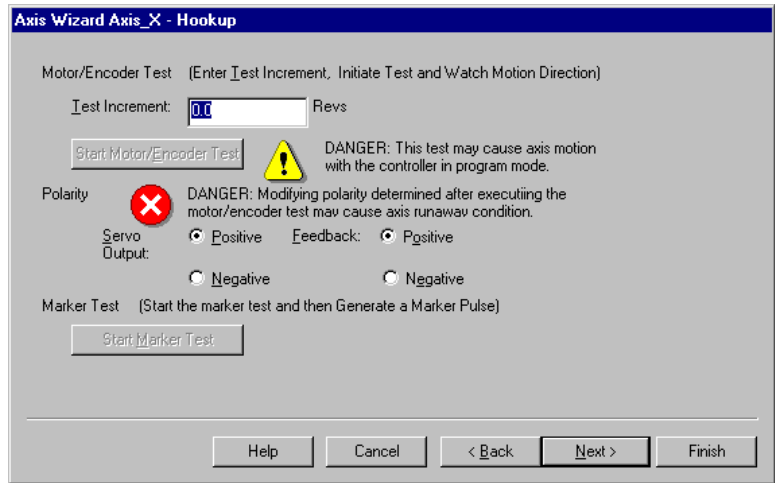
| Field | Entry |
|--------------------------|--|
| Drive type | Torque |
| Enable drive fault input | Select the checkbox |
| Drive fault input | Normally closed |
| Output limit | 10.0 |
| Enable servo update | <ul style="list-style-type: none"> • Position error • Velocity command • Servo output level |

25. Select **Next**. The Axis Wizard-Fault Action window appears.

26. Make entries in the following fields.

| Field | Entry |
|------------------------|---------------|
| Position error | Disable drive |
| Drive fault | Disable drive |
| Transducer noise | Status only |
| Transducer loss | Stop motion |
| Programmed stop action | Fast stop |

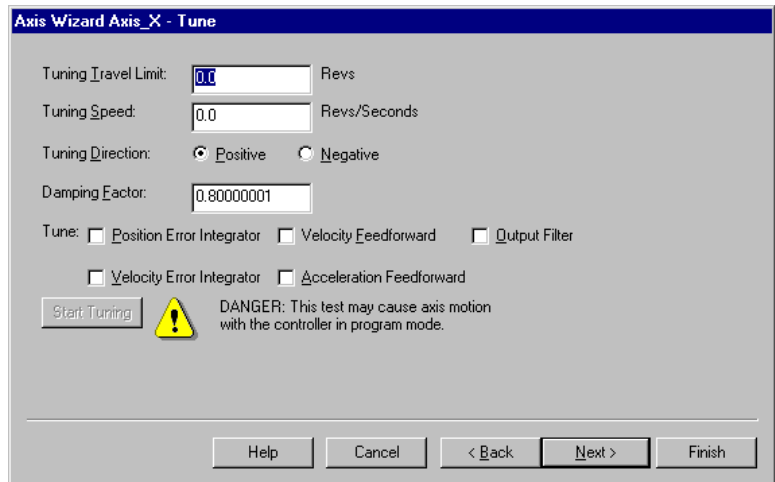
27. Select **Next**. The Axis Wizard-Hookup window appears.



28. Make entries in the following fields.

| Field | Entry |
|----------------|----------|
| Test increment | 5.0 |
| Servo output | Positive |
| Feedback | Positive |

29. Select **Next**. The Axis Wizard-Tune window appears.



30. Make entries in the following fields.

| Field | Entry |
|---------------------|------------|
| Tuning travel limit | 100.0 |
| Tuning speed | 20.0 |
| Tuning direction | Positive |
| Damping factor | 0.80000001 |

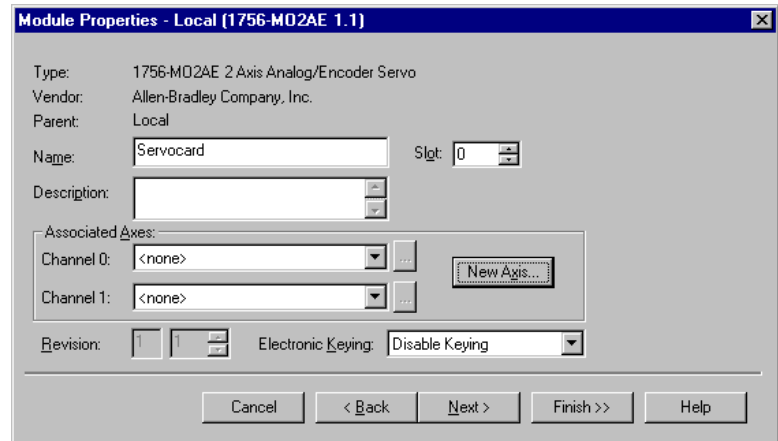
31. Select **Next**. The Axis Wizard-Gains window appears.

32. Do not make any entries in this window.

33. Select **Next**. The Axis Wizard-Dynamics window appears.

34. Do not make any entries in this window.

35. Select **Finish**. The Module Properties window appears.



36. In the *Channel 0* field, select **Axis_X**.

37. Select **Finish**. The Module Properties window will close.

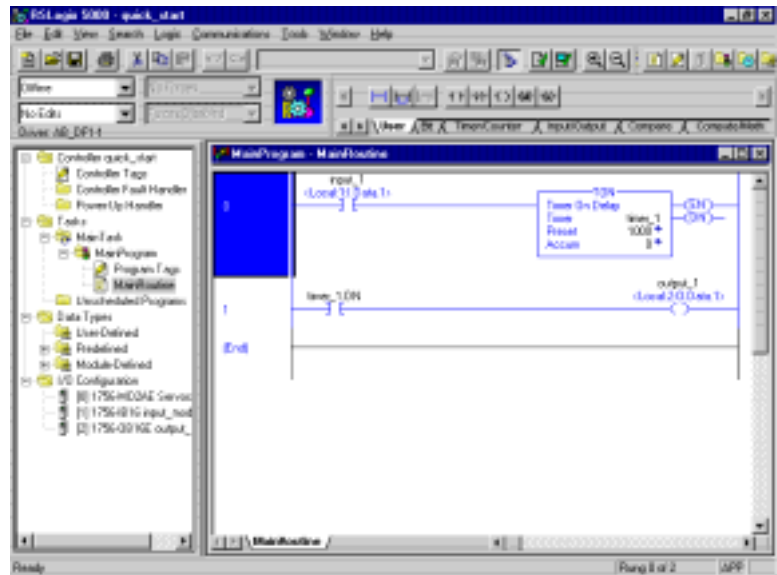
Running Hookup Diagnostics and Auto Tuning

Once you have added and configured your motion module and axis, you can run hookup diagnostics and auto tuning. To run diagnostics and tuning, you must download a program and go online.

Important: This section assumes that you have connected a drive to channel 0 of the 1756-M02AE module in slot 0. See Appendix B - *Loop and Interconnect Diagrams* for wiring information.

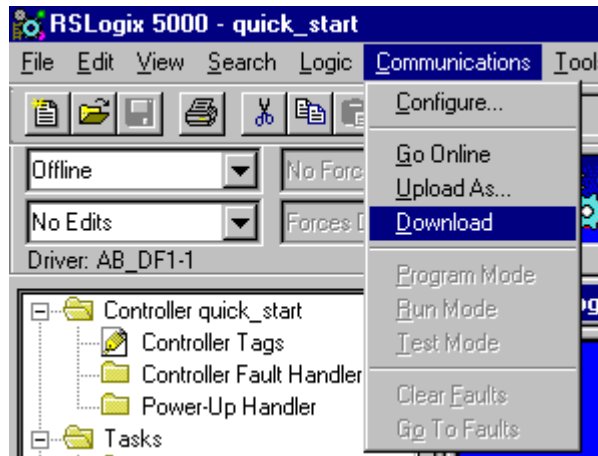
1. Double-click **Main Routine**. The following window appears.

Note: This section assumes you have completed the *Getting Started* chapter in the Logix5550 Controller User Manual, publication 1756-6.5.12.

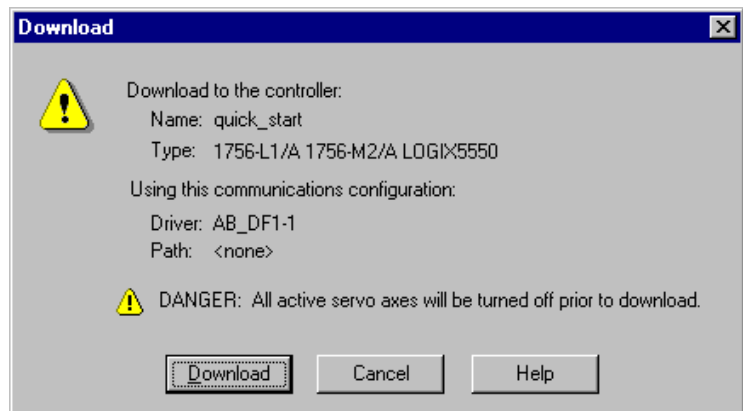


2. Make sure the keyswitch is in the REM position.

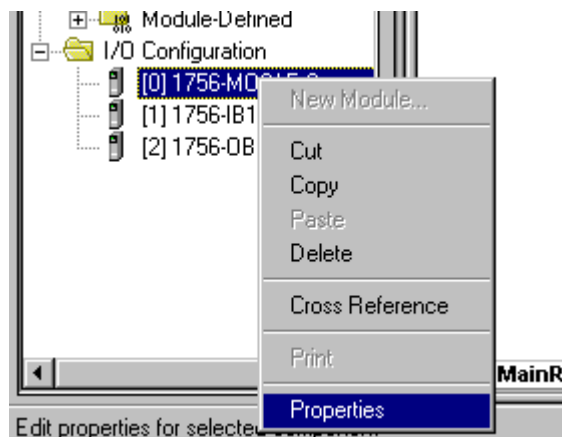
- From the Communications menu, select **Download**.



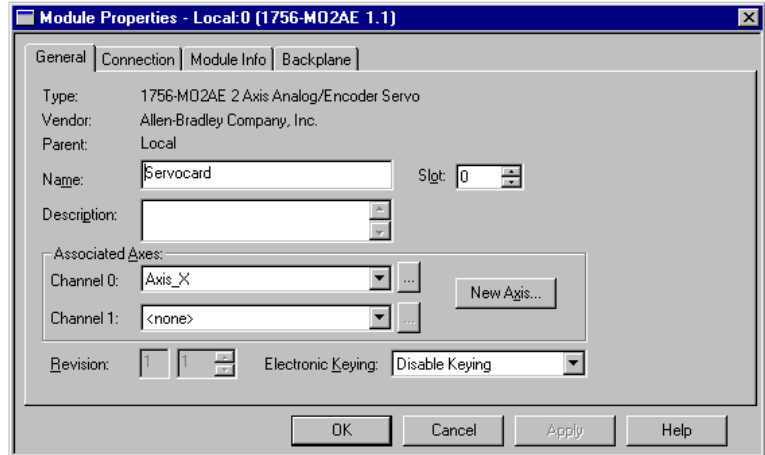
The following window appears.




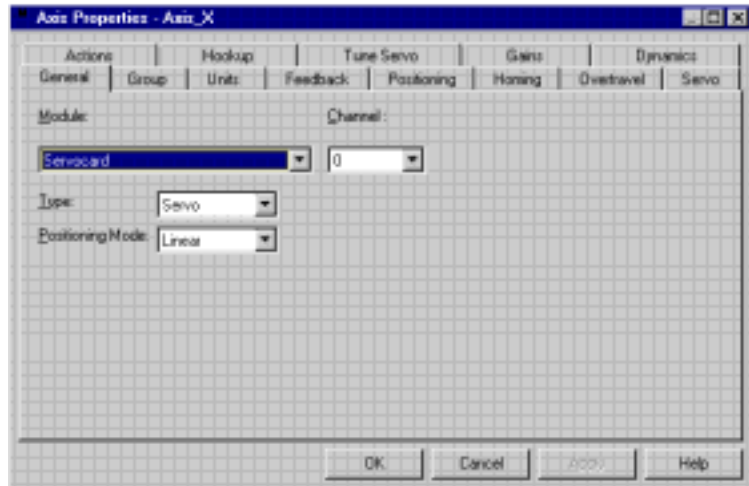
- Select **Download**.
- Under the I/O Configuration folder, right-click the 1756-M02AE module Servocard.



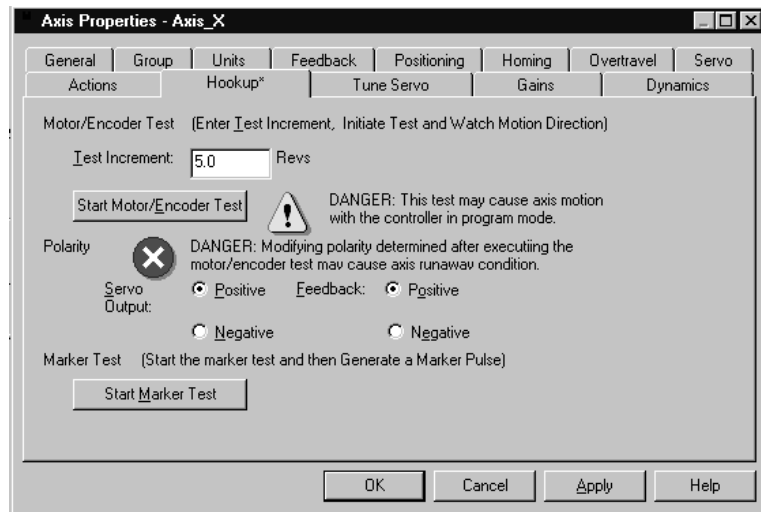
6. Select **Properties**. The Module Properties window appears.



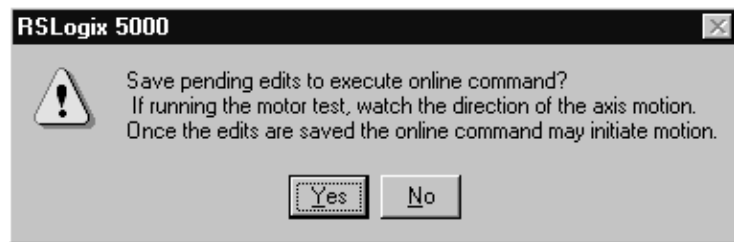
7. Next to the *Channel 0* field, select the  button. The Axis Properties window appears.



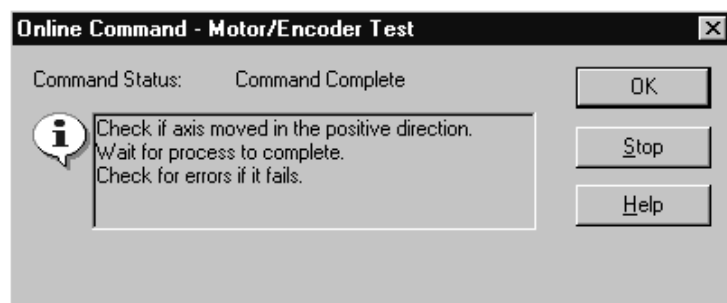
8. Select the Hookup tab. The following window appears.



9. Select **Start Motor/Encoder Test**. The following window appears.

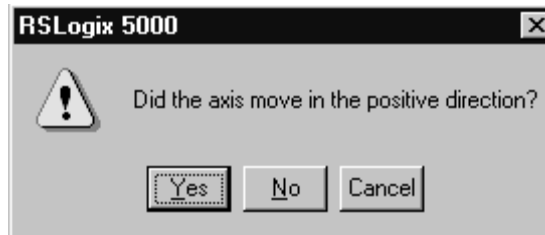


10. Select **Yes**. The following window appears.

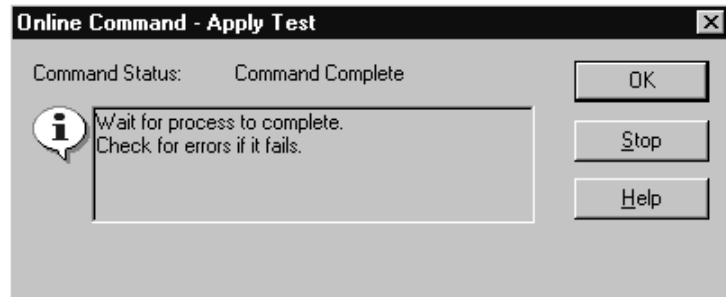


11. Watch the motor to see which way it turns.

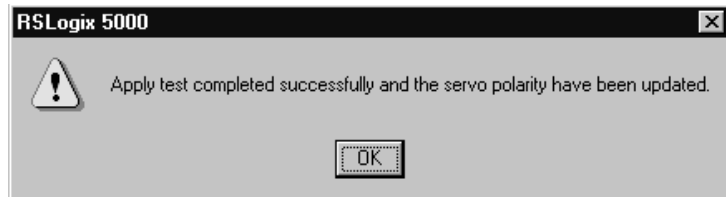
12. Select **OK**. The following window appears.



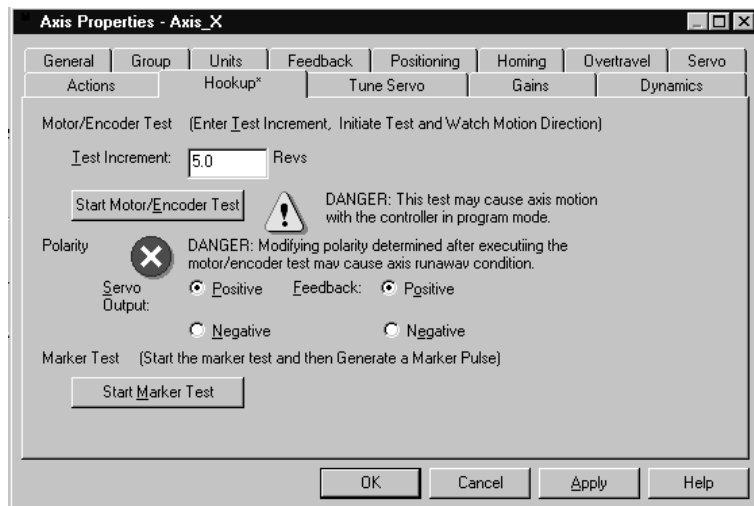
13. Select **Yes**. The following window appears.



14. Select **OK**. The following window appears.



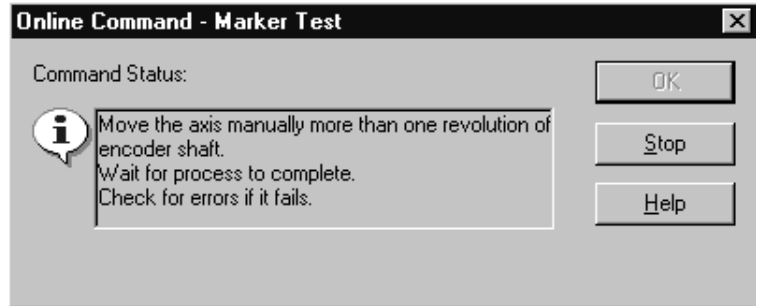
15. Select **OK**. The Axis Properties window appears.



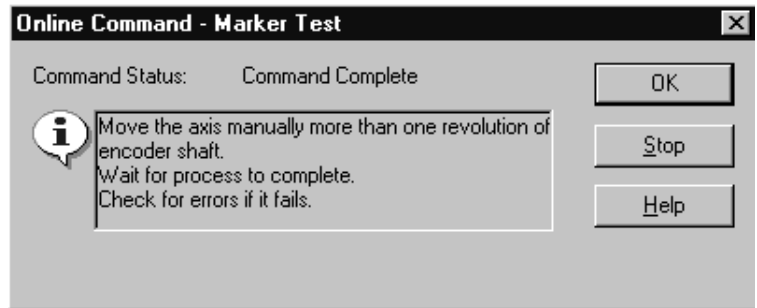
16. Select **Start Marker Test**. The following window appears.



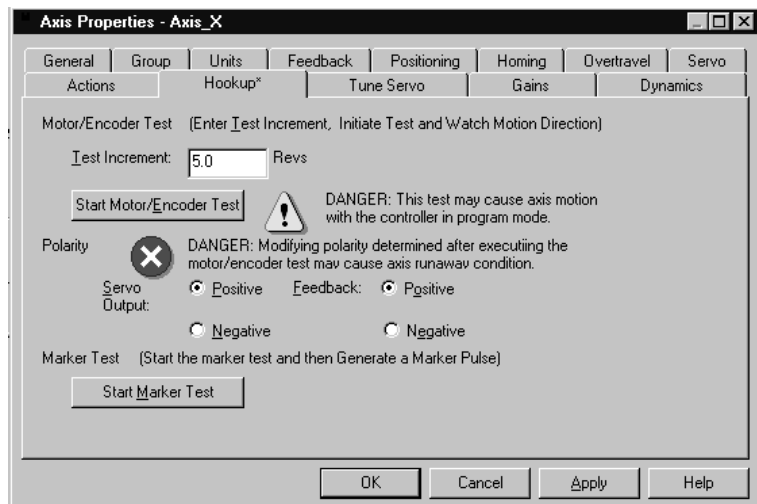
17. Select **Yes**. The following window appears.



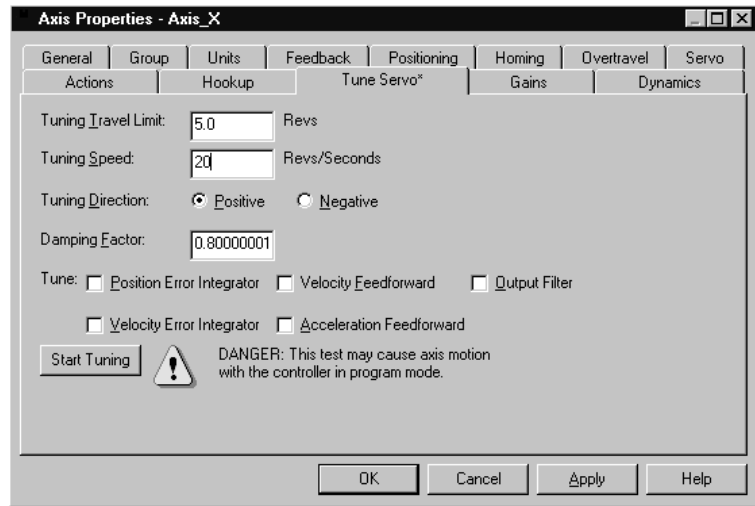
18. Slowly rotate the motor axis until the following window appears.



19. Select **OK**. The Axis Properties window appears.



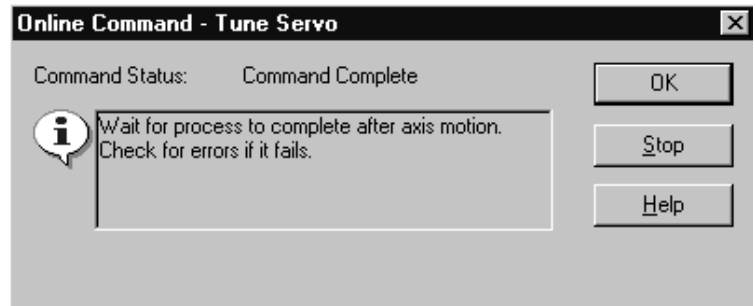
20. Select the Tune Servo tab. The following window appears.



21. Select **Start Tuning**. The following window appears.



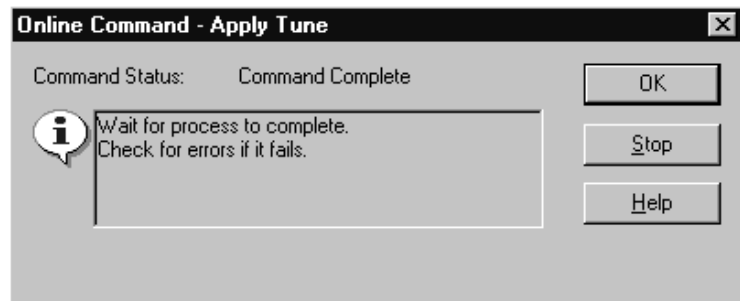
22. Select **Yes**. The following window appears.



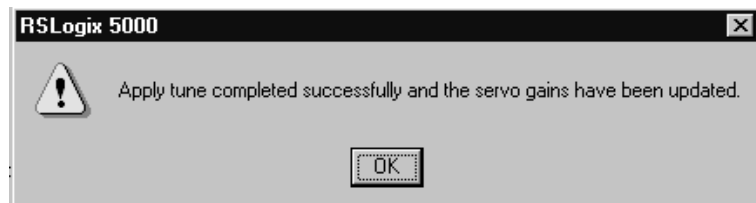
23. Select **OK**. The Tune Bandwidth window appears.



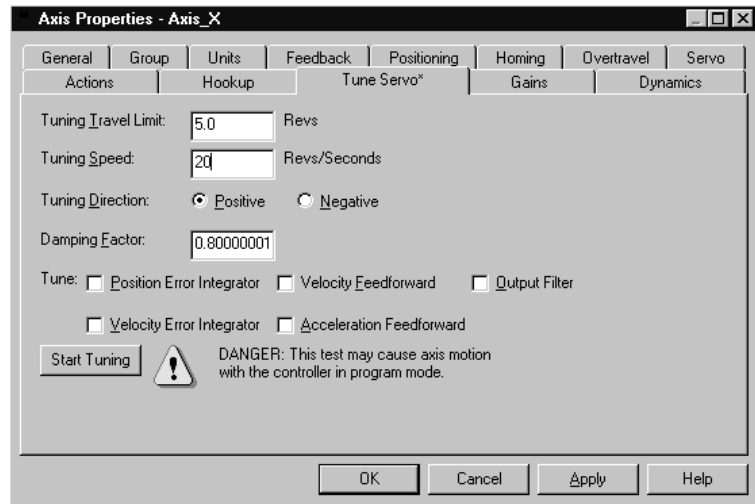
24. Select **OK**. The following window appears.



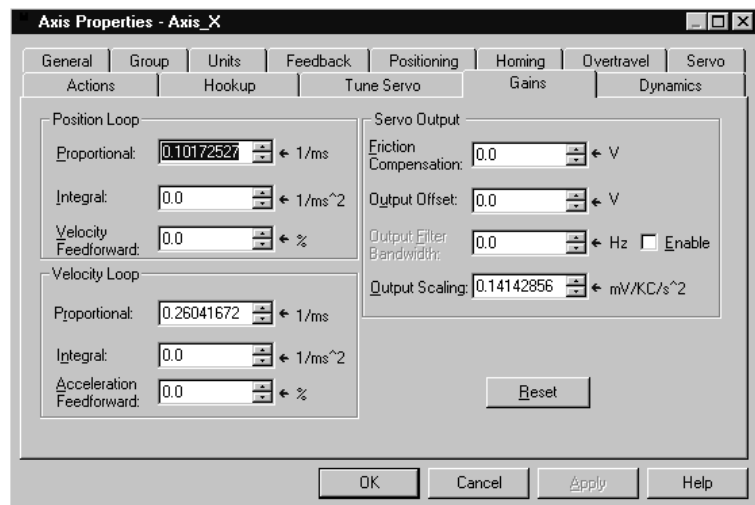
25. Select **OK**. The following window appears.



26. Select **OK**. The Axis Properties window appears.

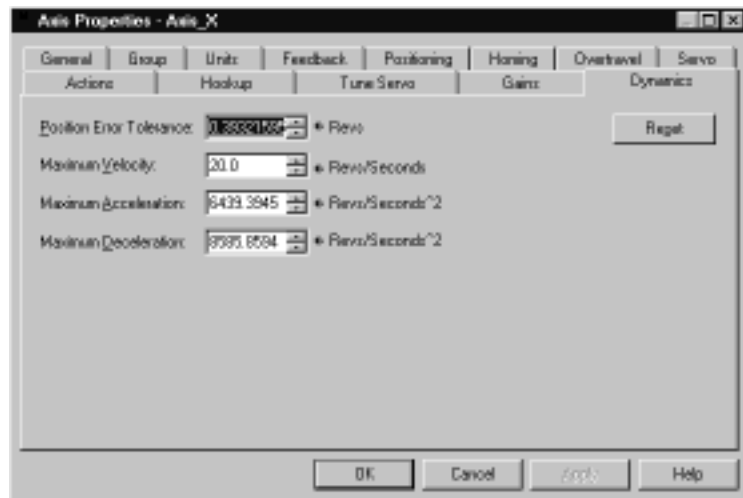


27. Select the Gains tab. The following window appears.



The window will show values for the position loop, velocity loop, and output compensation.

28. Select the Dynamics tab. The following window appears.



This window will show values for maximum velocity, error tolerance, maximum acceleration, and maximum deceleration.

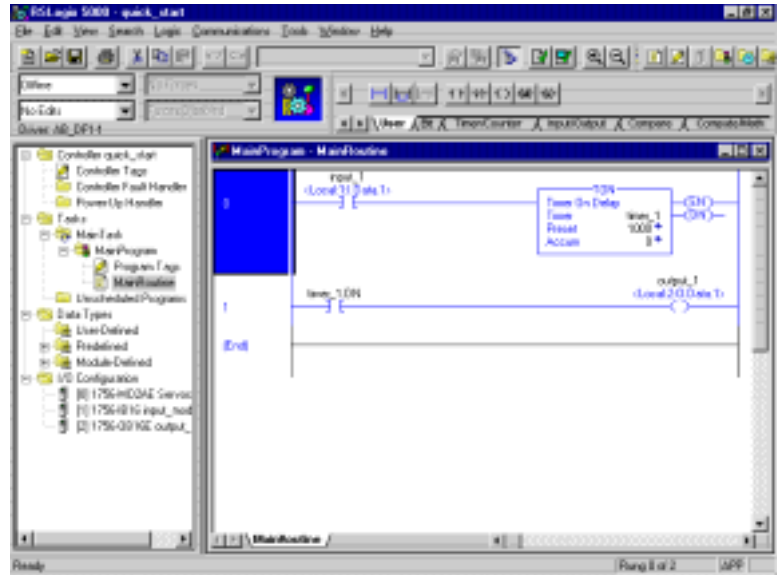
29. Select **OK**. The Axis Properties window will close.



This completes the configuration of Axis_X. You can use Axis_X for motion instructions within your application program.

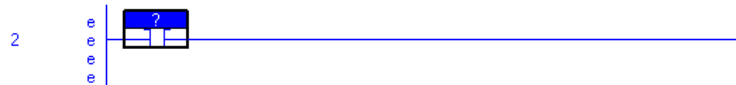
Entering a Ladder Logic Program

After completing all the motion control configuration, you can begin to enter your application program. To enter a ladder logic program:

1. From the Communications menu, select **Go Offline**.
2. Double-click **Main Routine**. The following window appears.

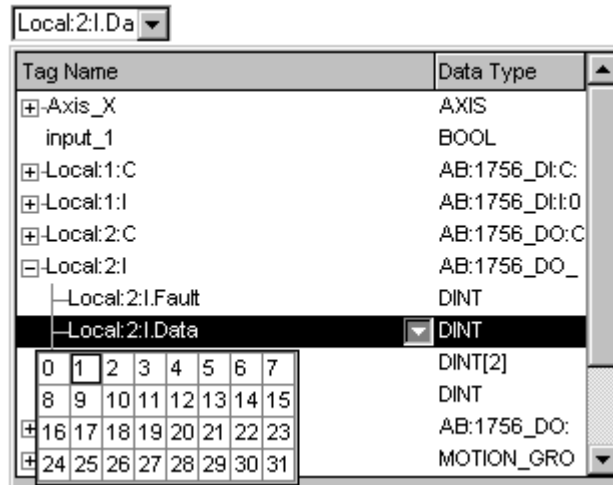


3. To add rung 2, select the  button.
4. To add an XIC to rung 2, select the  button in the User instructions. Rung 2 should look like the following.

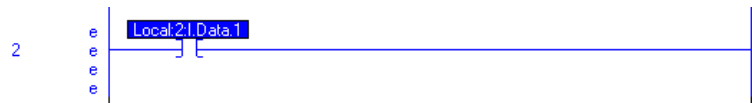


5. Double-click the question mark.

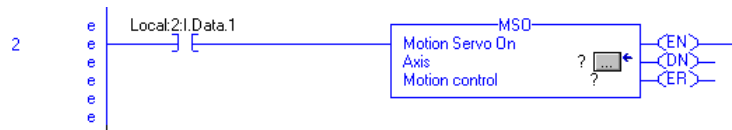
6. Select the down arrow. The following window appears.



7. Select **Local:2:I.Data.1**. Rung 2 should look like the following.

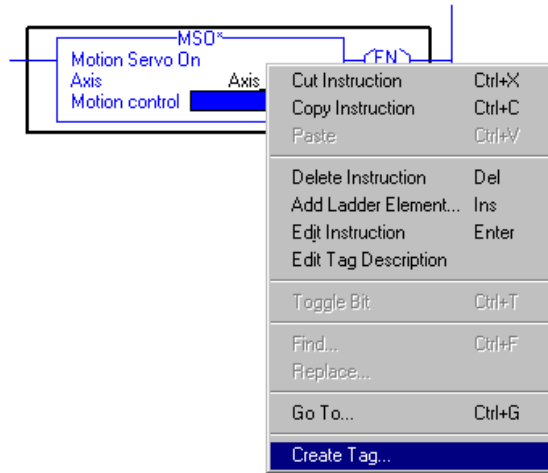


8. To add an MSO instruction to rung 2, select the **MSO** button from the Motion State instructions. Rung 2 should look like the following.

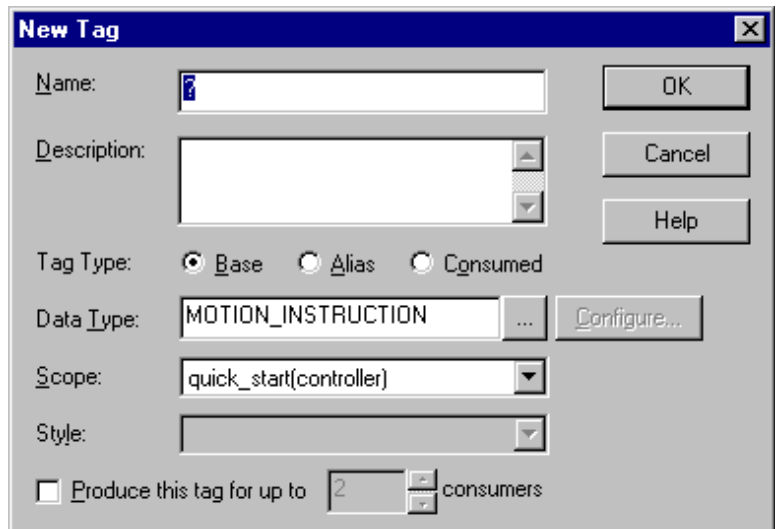


9. Next to the *Axis* field, double-click on the question mark.
10. Select **Controller Tags**.
11. Double-click **Axis_X**.

12. Right-click the *Motion Control* field.



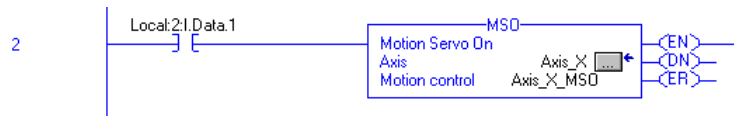
13. Select **Create Tag**. The following window appears.



14. Make entries in the following fields.

| Field | Entry |
|-------|------------|
| Name | Axis_X_MSO |

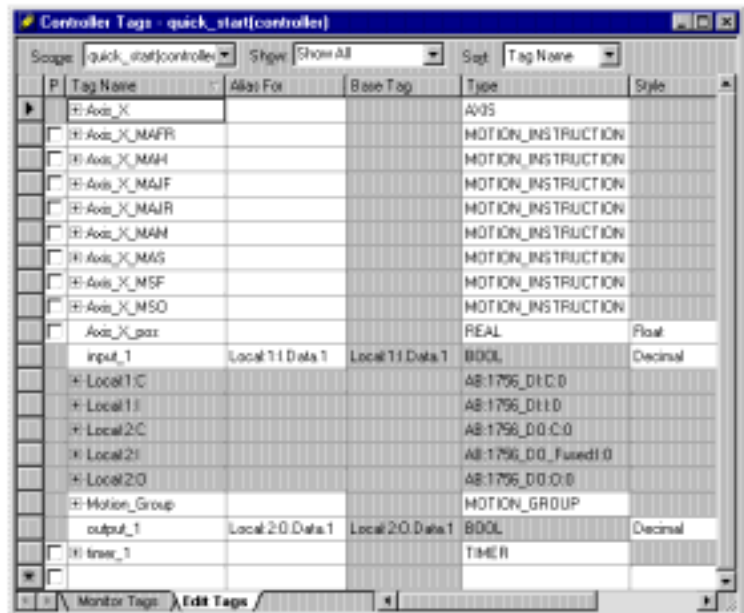
15. Select **OK**. Rung 2 should look like the following.



3. Add the following tags.

| Tag Name | Type | Style |
|-------------|--------------------|-------|
| Axis_X_MAFR | MOTION_INSTRUCTION | NA |
| Axis_X_MAH | MOTION_INSTRUCTION | NA |
| Axis_X_MAJF | MOTION_INSTRUCTION | NA |
| Axis_X_MAJR | MOTION_INSTRUCTION | NA |
| Axis_X_MAM | MOTION_INSTRUCTION | NA |
| Axis_X_MAS | MOTION_INSTRUCTION | NA |
| Axis_X_MSF | MOTION_INSTRUCTION | NA |
| Axis_X_MSO | MOTION_INSTRUCTION | NA |
| Axis_X_pos | REAL | Float |

When you close and re-open the Tag Editor, your Tag Editor window should look like the following.



Completing Your Application Program

After you create all the tags for your program, you can add the remaining rungs and instructions. To complete your application program:

1. Enter the following rungs of logic.

Rung 2:

When input 1 has a positive transition, the controller turns the Axis_X servo on. This instruction closes the position loop and activates the drive enable output.



Rung 3:

When input 2 has a positive transition, the controller turns the Axis_X servo off. This instruction opens the position loop and deactivates the drive enable output.



Rung 4:

When input 3 has a positive transition, the controller homes Axis_X



Rung 5:

When input 4 has a positive transition, the controller jogs Axis_X in the forward direction with a speed of 5.0.



Rung 6:

When input 5 has a positive transition, the controller jogs Axis_X in the reverse direction with a speed of 5.0.



Rung 7:

When input 4 or input 5 has a negative transition, the controller stops the jog on Axis_X.



Rung 8:

When input 6 has a positive transition, the controller moves Axis_X an incremental distance of 10.0 at a speed of 5.0.

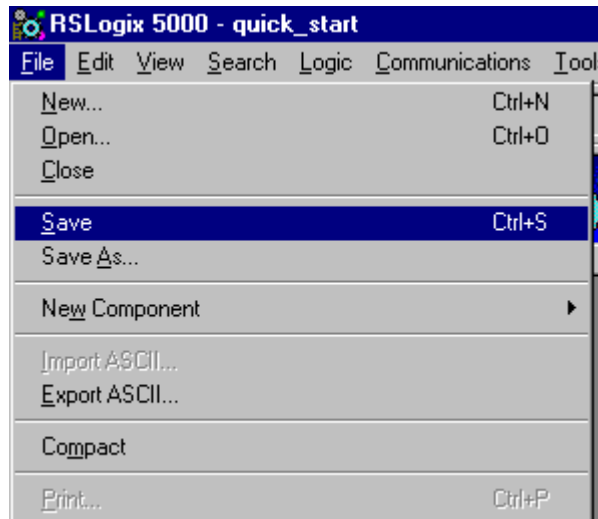


Rung 9:

The controller reads the Axis_X actual position attribute and stores its value in Axis_pos.



2. From the File menu, select **Save**.



Once you have created and saved your program, you can download it to your controller and test its operation. For more information about downloading and testing your program, refer to the Logix5550 Controller User Manual, publication 1756-6.5.12.

Adding and Configuring Your Motion Module

This chapter describes how to add and configure your motion module for use in your motion control application. The following table shows the contents of this chapter:

| For information about | See page |
|---|----------|
| Understanding Application Program Development | 4-1 |
| Selecting the Master Coordinated System Time | 4-2 |
| Adding the 1756-M02AE Module | 4-5 |
| Naming an Axis | 4-7 |
| Configuring a Motion Axis | 4-8 |
| Assigning Additional Motion Modules and Axes | 4-27 |
| Running Hookup Diagnostics and Auto Tuning | 4-28 |
| Developing a Motion Application Program | 4-38 |
| Understanding a Programming Example | 4-39 |

Understanding Application Program Development

Developing a motion control application program involves the following:

| Task | Description |
|---|---|
| Select the master coordinated system time | Sets one controller as the master controller. Once you complete this step, you can synchronize all the motion modules and Logix5550 controllers in your chassis |
| Add a motion module | Adds a motion module to your application program |
| Name an axis | Adds an axis to your application program |
| Configure an axis | Configures each axis for motion control |
| Assign additional servo modules and axes | Adds additional modules and axes to your application program |
| Run hookup diagnostics and auto tuning | Completes hookup diagnostics and auto tuning for each axis |
| Develop a motion application program | Create a program for your motion control application |

This chapter will describe each of these tasks.

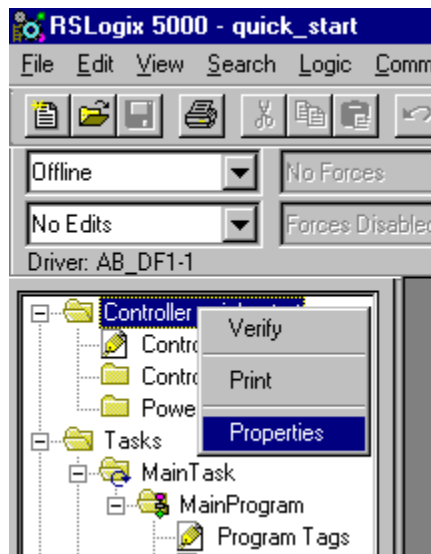
Selecting the Master Coordinated System Time

By selecting the master controller for your application, you can synchronize all the motion modules and Logix5550 controllers in your chassis.

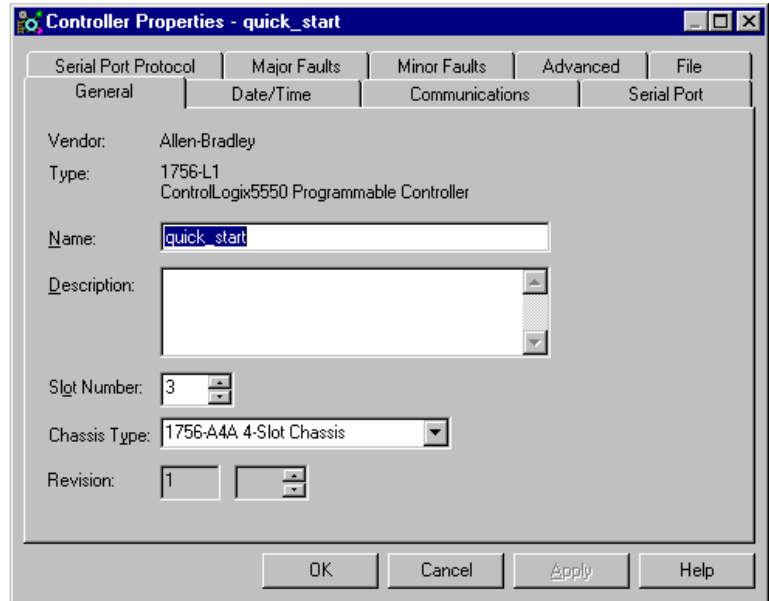
Note: For the motion module to operate correctly, you must select a master controller in each chassis that contains motion modules. Each chassis should contain only one master controller.

To select the master coordinated system time:

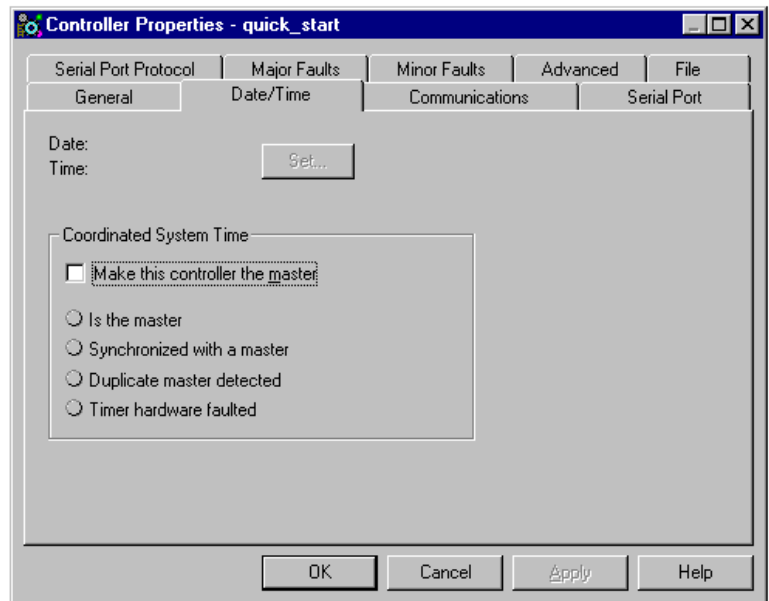
1. Right-click the Controller folder.



2. Select **Properties**. The Controller Properties window appears.



3. Select the Date/Time tab. The following window appears.



4.

| If | And | Then |
|------------------------------------|--|--|
| Your controller uses a motion axis | No other controllers in your chassis are configured as the master controller | <ul style="list-style-type: none">• Select Make this controller the master• Select OK |
| Your controller uses a motion axis | Another controller in your chassis is configured as the master controller | Select OK |

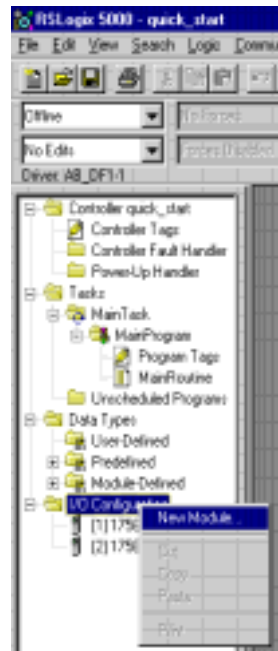
5. Select **OK**.

Adding the 1756-M02AE Module

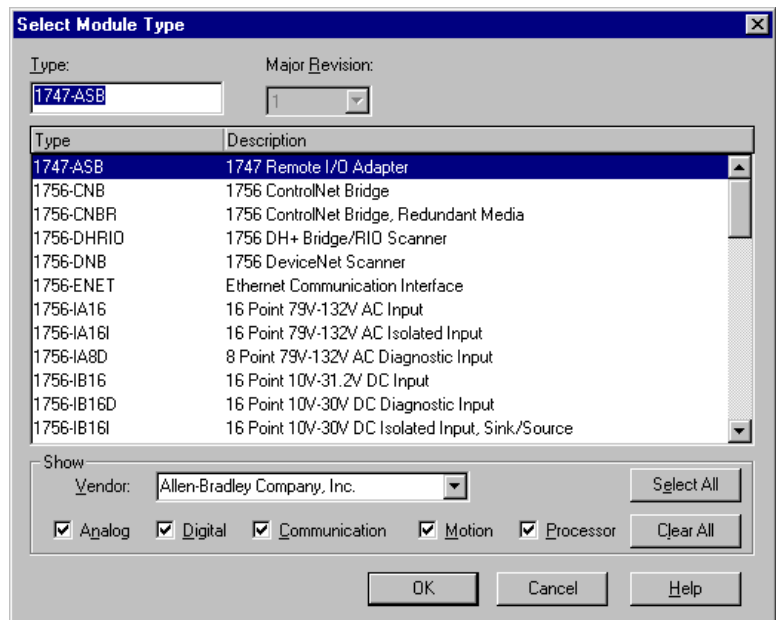
To use your motion module in a control system, you must add your motion module to the application program.

To add a motion module:

1. Right-click the I/O Configuration folder.



2. Select **New Module**. The Select Module Type window appears.



3. In the *Type* field, select **1756-M02AE 2 Axis Analog/Encoder Servo**.

4. Select **OK**. The New Module window appears.



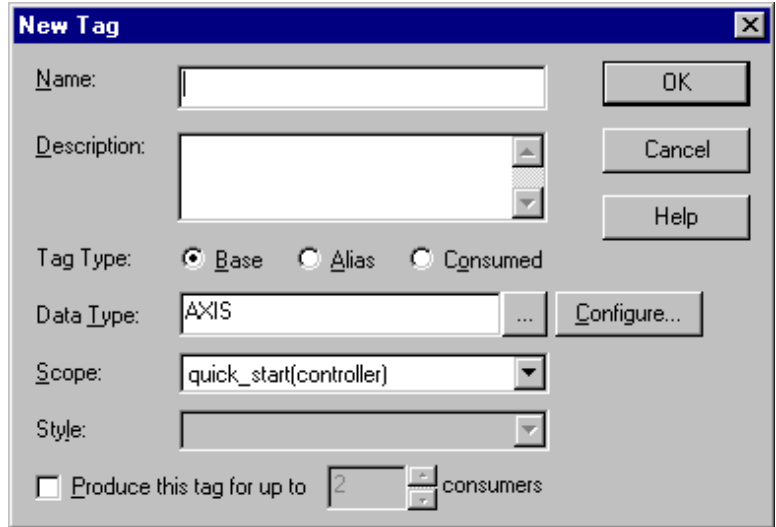
5. Make entries in the following fields.

| Field | Entry | | | | | | | | |
|---|--|--------|--------|---|-------------------|---|----------------|---|-------------|
| Name | Type a name for the servo module. The name can: <ul style="list-style-type: none"> • have a maximum of 40 characters • contain letters, numbers and underscores (_). | | | | | | | | |
| Slot | Enter the number of the chassis slot that contains your module. | | | | | | | | |
| Description | Type a description for your motion module. Note: This field is optional. | | | | | | | | |
| Electronic keying | Select the electronic keying level. | | | | | | | | |
| | <table border="1"> <thead> <tr> <th>To</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Match the vendor, catalog number, and major revision attributes of the physical module and the software configured module</td> <td>Compatible module</td> </tr> <tr> <td>Disable the electronic keying protection mode</td> <td>Disable keying</td> </tr> <tr> <td>Match the vendor, catalog number, major revision, and minor revision attributes of the physical module and the software configured module</td> <td>Exact match</td> </tr> </tbody> </table> | To | Select | Match the vendor, catalog number, and major revision attributes of the physical module and the software configured module | Compatible module | Disable the electronic keying protection mode | Disable keying | Match the vendor, catalog number, major revision, and minor revision attributes of the physical module and the software configured module | Exact match |
| | To | Select | | | | | | | |
| Match the vendor, catalog number, and major revision attributes of the physical module and the software configured module | Compatible module | | | | | | | | |
| Disable the electronic keying protection mode | Disable keying | | | | | | | | |
| Match the vendor, catalog number, major revision, and minor revision attributes of the physical module and the software configured module | Exact match | | | | | | | | |
| | | | | | | | | | |

Naming an Axis

Naming an axis adds it to your application. To name an axis:

1. In the New Module window (shown in step 4 of the *Adding the 1756-M02AE Module* section), select **New Axis**. The New Tag window appears.



2. Make entries in the following fields.

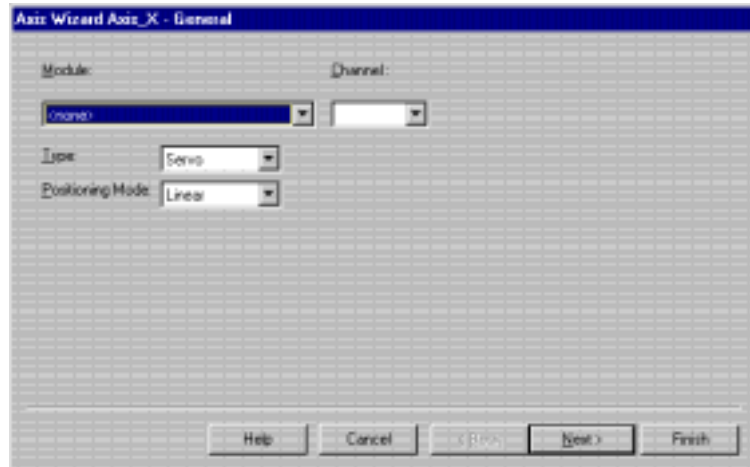
| Field | Entry | | | |
|---------------------------|---|-----------------|--------|---------------------------|
| Name | Type a name for the servo axis. The name can: <ul style="list-style-type: none"> • have a maximum of 40 characters • contain letters, numbers and underscores (_). | | | |
| Description | Type a description for your motion axis. Note: This field is optional. | | | |
| Data type | AXIS | | | |
| Scope | Select the scope of the axis variable. | | | |
| | <table border="1"> <thead> <tr> <th>To use the axis</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Within the entire program</td> <td>Controller</td> </tr> </tbody> </table> | To use the axis | Select | Within the entire program |
| To use the axis | Select | | | |
| Within the entire program | Controller | | | |

Configuring a Motion Axis

To configure your new axis:

Note: When you configure your axis, some fields may be unavailable (greyed-out) because you are using a type of axis, fault, etc.

1. In the New Tag window (shown in step 1 of the *Naming an Axis* section), select **Configure**. The Axis Wizard-General window appears.



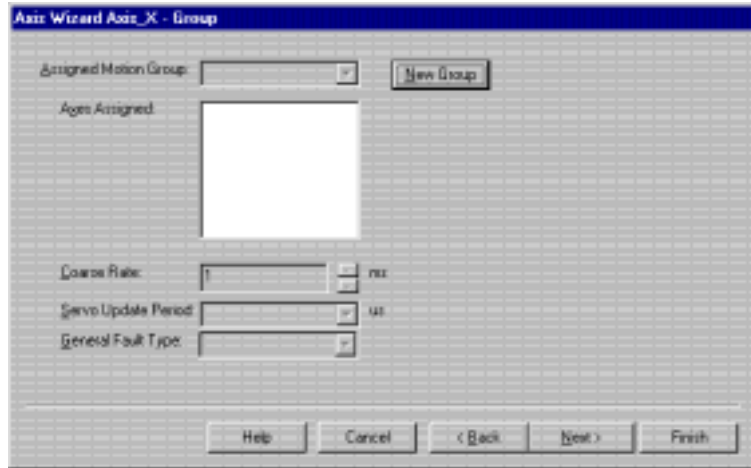
2. Make entries in the following fields.

| Field | Entry | |
|------------------|--|---------------|
| Type | Select the type of axis you are using. | |
| | To use your axis for | Select |
| | Full servo operation | Servo |
| | Monitoring position | Position-only |
| Positioning mode | Select the positioning mode for you module. | |
| | To enable | Select |
| | A maximum linear excursion of one billion encoder counts | Linear |
| | The rotary unwind option of the axis | Rotary |

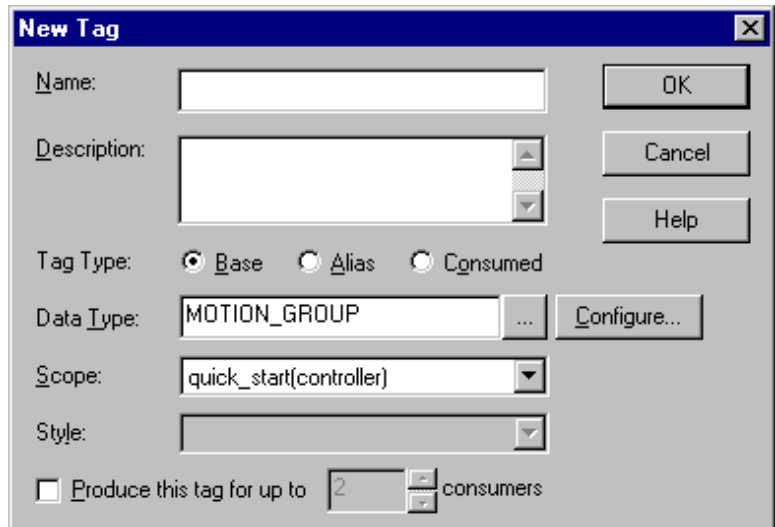
3.

| If | Then |
|---|----------------|
| You have already created a motion group for this axis | Go to step 13. |
| You want to create a new motion group | Go to step 4. |

4. Select **Next**. The Axis Wizard-Group window appears.



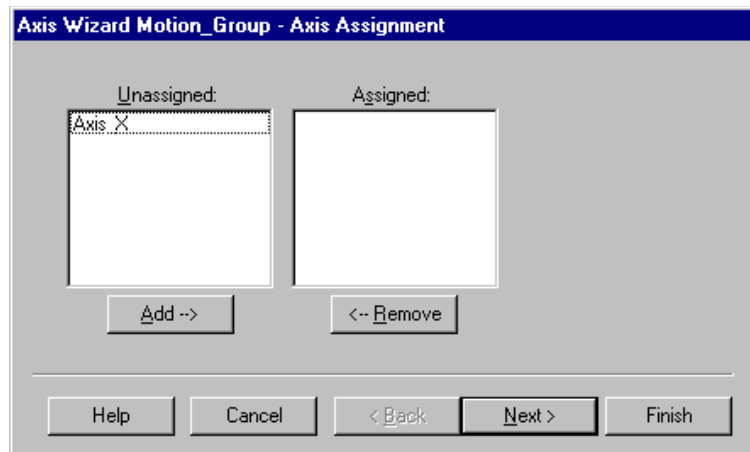
5. Select **New Group**. The New Tag window appears.



6. Make entries in the following fields.

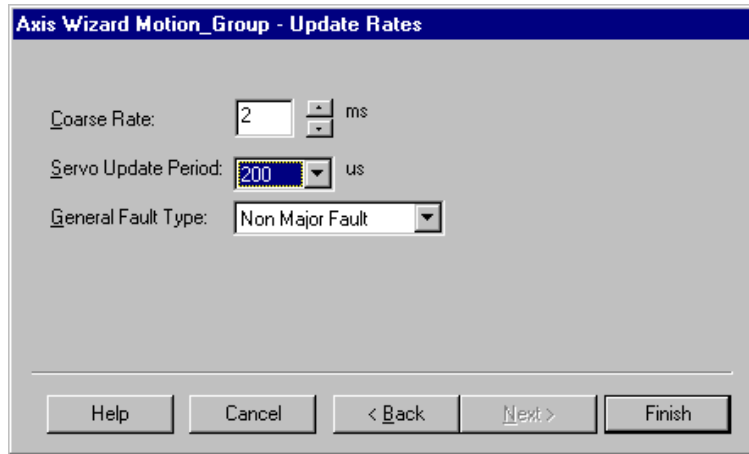
| Field | Entry | | | |
|---------------------------|---|-----------------|--------|---------------------------|
| Name | Type a name for the motion group. The name can: <ul style="list-style-type: none"> • have a maximum of 40 characters • contain letters, numbers and underscores (_). | | | |
| Description | Type a description for your motion group. Note: This field is optional. | | | |
| Data type | MOTION_GROUP | | | |
| Scope | Select the scope of the axis variable. | | | |
| | <table border="1"> <thead> <tr> <th>To use the axis</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Within the entire program</td> <td>Controller</td> </tr> </tbody> </table> | To use the axis | Select | Within the entire program |
| To use the axis | Select | | | |
| Within the entire program | Controller | | | |

7. Select **Configure**. The Axis Wizard-Axis Assignment window appears.



8. From the *Unassigned* field, select your axis.
9. Select **Add**.

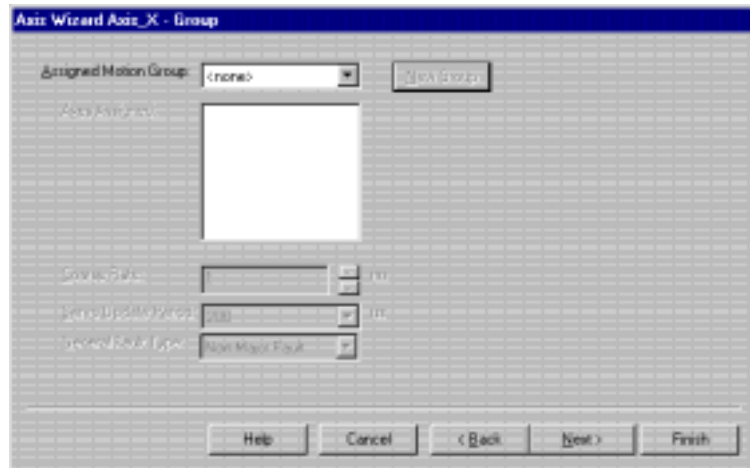
10. Select **Next**. The Axis Wizard-Update Rates window appears.



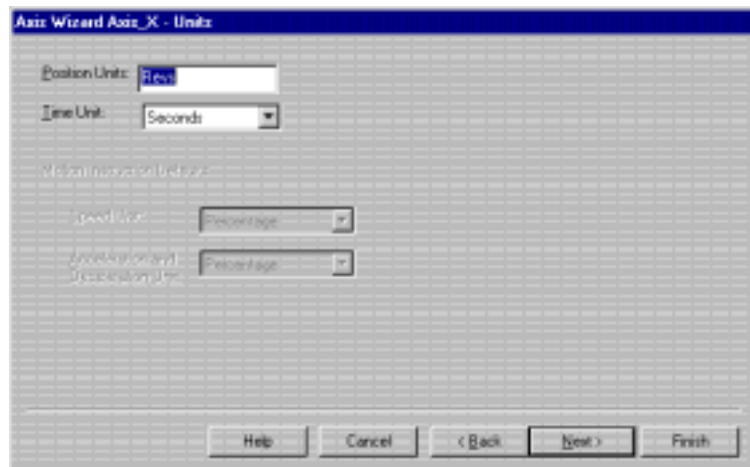
11. Make entries in the following fields.

| Field | Entry | |
|---------------------|---|-----------------|
| Coarse rate | Type the coarse update rate for the motion group. | |
| Servo update period | Select the update period for your motion group. | |
| | For | Select |
| | 200 μ s update rate | 200 |
| General fault type | Select the type of fault for group faults. | |
| | To classify group faults | Select |
| | As minor faults | Non major fault |
| | As major faults | Major fault |

12. Select **Finish**. The Axis Wizard-Group window appears.



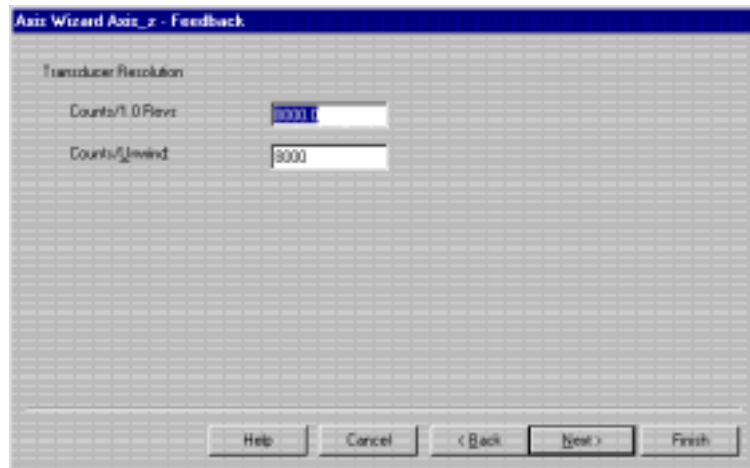
13. In the *Assigned Motion Group* field, select your motion group.
14. In the Coarse Rate field, type the coarse update rate based on the number of axes in your application. For more information about coarse update rates, refer to Appendix A - *Specifications and Performance*.
15. Select **Next**. The Axis Wizard-Units window appears.



16. Make entries in the following fields.

| Field | Entry | |
|------------------------------------|--|----------------------------|
| Position units | Type the units for your axis. For example, a linear axis may use inches, meters, etc. | |
| Time unit | Select the time unit for your axis. | |
| | To use | Select |
| | Seconds as the time unit | Seconds |
| Speed unit | Select the velocity unit for your axis. | |
| | To use | Select |
| | A percentage of the maximum velocity | Percentage |
| | The actual velocity | Units per sec |
| Acceleration and deceleration unit | Select the acceleration and deceleration units. | |
| | To use | Select |
| | A percentage of the maximum acceleration and deceleration | Percentage |
| | The actual acceleration and deceleration rates | Units per sec ² |

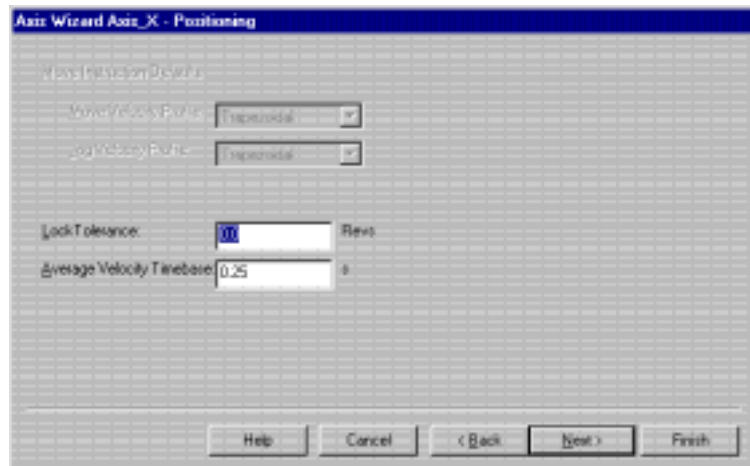
17. Select **Next**. The Axis Wizard-Feedback window appears.



18. Make entries in the following fields.

| Field | Entry |
|-----------------|---|
| Counts/1.0 revs | <p>Type the number of transducer counts per axis position unit. This value allows the conversion of encoder counts into axis position units.</p> <p>For example, an axis uses a 1000-line encoder coupled directly to a 5-pitch lead screw (5 turns per inch). The counts/1.0 revs value is:</p> $\frac{1000 \text{ lines}}{\text{rev}} \times \frac{4 \text{ counts}}{\text{line}} \times \frac{5 \text{ revs}}{\text{inch}} = \frac{20000 \text{ counts}}{\text{inch}}$ |
| Counts/unwind | <p>Type the position unwind value. This value allows the controller to perform an automatic electronic unwind of a rotary axis.</p> <p>For example, a rotary axis uses degrees as units. There are 10 encoder counts per degree. The counts/unwind value is:</p> $10 \text{ counts} \times 360 \text{ degrees} = 3600 \text{ encodercounts}$ |

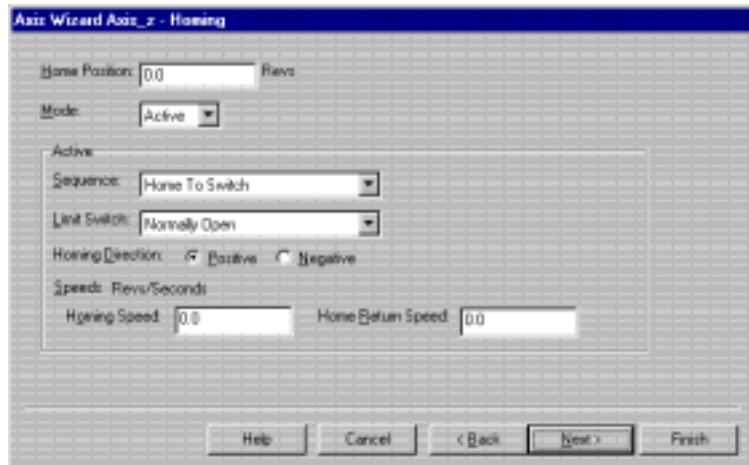
19. Select **Next**. The Axis Wizard-Positioning window appears.



20. Make entries in the following fields.

| Field | Entry | |
|---------------------------|--|----------------|
| Move velocity profile | Select the move profile for your axis. | |
| | To use: | Select: |
| | <ul style="list-style-type: none"> Linear acceleration The fastest acceleration and deceleration rates | Trapezoidal |
| Jog velocity profile | Select the jog profile for your axis. | |
| | To use: | Select: |
| | <ul style="list-style-type: none"> Controlled jerk Least motor stress | S-curve |
| Lock tolerance | Type the allowable position error the servo module will tolerate when giving a true position locked status indication. | |
| Average velocity timebase | Type the time in seconds for calculating the average velocity of your axis. | |

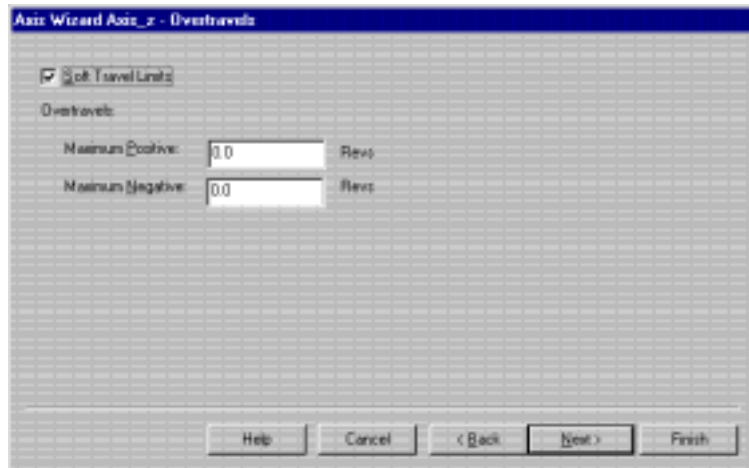
21. Select **Next**. The Axis Wizard-Homing window appears.



22. Make entries in the following fields.

| Field | Entry | |
|-------------------|--|----------------------------|
| Home position | Type the absolute position for the axis after a homing sequence completes. | |
| Mode | Select the type of homing to use. | |
| | To | Select |
| | <ul style="list-style-type: none"> Use a homing sequence Use the trapezoidal velocity profile. | Active |
| | Redefine the current absolute position on the next occurrence of the encoder marker | Passive |
| Sequence | Select the type of active homing sequence to use. | |
| | To | Select |
| | Allow the controller to immediately assign the home position to the current axis position | Immediate home |
| | Move the axis until it detects the home limit switch | Home to switch |
| | <ul style="list-style-type: none"> Move the axis until it detects the home limit switch Use the most precise active homing sequence. | Home to switch with marker |
| | Move the axis until it detects the encoder marker | Home to marker only |
| Limit switch | Select the default setting of the home switch. | |
| | To set the home switch | Select |
| | To open | Normally open |
| | To closed | Normally closed |
| Homing direction | Select the initial direction of the homing motion. | |
| | To use | Select |
| | A positive direction | Positive |
| | A negative direction | Negative |
| Homing speed | Type the initial speed of the jog profile used in an active homing sequence. | |
| Home return speed | Type the return speed of the jog profile used in an active homing sequence. | |

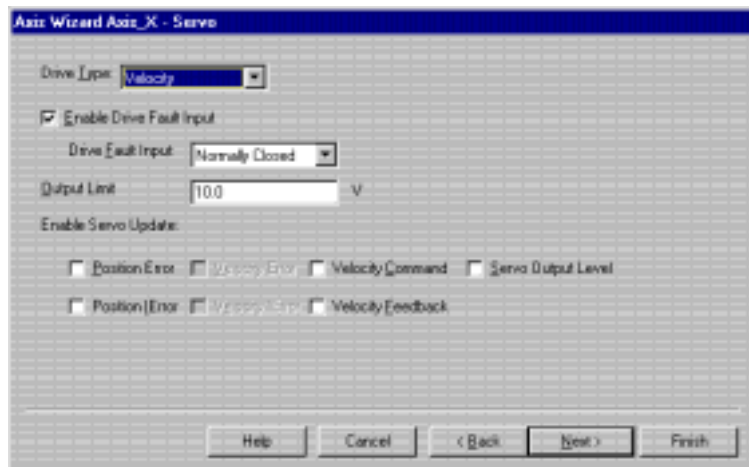
23. Select **Next**. The Axis Wizard-Overtravels window appears.



24. Make entries in the following fields.

| Field | Entry |
|--------------------|---|
| Soft travel limits | If you want to use soft overtravel limits, select Soft Travel Limits . |
| Maximum positive | Type the maximum overtravel value in the positive direction. |
| Maximum negative | Type the maximum overtravel value in the negative direction. |

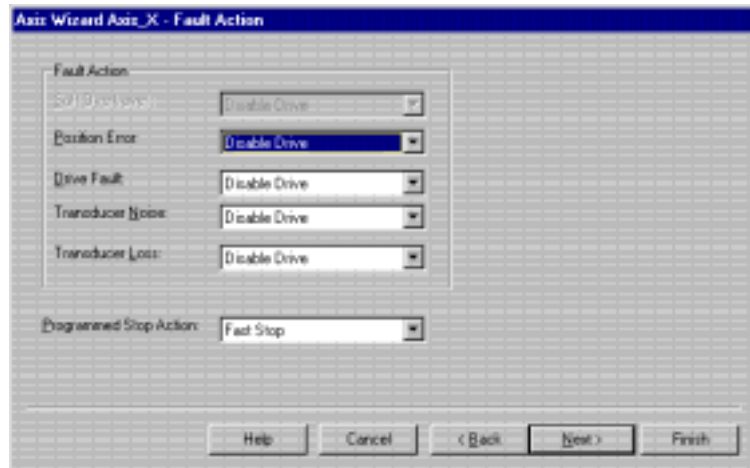
25. Select **Next**. The Axis Wizard-Servo window appears.



26. Make entries in the following fields.

| Field | Entry | |
|--------------------------|---|-----------------|
| Drive type | Select the drive type you are using. | |
| | To use | Select |
| | A velocity loop servo drive | Velocity |
| | A torque loop servo drive | Torque |
| Enable drive fault input | If you are using the servo module drive fault input, select Enable Drive Fault Input . | |
| Drive fault input | Select the type of drive fault input. | |
| | If the drive fault input | Select |
| | Closed in reference to the servo module | Normally closed |
| | Open in reference to the servo module | Normally open |
| Output limit | Type the maximum servo output voltage of your axis. | |
| Enable servo update | Select the status attributes you want to regularly update. | |

27. Select **Next**. The Axis Wizard-Fault Action window appears.



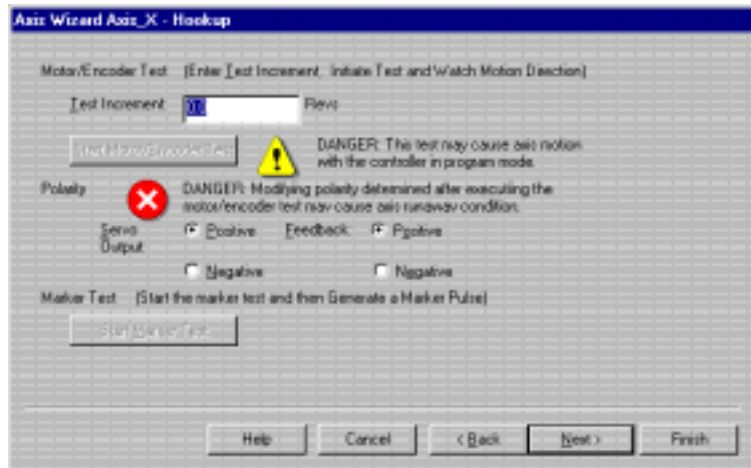
28. Make entries in the following fields.

| Field | Entry | |
|-----------------|--|---------------|
| Soft overtravel | Select the type of action when this fault occurs. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output • Open the OK contact | Shutdown |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output | Disable drive |
| | Decelerate the axis to a stop according to the MaximumDeceleration value | Stop motion |
| | Handle the fault using your application program | Status only |
| Position error | Select the type of action when this fault occurs. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output • Open the OK contact | Shutdown |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output | Disable drive |
| | Decelerate the axis to a stop according to the MaximumDeceleration value | Stop motion |
| | Handle the fault using your application program | Status only |

| Field | Entry | |
|---|--|---------------|
| Drive fault | Select the type of action when this fault occurs. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output • Open the OK contact | Shutdown |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output | Disable drive |
| | Decelerate the axis to a stop according to the MaximumDeceleration value | Stop motion |
| Transducer noise | Select the type of action when this fault occurs. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output • Open the OK contact | Shutdown |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output | Disable drive |
| | Decelerate the axis to a stop according to the MaximumDeceleration value | Stop motion |
| Handle the fault using your application program | Status only | |

| Field | Entry | |
|--|--|---------------|
| Transducer loss | Select the type of action when this fault occurs. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output • Open the OK contact | Shutdown |
| | <ul style="list-style-type: none"> • Disable servo action • Zero the servo amplifier output • Deactivate the drive enable output | Disable drive |
| | Decelerate the axis to a stop according to the MaximumDeceleration value | Stop motion |
| Handle the fault using your application program | Status only | |
| Programmed stop action | Select how this axis will stop when | |
| | <ul style="list-style-type: none"> • The ControlLogix controller undergoes a critical mode change • You use a Motion Group Programmed Stop (MGPS) instruction. | |
| | To | Select |
| | <ul style="list-style-type: none"> • Decelerate the axis to a stop using the MaximumDeceleration value. • Maintain servo action after axis motion stops. | Fast stop |
| <ul style="list-style-type: none"> • Decelerate the axis to a stop using the MaximumDeceleration value. • Place the axis in the shutdown state after axis motion stops. <p>Note: You must use shutdown reset instructions (MASR or MGSR) to recover the axis from the shutdown state.</p> | Fast shutdown | |
| <ul style="list-style-type: none"> • Immediately place the axis in the shutdown state. <p>Note: Unless you configure the axis with dynamic breaking, the axis will coast to a stop.</p> <p>Note: You must use shutdown reset instructions (MASR or MGSR) to recover the axis from the shutdown state.</p> | Hard shutdown | |

29. Select **Next**. The Axis Wizard-Hookup window appears.

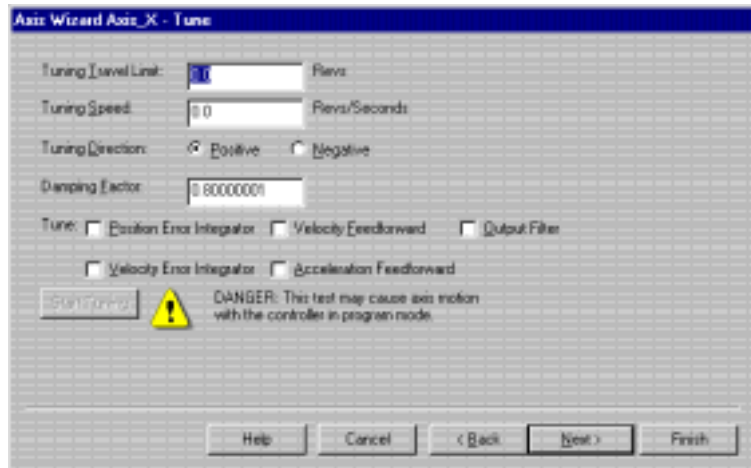


Note: To use the hookup diagnostic tests, you must ensure the controller is online and the application program is downloaded. If the controller is offline, the **Start Motor/Encoder Test** and **Start Marker Test** buttons will be greyed-out.

30. Make entries in the following fields.

| Field | Entry | | | | | | |
|-----------------------|--|--------|--------|-----------------------|----------|-----------------------|----------|
| Test increment | Type the amount of motion that is necessary to test: <ul style="list-style-type: none"> The connection to the encoder. The direction of the encoder. | | | | | | |
| Servo output | Select the polarity of the servo output to the drive. | | | | | | |
| | <table border="1"> <thead> <tr> <th>To</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Use positive polarity</td> <td>Positive</td> </tr> <tr> <td>Use negative polarity</td> <td>Negative</td> </tr> </tbody> </table> | To | Select | Use positive polarity | Positive | Use negative polarity | Negative |
| | To | Select | | | | | |
| Use positive polarity | Positive | | | | | | |
| Use negative polarity | Negative | | | | | | |
| Use negative polarity | Negative | | | | | | |
| Feedback | Select the polarity of the encoder feedback. | | | | | | |
| | <table border="1"> <thead> <tr> <th>To</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Use positive polarity</td> <td>Positive</td> </tr> <tr> <td>Use negative polarity</td> <td>Negative</td> </tr> </tbody> </table> | To | Select | Use positive polarity | Positive | Use negative polarity | Negative |
| | To | Select | | | | | |
| Use positive polarity | Positive | | | | | | |
| Use negative polarity | Negative | | | | | | |
| Use negative polarity | Negative | | | | | | |

31. Select **Next**. The Axis Wizard-Tune window appears.

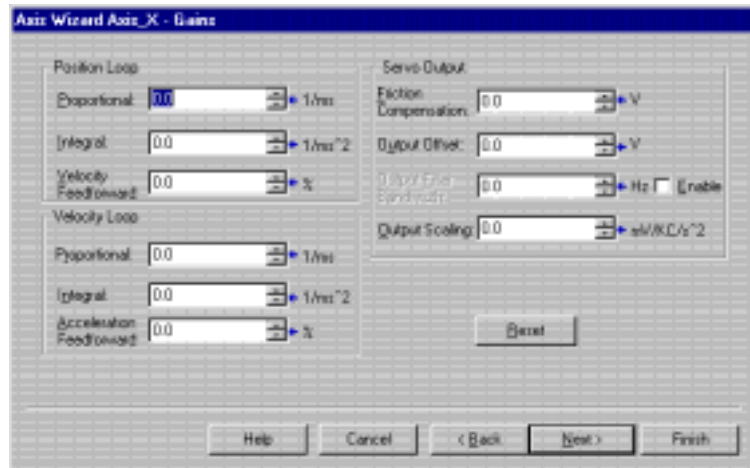


Note: To use auto tuning, you must ensure the controller is online and the application program is downloaded. If the controller is offline, the **Start Tuning** button will be greyed-out.

32. Make entries in the following fields.

| Field | Entry | | | | | | |
|----------------------------|--|--------|--------|----------------------------|----------|----------------------------|----------|
| Tuning travel limit | Type the limit of axis motion during the auto tuning. | | | | | | |
| Tuning speed | Type the maximum speed initiated during auto tuning. | | | | | | |
| Tuning direction | Select the direction of the tuning motion profile. | | | | | | |
| | <table border="1"> <thead> <tr> <th>To</th> <th>Select</th> </tr> </thead> <tbody> <tr> <td>Use the positive direction</td> <td>Positive</td> </tr> <tr> <td>Use the negative direction</td> <td>Negative</td> </tr> </tbody> </table> | To | Select | Use the positive direction | Positive | Use the negative direction | Negative |
| | To | Select | | | | | |
| Use the positive direction | Positive | | | | | | |
| Use the negative direction | Negative | | | | | | |
| Damping factor | Type the value to calculate the maximum position servo bandwidth. | | | | | | |
| Tune | Select the values you want to calculate during tuning. | | | | | | |

33. Select **Next**. The Axis Wizard-Gains window appears.

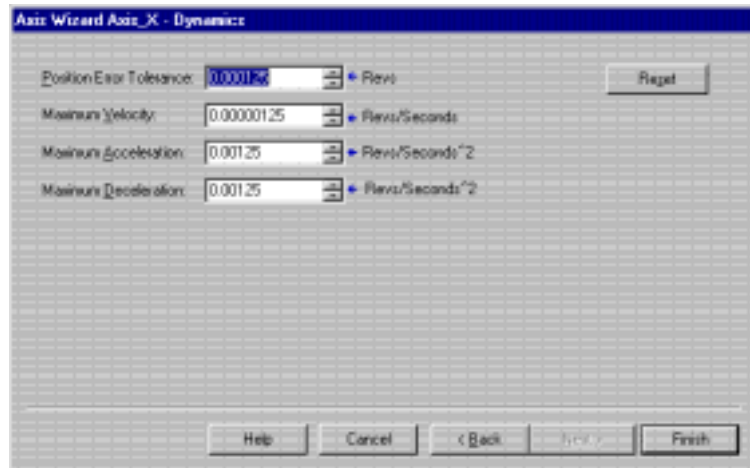


34. Make entries in the following fields.

| Field | Entry |
|--|--|
| Position loop - Proportional | Type the value of the position proportional gain. This value is multiplied by the position error to produce a portion of the velocity command. |
| Position loop - Integral | Type the value of the position integral gain correction. You can use this value to improve the steady-state positioning performance of the system. |
| Position loop - Velocity feedforward | Type the velocity feedforward gain value. You can use this value to provide the velocity command output necessary to generate the commanded velocity. |
| Velocity loop - Proportional | Type the value of the velocity proportional gain. This value is multiplied by the velocity error to produce a portion of the servo output or torque command. |
| Velocity loop - Integral | Type the value of the velocity integral gain correction. This value is multiplied by the velocity integral error to produce a portion of the servo output or torque command. |
| Velocity loop - Acceleration feedforward | Type the acceleration feedforward gain value. You can use this value to provide the torque command output necessary to generate the commanded acceleration. |
| Friction compensation | Type the output level necessary to overcome the static friction of your axis. |
| Output offset | Type a value to offset the cumulative offsets of the servo module DAC output and the servo drive input. |
| Output filter bandwidth | Type the value of the bandwidth of the servo low-pass digital output filter. |
| Output scaling | Type the value to convert the output of the servo loop into the equivalent drive voltage. |
| Reset | To reset the values to those determined during auto tuning, select Reset . |

Note: You can also determine these entries by performing auto tuning. For more information, see the *Running Hookup Diagnostics and Auto Tuning* section.

35. Select **Next**. The Axis Wizard-Dynamics window appears.

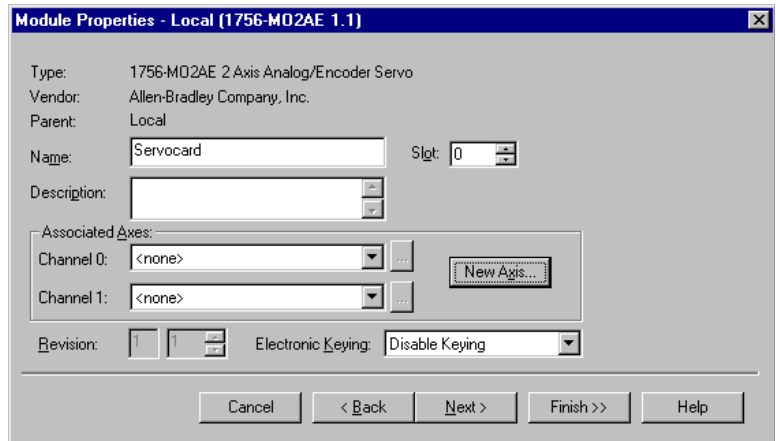


36. Make entries in the following fields.

| Field | Entry |
|--------------------------|--|
| Position error tolerance | Type the value of the position error the servo module can tolerate before a position error fault occurs. |
| Maximum velocity | Type the value of the maximum steady-state speed of the axis. |
| Maximum acceleration | Type the maximum acceleration to apply to an axis. |
| Maximum deceleration | Type the maximum deceleration to apply to an axis. |
| Reset | To reset the values to those determined during auto tuning, select Reset . |

Note: You can also determine these entries by performing auto tuning. For more information, see the *Running Hookup Diagnostics and Auto Tuning* section.

37. Select **Finish**. The Module Properties window appears.



38.

| If | Then |
|---|--|
| You want to assign your axis to channel 0 | In the <i>Channel 0</i> field, select your axis from the drop-down menu. |
| You want to assign your axis to channel 1 | In the <i>Channel 1</i> field, select your axis from the drop-down menu. |

39.

| If | Then |
|-------------------------------------|--|
| You want to add another axis | Go to the <i>Naming an Axis</i> section. |
| You do not want to add another axis | Select Finish . |

Assigning Additional Motion Modules and Axes

You can assign additional modules and axes by repeating the preceding sections. To name and assign another axis, refer to the *Naming an Axis* section.

You can assign up to 16 1756-M02AE modules to each Logix5550 controller. Each module uses a maximum of two axes. To add an additional motion module, refer to the *Adding the 1756-M02AE Module* section.

Running Hookup Diagnostics and Auto Tuning

Once you have added and configured your motion module and axes, you can run hookup diagnostics and auto tuning. To run diagnostics and tuning, you must download a program and go online.

To run diagnostics and tuning:

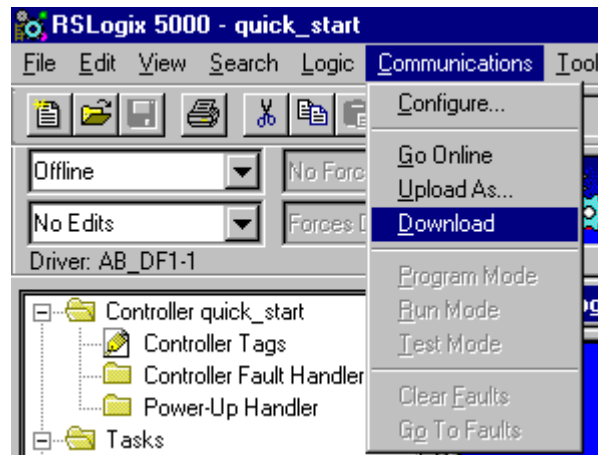
1. Double-click **Main Routine**.

- 2.

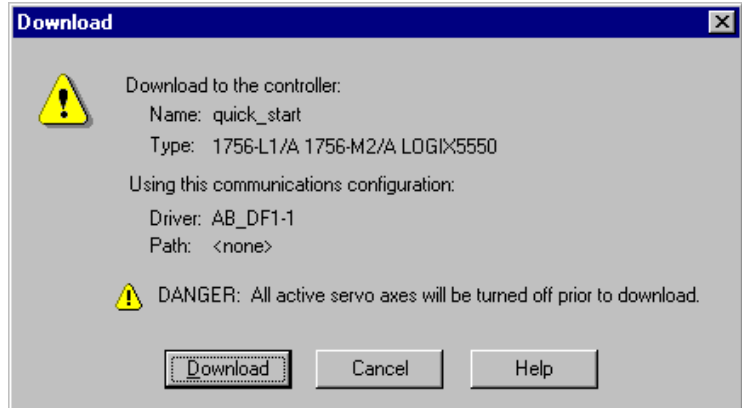
| If | Then |
|--|--|
| The Main Routine window only shows rung 0 and the end rung | <ul style="list-style-type: none"> • Select rung 0 • Delete rung 0 • Go to step 3 |
| The Main routine window shows ladder logic rungs | Go to step 3 |

3. Make sure the keyswitch is in the REM position.

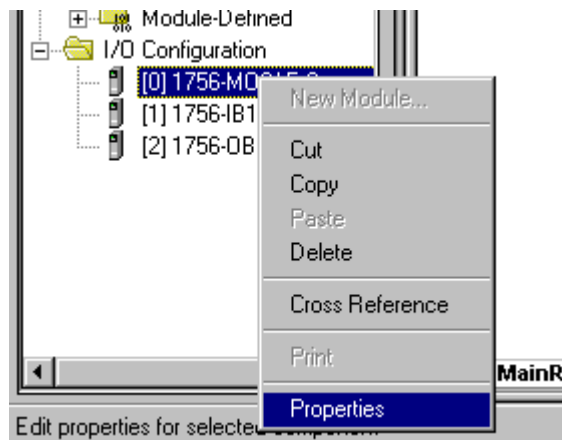
4. From the Communications menu, select **Download**.



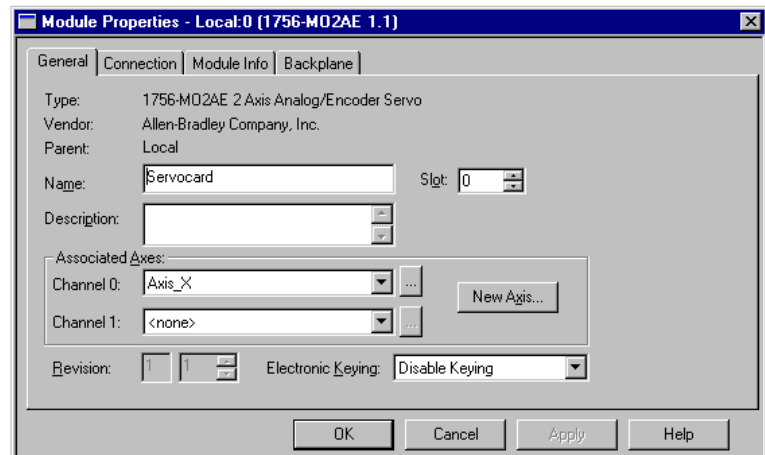
A window similar to the following appears.




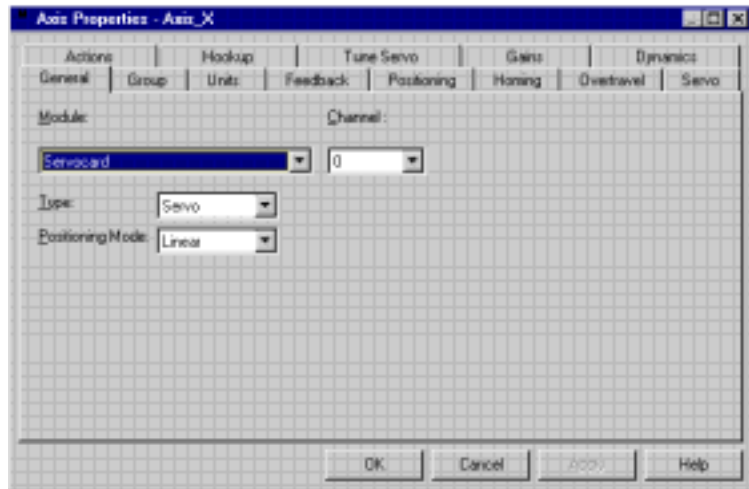
5. Select **Download**.
6. Under the I/O Configuration folder, right-click the 1756-M02AE module you want to use.



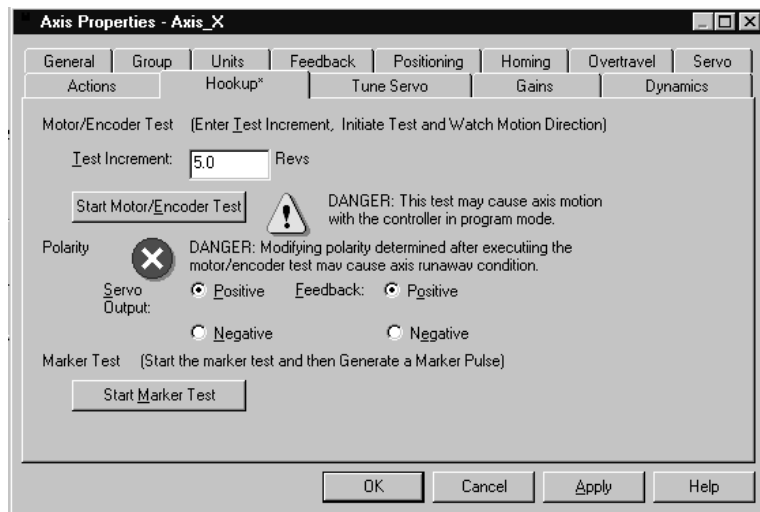
7. Select **Properties**. The Module Properties window appears.



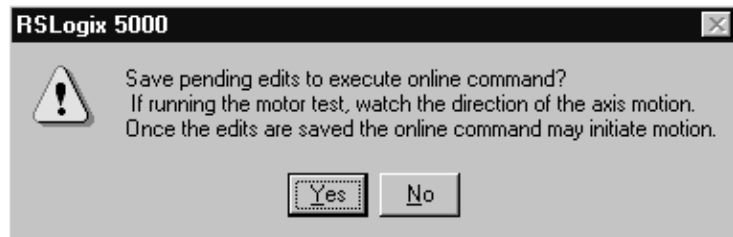
8. Next to the *Channel* field of your axis, select the  button. The Axis Properties window appears.



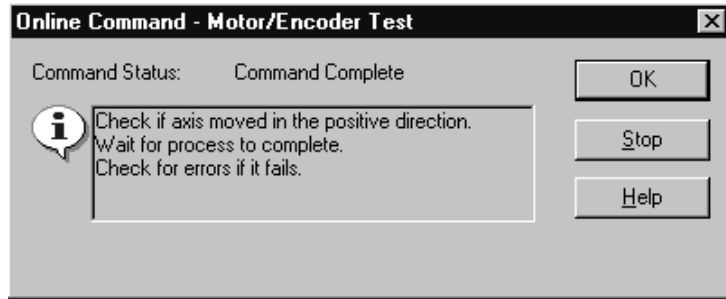
9. Select the Hookup tab. The following window appears.



10. Select **Start Motor/Encoder Test**. The following window appears.

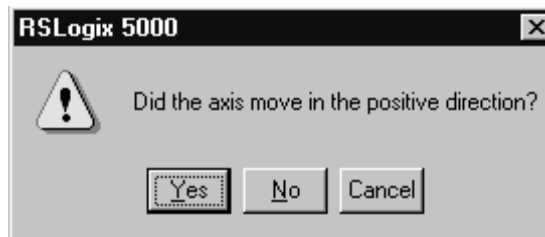


11. Select **Yes**. The following window appears.



12. Watch the motor to see which way it turns.

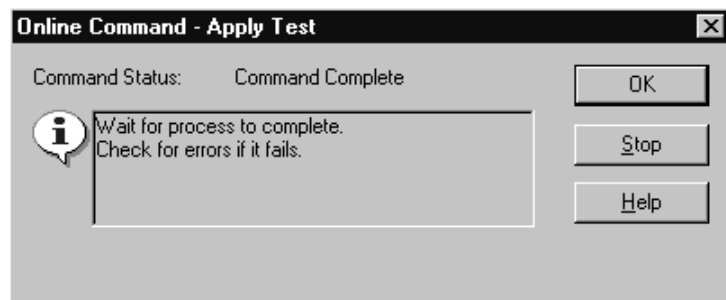
13. Select **OK**. The following window appears.



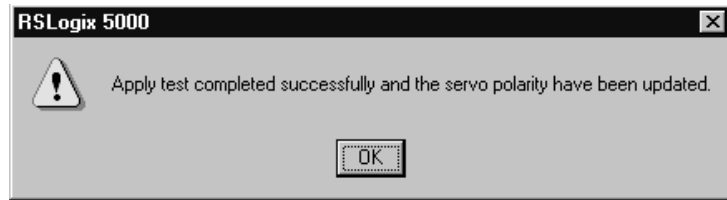
14.

| If the axis | Then |
|---------------------------------|-------------------|
| Moved in the positive direction | Select Yes |
| Moved in the negative direction | Select No |

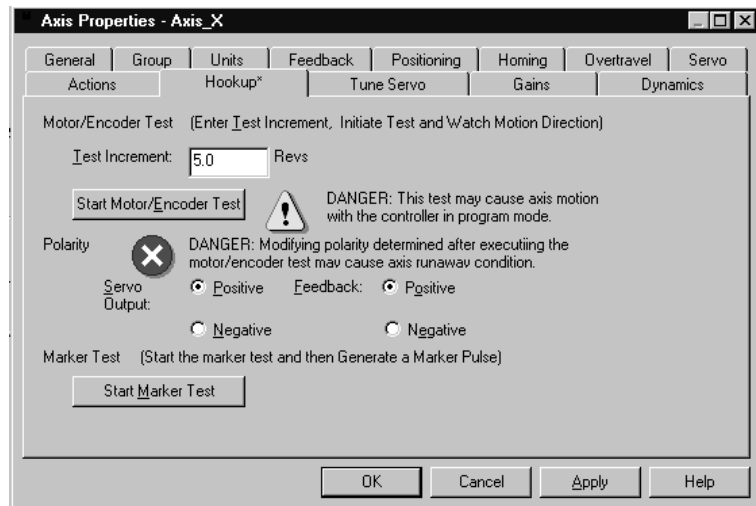
The following window appears.



15. Select **OK**. The following window appears.



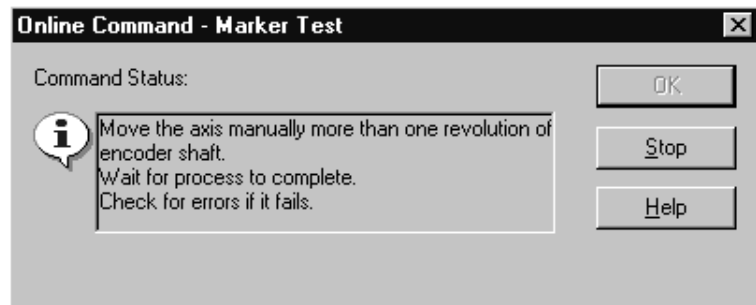
16. Select **OK**. The Axis Properties window appears.



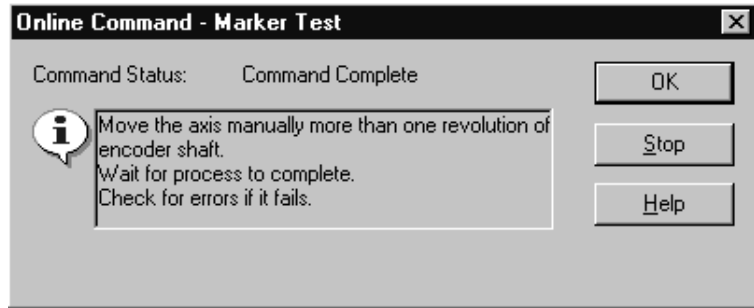
17. Select **Start Marker Test**. The following window appears.



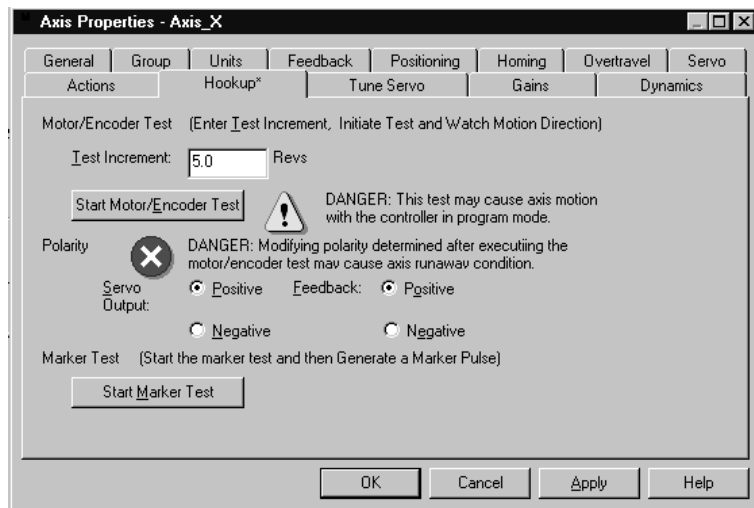
18. Select **Yes**. The following window appears.



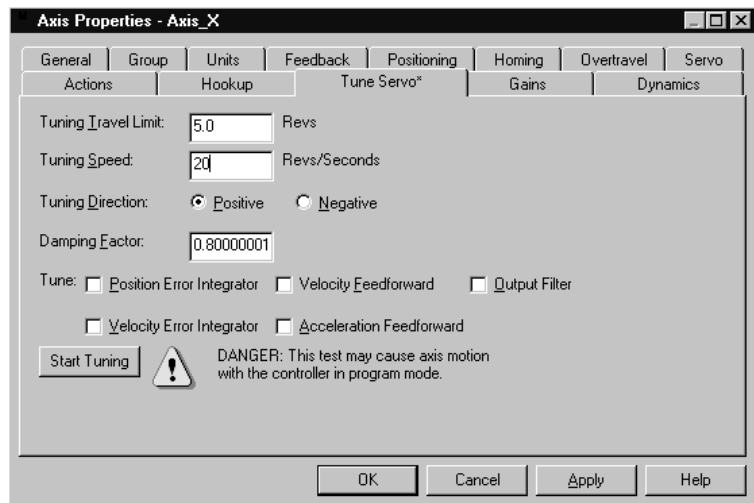
19. Slowly rotate the motor axis until the following window appears.



20. Select **OK**. The Axis Properties window appears.



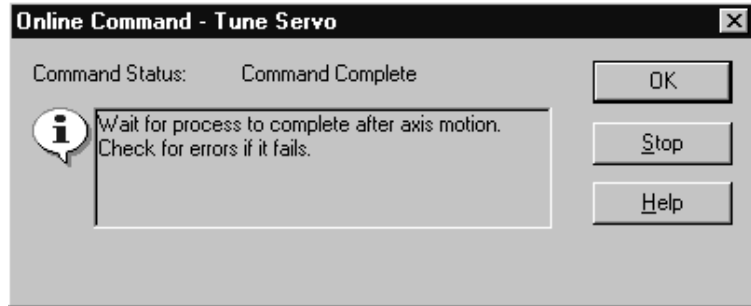
21. Select the Tune Servo tab. The following window appears.



22. Select **Start Tuning**. The following window appears.



23. Select **Yes**. The following window appears.



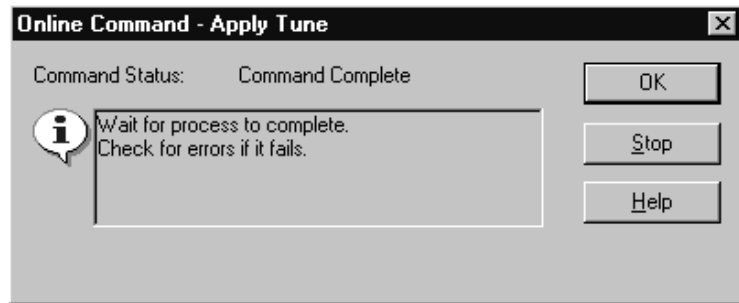
24. Select **OK**. The Tune Bandwidth window appears.



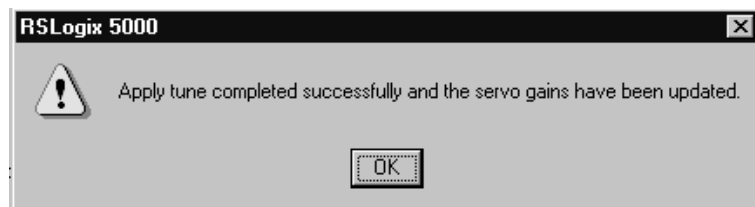
25.

| If: | Then: |
|---|---|
| You do not want to change the bandwidth | Go to step 26. |
| You want to change the bandwidth | <ol style="list-style-type: none"> 1. In the <i>Bandwidth</i> field, type the position servo bandwidth, which is the unity bandwidth used to calculate gains. 2. Go to step 26. |

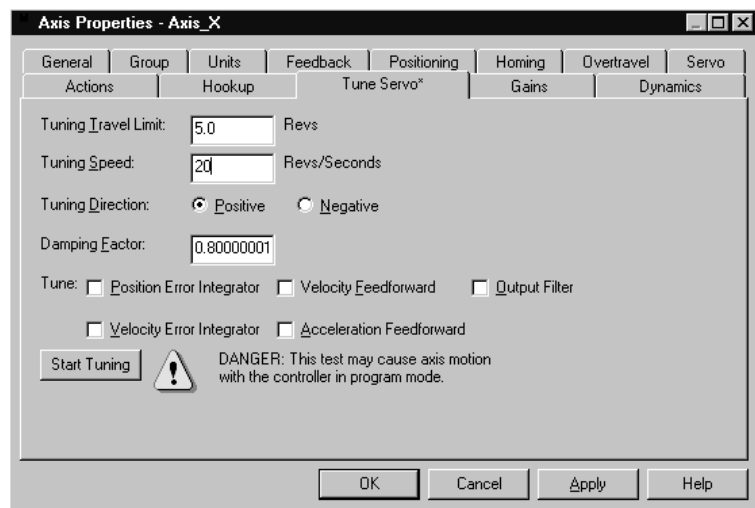
26. Select **OK**. The following window appears.



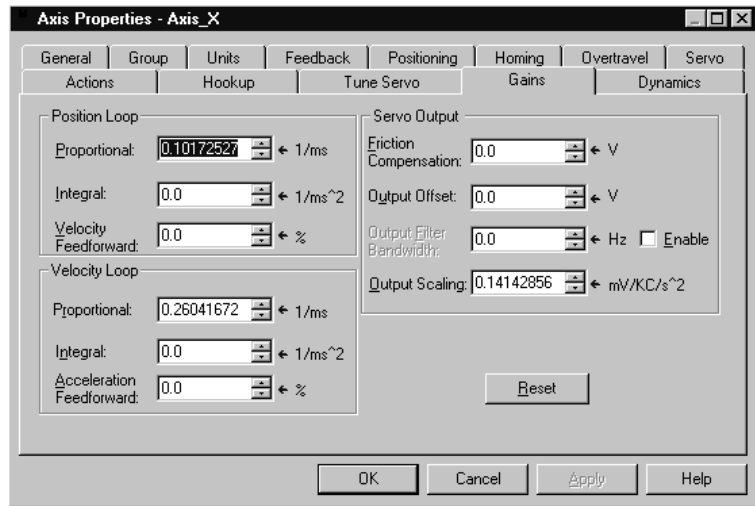
27. Select **OK**. The following window appears.



28. Select **OK**. The Axis Properties window appears.



29. Select the Gains tab. The following window appears.

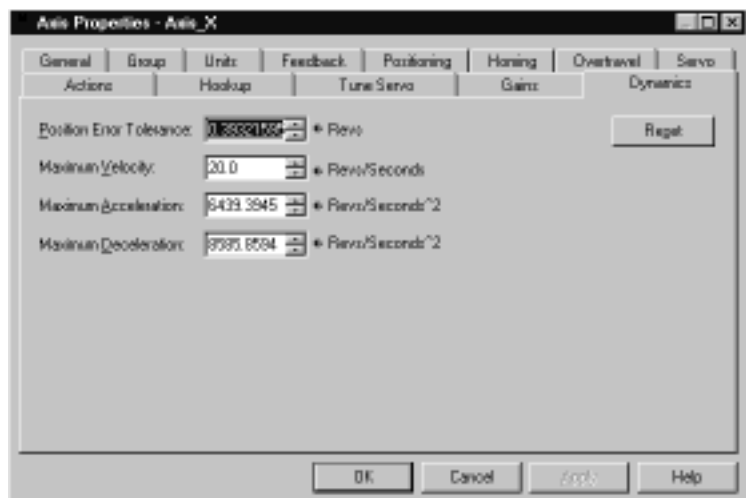


The window will show new values for the position loop, velocity loop, and output compensation.

30.

| If | Then |
|--|--|
| You want to change the position loop, velocity, loop, and servo output values | <ol style="list-style-type: none"> 1. Type the new values in the appropriate fields. 2. Go to step 31. |
| You do not want to change the position loop, velocity, and servo output values | Go to step 31. |

31. Select the Dynamics tab. The following window appears.



This window will show new values for maximum velocity, error tolerance, maximum acceleration, and maximum deceleration.

32.

| If | Then |
|---|--|
| You want to change the dynamics values | 1. Type the new values in the appropriate fields. 2. Go to step 33. |
| You do not want to change the dynamics values | Go to step 33. |

33. Select **OK**. The Axis Properties window will close.

Developing a Motion Application Program

To write a motion application program, you can insert motion instructions directly into the ladder diagram application program. The motion instruction set consists of five groups of motion instructions:

- Motion state instructions
- Motion move instructions
- Motion group instructions
- Motion event instructions
- Motion configuration instructions

For more information about these instructions, refer to Chapter 5 - *Understanding Motion Instructions*.

Understanding a Programming Example

The following figure shows several rungs of a motion control application program.

Rung 0:

Enables the Feed and Cut axes when you press the servo_on button.

Rung 1:

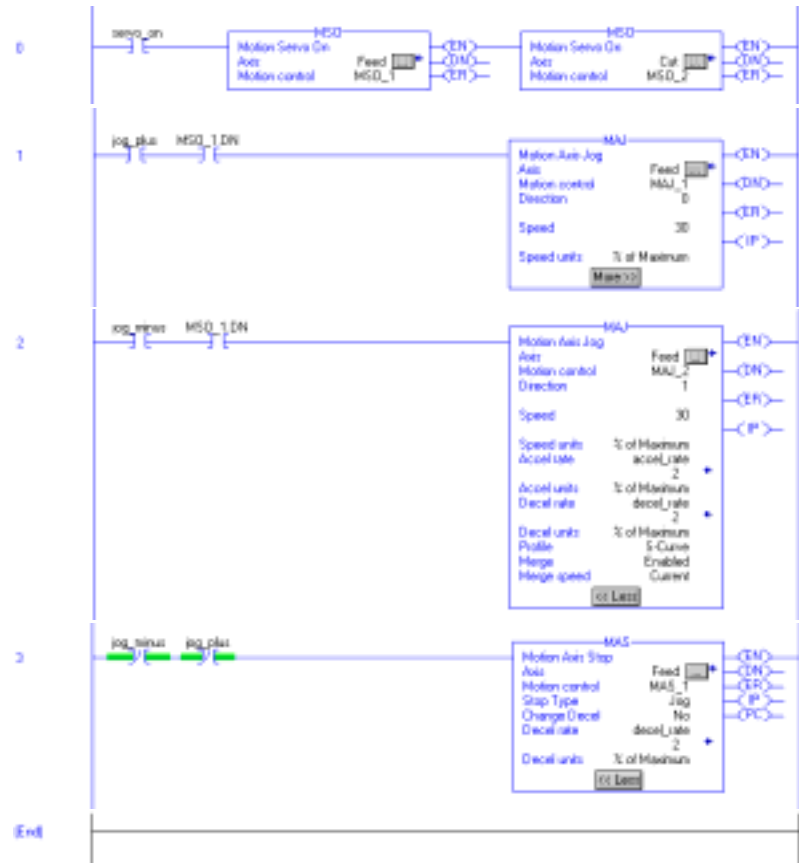
Jogs the Feed axis in the positive direction when you press the jog_plus button.

Rung 2:

Jogs the Feed axis in the reverse direction when you press the jog_minus button.

Rung 3:

Stops the Feed axis when you release with the jog_plus button or the jog_minus button.



For more information about instructions and creating application programs, refer to the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

Understanding Motion Instructions

This chapter describes the 27 motion instructions for RSLogix 5000 programming software. The following table shows the contents of this chapter:

| For information about | See page |
|---|----------|
| Understanding Motion State Instructions | 5-2 |
| Understanding Motion Move Instructions | 5-3 |
| Understanding Motion Group Instructions | 5-4 |
| Understanding Motion Event Instructions | 5-5 |
| Understanding Motion Configuration Instructions | 5-6 |

The motion instructions for the RSLogix 5000 programming software consist of five main categories:

- Motion state instructions
- Motion move instructions
- Motion group instructions
- Motion event instructions
- Motion configuration instructions

| For more information about | Refer to |
|------------------------------------|---|
| Motion instructions | The Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1 |
| Types of motion instruction timing | Appendix E - <i>Instruction Timing</i> |

Understanding Motion State Instructions

Motion state instructions directly control or change the operating state of an axis.

The motion state instructions are:

| Instruction | Abbreviation | Description | Type of Timing | Typical Execution Time |
|----------------------------|--------------|--|----------------|------------------------|
| Motion Servo On | MSO | Enables the servo drive and activates the axis servo loop | Message | 195 μ s |
| Motion Servo Off | MSF | Disables the servo drive and deactivates the axis servo loop | Message | 185 μ s |
| Motion Axis Shutdown | MASD | Forces an axis into the shutdown operating state Note: Once the axis is in the shutdown state, the controller will block any instructions that initiate axis motion. | Message | 165 μ s |
| Motion Axis Shutdown Reset | MASR | Changes an axis from an existing shutdown operating state to an axis ready operating state Note: If all of the axes of a servo module are removed from the shutdown state as a result of this instruction, the OK relay contacts for the module will close. | Message | 165 μ s |
| Motion Direct Drive On | MDO | Enables the servo drive and sets the servo output voltage of an axis | Message | 270 μ s |
| Motion Direct Drive Off | MDF | Disables the servo drive and sets the servo output voltage to the output offset voltage | Message | 165 μ s |
| Motion Axis Fault Reset | MAFR | Clears all motion faults | Message | 165 μ s |

For more information about motion state instructions, refer to the *Motion State Instructions* chapter of the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

For more information about instruction timing, refer to Appendix E - *Instruction Timing*.

Understanding Motion Move Instructions

Motion move instructions control axis motion.

The motion move instructions are:

| Instruction | Abbreviation | Description | Type of Timing | Typical Execution Time |
|--------------------------|--------------|---|-------------------|------------------------|
| Motion Axis Stop | MAS | Initiates a controlled stop of any motion process on an axis | Immediate Process | 230 μ s |
| Motion Axis Home | MAH | Homes an axis | Message Process | 60 μ s |
| Motion Axis Jog | MAJ | Initiates a jog motion profile for an axis | Immediate Process | 570 μ s |
| Motion Axis Move | MAM | Initiates a move profile for an axis | Immediate Process | 684 μ s |
| Motion Axis Gear | MAG | Enables electronic gearing between two axes | Immediate Process | 250 μ s |
| Motion Change Dynamics | MCD | Changes the speed, acceleration rate, or deceleration rate of a move profile or jog profile in progress | Immediate | 545 μ s |
| Motion Redefine Position | MRP | Changes the command or actual position of an axis | Message | 349 μ s |

For more information about motion state instructions, refer to the *Motion Move Instructions* chapter of Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

For more information about instruction timing, refer to Appendix E - *Instruction Timing*.

Understanding Motion Group Instructions

Motion group instructions initiate action on all axes in a group.

The motion group instructions are:

| Instruction | Abbreviation | Description | Type of Timing | Typical Execution Time |
|------------------------------|--------------|--|--------------------|------------------------|
| Motion Group Stop | MGS | Initiates a stop of motion on a group of axes | Process | 90 μ s |
| Motion Group Programmed Stop | MGPS | Initiates a stop of all motion on all the axes in a group using the method that you set for each axis. | Message Process | 60 μ s |
| Motion Group Shutdown | MGSD | Forces all the axes in a group into the shutdown operating state | Message | 60 μ s |
| Motion Group Shutdown Reset | MGSR | Transitions a group of axes from the shutdown operating state to the axis ready operating state | Message | 60 μ s |
| Motion Group Strobe Position | MGSP | Latches the current command and actual positions of all the axes in a group | Immediate | 45 μ s |

For more information about motion state instructions, refer to the *Motion Group Instructions* chapter of Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

For more information about instruction timing, refer to Appendix E - *Instruction Timing*.

Understanding Motion Event Instructions

Motion event instructions control the arming and disarming of special event checking functions, such as registration and watch position.

The motion event instructions are:

| Instruction | Abbreviation | Description | Type of Timing | Typical Execution Time |
|------------------------------|--------------|--|-----------------|------------------------|
| Motion Arm Watch Position | MAW | Arms watch-position event checking for an axis | Message Process | 340 μ s |
| Motion Disarm Watch Position | MDW | Disarms watch-position event checking for an axis | Message | 165 μ s |
| Motion Arm Registration | MAR | Arms servo module registration event checking for an axis | Message Process | 480 μ s |
| Motion Disarm Registration | MDR | Disarms servo module registration event checking for an axis | Message | 165 μ s |

For more information about motion state instructions, refer to the *Motion Event Instructions* chapter of Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

For more information about instruction timing, refer to Appendix E - *Instruction Timing*.

Understanding Motion Configuration Instructions

Motion configuration instructions allow you to tune an axis and to run diagnostic tests for your control system. These tests include:

- A motor/encoder hookup test
- An encoder hookup test
- A marker test

The motion configuration instructions are:

| Instruction | Abbreviation | Description | Type of Timing | Typical Execution Time |
|--------------------------------|--------------|--|--------------------|----------------------------------|
| Motion Apply Axis Tuning | MAAT | Computes a complete set of servo gains and dynamic limits based on a previously executed MRAT instruction Note: The MAAT instruction also updates the servo module with the new gain parameters. | Message | 870 μ s |
| Motion Run Axis Tuning | MRAT | Commands the servo module to run a tuning motion profile for an axis | Message Process | less than the coarse update rate |
| Motion Apply Hookup Diagnostic | MAHD | Applies the results of a previously executed MRHD instruction Note: The MAHD instruction generates a new set of encoder and servo polarities based on the observed direction of motion during the MRHD instruction. | Message | 170 μ s |
| Motion Run Hookup Diagnostic | MRHD | Commands the servo module to run one of three diagnostic tests on an axis | Message Process | less than the coarse update rate |

For more information about motion state instructions, refer to the *Motion Configuration Instructions* chapter of Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

For more information about instruction timing, refer to Appendix E - *Instruction Timing*.

Troubleshooting

This chapter describes how to troubleshoot your ControlLogix motion control system. The following table shows the contents of this chapter:

| For information about | See page |
|---|----------|
| Understanding Module Status Using the OK Indicator | 6-1 |
| Understanding Module Status Using the FDBK Indicator | 6-2 |
| Understanding Module Status Using the DRIVE Indicator | 6-3 |

Understanding Module Status Using the OK Indicator

| If the OK LED displays | Then the module status is | Take this action |
|------------------------|--|--|
| Off | The module is not operating. | <ul style="list-style-type: none"> Apply chassis power. Verify the module is completely inserted into the chassis and backplane. |
| Flashing green light | The module has passed internal diagnostics, but it is not communicating axis data over the backplane. | <ul style="list-style-type: none"> None, if you have not configured the module. If you have configured the module, check the slot number in the 1756-M02AE Properties dialog box. |
| Steady green light | <ul style="list-style-type: none"> Axis data is being exchanged with the module. The module is in the normal operating state. | None. The module is ready for action. |
| Flashing red light | <ul style="list-style-type: none"> A major recoverable failure has occurred. A communication fault, timer fault, or NVS update is in progress. | <ul style="list-style-type: none"> Check the servo fault word for the source of the error. Clear the fault condition using the motion instructions. Resume normal operation. If the flashing persists, reconfigure the module. |
| Solid red light | A potential non-recoverable fault has occurred. | <ul style="list-style-type: none"> Reboot the module. If the solid red persists, replace the module. |

Understanding Module Status Using the FDBK Indicator

| If the FDBK LED displays | Then the module status is | Take this action |
|--------------------------|--|---|
| Off | The axis is not used. | <ul style="list-style-type: none"> • None, if you are not using this axis. • If you are using this axis, make sure you configured the module and associated an axis tag with the module. |
| Flashing green light | The axis is in the normal servo loop inactive state. | None. You can change the servo axis state by executing motion instructions. |
| Steady green light | The axis is in the normal servo loop active state. | None. You can change the servo axis state by executing motion instructions. |
| Flashing red light | The axis servo loop error tolerance has been exceeded. | <ul style="list-style-type: none"> • Correct the source of the problem. • Clear the servo fault using a fault reset instruction. • Resume normal operation. |
| Solid red light | An axis encoder feedback fault has occurred. | <ul style="list-style-type: none"> • Correct the source of the problem by checking the encoder and power connections. • Clear the servo fault using the MAFR instruction. • Resume normal operation. |

Understanding Module Status Using the DRIVE Indicator

| If the DRIVE LED displays | Then the module status is | Take this action |
|---------------------------|---|---|
| Off | <ul style="list-style-type: none"> • The axis is not used. • The axis is a position-only axis type. | <ul style="list-style-type: none"> • None, if you are not using the axis or have configured it as a position-only axis. • Otherwise, make sure you have configured the module, associated an axis tag with the module, and configured the axis as a servo axis. |
| Flashing green light | The axis drive is in the normal disabled state. | None. You can change the servo axis state by executing a motion instruction. |
| Steady green light | The axis drive is in the normal enabled state. | None. You can change the servo axis state by executing a motion instruction. |
| Flashing red light | The axis drive output is in the Shutdown state. | <ul style="list-style-type: none"> • Check for faults that may have generated this state. • Execute the shutdown reset motion instruction. • Resume normal operation. |
| Solid red light | The axis drive is faulted. | <ul style="list-style-type: none"> • Check the drive status. • Clear the drive fault condition at the drive. • Execute a fault reset motion instruction. • Resume normal operation. |

Specifications and Performance


This appendix shows specifications and performance guidelines for the motion module. The following table shows the contents of this appendix:

| For information about | See page |
|---|----------|
| Understanding Motion Module Specifications | A-1 |
| Understanding Coarse Update Rate Calculations | A-4 |

Understanding Motion Module Specifications

| | | |
|---|---|---------------------|
| Number of axes per chassis | Configurable | |
| Motion commands | 27 | |
| Number of axes per module | 2 axes maximum | |
| Maximum number of axes per coarse update rate | Coarse Update Rate: | Max number of axes: |
| Note: The coarse update rates assume that the servo is on for each axis and that each axis has an active trapezoidal move. For more information, refer to the <i>Understanding Coarse Update Rate Calculations</i> section. | 2 ms | 2 |
| | 3 ms | 3 |
| | 4 ms | 4 |
| | 5 ms | 6 |
| | 6 ms | 7 |
| | 7 ms | 8 |
| | 8 ms | 10 |
| | 9 ms | 11 |
| | 10 ms | 13 |
| | 11 ms | 14 |
| | 12 ms | 15 |
| | 13 ms | 17 |
| | 14 ms | 18 |
| | 15 ms | 20 |
| 16 ms | 21 | |
| 17 ms | 22 | |
| 18 ms | 24 | |
| 19 ms | 25 | |
| 20 ms | 26 | |
| 21 ms | 28 | |
| 22 ms | 29 | |
| 23 ms | 30 | |
| 24 ms | 32 | |
| Servo loop | | |
| Type | Nested PI digital position and velocity servo | |
| Gain resolution | 32-bit floating point | |
| Absolute position range | ±1,000,000,000 encoder counts | |
| Rate | 5 kHz | |
| Module location | 1756 ControlLogix chassis | |

| | |
|-------------------------------|--|
| Module keying | Electronic |
| Power dissipation | 5.5W maximum |
| Backplane current | 5V dc @ 700 mA 24V dc @ 2.5 mA |
| Encoder input | |
| Type | Incremental AB quadrature with marker |
| Mode | 4X quadrature |
| Rate | 4 MHz counts per second maximum |
| Electrical interface | Optically isolated 5V differential |
| Voltage range | 3.4V to 5.0V differential |
| Input impedance | 531 Ohms differential |
| Registration inputs | |
| Type | Optically isolated, current sinking input |
| 24V input voltage | +24V dc nominal |
| Maximum | 26.4V |
| Minimum | 18.5V |
| Maximum off | 6.1V |
| 5V input voltage | +5V dc nominal |
| Maximum | 5.5V |
| Minimum | 3.7V |
| Maximum off | 2.0V |
| Input impedance | |
| 24V input | 1.2 kOhms |
| 5V input | 9.5 kOhms |
| Response time | 1 μ s |
| All other inputs | |
| Type | Optically isolated, current sinking input |
| Input voltage | +24V dc nominal |
| Maximum | 26.4V |
| Maximum on | 17.0V |
| Maximum off | 8.5V |
| Input impedance | 7.5 kOhms |
| Servo output | |
| Type | Analog voltage |
| Isolation | 200 kOhms |
| Voltage range | \pm 10V |
| Voltage resolution | 16 bits |
| Load | 5.6 kOhms resistive minimum |
| All other outputs | |
| Type | Solid-state isolated relay contacts |
| Operating voltage | +24V dc nominal |
| Maximum | 26.4V |
| Operating current | 75 mA |
| RTB keying | User-defined |
| Field wiring arm | 36-position RTB (1756-TBCH or -TBS6H) ¹ |
| RTB screw torque (cage clamp) | 5lb-in. (0.5 Nm) maximum |

| | |
|---|--|
| Conductors | |
| Wire size | 14 gauge (2mm ²) stranded maximum ¹ 3/64 inch (1.2 mm) insulation maximum |
| Category | 2,3 |
| Screwdriver blade width for RTB | 1/8 inch (3.2 mm) maximum |
| Environmental conditions | |
| Operating temperature | 0 to 60°C (32 to 140°F) |
| Storage temperature | -40 to 85°C (-40 to 185°F) |
| Relative humidity | 5 to 95% noncondensing |
| Agency certification (when product or packaging is marked) |  Class 1 Div 2 hazardous marked for all applicable directives FM approved |

¹ Maximum wire size will require the extended depth RTB housing (1756-TBE).

² Use this conductor category information for planning conductor routing as described in the system level installation manual.

³ Refer to *Programmable Controller Wiring and Grounding Guidelines*, publication number 1770-4.1.

Understanding Coarse Update Rate Calculations

To calculate the coarse update rate for the number of modules and axes in your application, you can use the following formula:

$$\text{Baseline task time} + \left[\begin{array}{c} \text{(Actions} \\ \text{for axis 1)} \end{array} + \begin{array}{c} \text{(Actions} \\ \text{for axis 2)} \end{array} + \begin{array}{c} \text{(Actions} \\ \text{for axis n)} \end{array} \right] = \text{Execution time}$$

You can use the sample calculation worksheet in this section to determine your coarse update rate. To determine the values for your equation, refer to the following tables (Table 1 and Table 2).

Defining the Baseline Task Time (Table 1)

The baseline task time is the time to update a number of servo modules. For example, to update 3 modules requires 765 μs .

The following table shows the baseline task times for motion modules.

| Number of modules | Baseline task time (in μs) |
|----------------------|--|
| 1 (2 axes maximum) | 415 |
| 2 (4 axes maximum) | 590 |
| 3 (6 axes maximum) | 765 |
| 4 (8 axes maximum) | 940 |
| 5 (10 axes maximum) | 1115 |
| 6 (12 axes maximum) | 1290 |
| 7 (14 axes maximum) | 1465 |
| 8 (16 axes maximum) | 1640 |
| 9 (18 axes maximum) | 1815 |
| 10 (20 axes maximum) | 1960 |
| 11 (22 axes maximum) | 2165 |
| 12 (24 axes maximum) | 2340 |
| 13 (26 axes maximum) | 2515 |
| 14 (28 axes maximum) | 2690 |
| 15 (30 axes maximum) | 2865 |
| 16 (32 axes maximum) | 3040 |

Understanding Action Timing (Table 2)

Every action performed by an axis requires an amount of time. For example to perform a trapezoidal move requires 440 μ s.

The following table shows execution times for common motion actions.

| Action | Maximum execution time (in μ s) |
|------------------------------------|-------------------------------------|
| Turning the servo on | 60 |
| Performing a trapezoidal move | 440 |
| Performing an s-curve move | 180 |
| Performing a trapezoidal jog | 70 |
| Performing an s-curve jog | 80 |
| Performing an actual position gear | 440 |
| Performing a command position gear | 320 |

Using the Sample Calculations Worksheet

You can use this sample calculation worksheet to determine the coarse update time for the number of modules in your application.

1. Complete the following table.

| | | | |
|--|---|---|----|
| System | | | |
| Describe the type of system you are using. | | | |
| | 1 | Enter the number of modules. | |
| | 2 | Enter baseline task time (from Table 1) | μs |

2. For each axis in your application, use the following table to determine the action value for each axis.

| | | | |
|--|----|--|----|
| Actions | | | |
| If you are using an action, enter its execution time shown in Table 2. | | | |
| If you are not using an action, enter zero (0). | | | |
| | 3 | Servo on | μs |
| | 4 | Trapezoidal move | μs |
| | 5 | S-curve move | μs |
| | 6 | Trapezoidal jog | μs |
| | 7 | S-curve jog | μs |
| | 8 | Actual gear | μs |
| | 9 | Command gear | μs |
| | 10 | Add lines 3 through 9. Place total here. | μs |

3. Calculate the coarse rate you want for your application.

| | | | |
|-------------------------------------|----|-------------------------------------|----|
| Coarse rate | | | |
| Determine the coarse rate you want. | | | |
| | 11 | The coarse rate you want | μs |
| | 12 | $0.80 * \text{number from line 11}$ | μs |

4. To calculate the coarse update rate for your application, add line 2 to the action value for each axis (line 10).
5. If your coarse update rate from step 4 is less than line 12, you can achieve your coarse update rate (line 11) with your current system.

Understanding Sample Calculation 1

You have the following situation:

- You have a system consisting of 2 modules and 4 axes.
- You are turning the servo on and performing a trapezoidal move for each axis.
- You want a coarse update rate of 4 ms.

1. Complete the following table.

| System | | | |
|--|---|---|-------------|
| Describe the type of system you are using. | | | |
| | 1 | Enter the number of modules. | 2 |
| | 2 | Enter baseline task time (from Table 1) | 590 μ s |

2. For each axis in your application, use the following table to determine the action value for each axis.

Axes 1, 2, 3, and 4:

| Actions | | | |
|--|----|--|-------------|
| If you are using an action, enter its execution time shown in Table 2. | | | |
| If you are not using an action, enter zero (0). | | | |
| | 3 | Servo on | 60 μ s |
| | 4 | Trapezoidal move | 440 μ s |
| | 5 | S-curve move | μ s |
| | 6 | Trapezoidal jog | μ s |
| | 7 | S-curve jog | μ s |
| | 8 | Actual gear | μ s |
| | 9 | Command gear | μ s |
| | 10 | Add lines 3 through 9. Place total here. | 500 μ s |

3. Calculate the coarse rate you want for your application.

Coarse rate

Determine the coarse rate you want.

| | | | |
|--|----|-------------------------------------|--------------------|
| | 11 | The coarse rate you want | 4000 μs |
| | 12 | $0.80 * \text{number from line 11}$ | 3200 μs |

4. The calculated coarse rate for this application is

| | |
|-----------------------------------|--------------------|
| Baseline task time (line 2) | 590 μs |
| Action value for axis 1 (line 10) | 500 μs |
| Action value for axis 2 (line 10) | 500 μs |
| Action value for axis 3 (line 10) | 500 μs |
| Action value for axis 4 (line 10) | 500 μs |
| TOTAL | 2590 μs |

Conclusion: You can achieve the coarse update rate with your system because 2590 μs is less than 3200 μs .

Understanding Sample Calculation 2

You have the following situation:

- You have a system consisting of 2 modules and 4 axes.
- You are turning the servo on and performing a trapezoidal move for each axis.
- You want a coarse update rate of 3 ms.

1. Complete the following table.

| System | | | |
|--|---|---|-------------|
| Describe the type of system you are using. | | | |
| | 1 | Enter the number of modules. | 2 |
| | 2 | Enter baseline task time (from Table 1) | 590 μ s |

2. For each axis in your application, use the following table to determine the action value for each axis.

Axes 1, 2, 3, and 4:

| Actions | | | |
|--|----|--|-------------|
| If you are using an action, enter its execution time shown in Table 2. | | | |
| If you are not using an action, enter zero (0). | | | |
| | 3 | Servo on | 60 μ s |
| | 4 | Trapezoidal move | 440 μ s |
| | 5 | S-curve move | μ s |
| | 6 | Trapezoidal jog | μ s |
| | 7 | S-curve jog | μ s |
| | 8 | Actual gear | μ s |
| | 9 | Command gear | μ s |
| | 10 | Add lines 3 through 9. Place total here. | 500 μ s |

3. Calculate the coarse rate you want for your application.

Coarse rate

Determine the coarse rate you want.

| | | | |
|--|----|-------------------------------------|--------------|
| | 11 | The coarse rate you want | 3000 μ s |
| | 12 | $0.80 * \text{number from line 11}$ | 2400 μ s |

4. The calculated coarse rate for this application is

| | |
|-----------------------------------|--------------|
| Baseline task time (line 2) | 590 μ s |
| Action value for axis 1 (line 10) | 500 μ s |
| Action value for axis 2 (line 10) | 500 μ s |
| Action value for axis 3 (line 10) | 500 μ s |
| Action value for axis 4 (line 10) | 500 μ s |
| TOTAL | 2590 μ s |

Conclusion: You can not achieve the coarse update rate with your system because 2590 μ s is greater than 2400 μ s.

Loop and Interconnect Diagrams

This appendix shows the loop interconnect diagrams for common motion configurations. The following table shows the contents of this appendix:

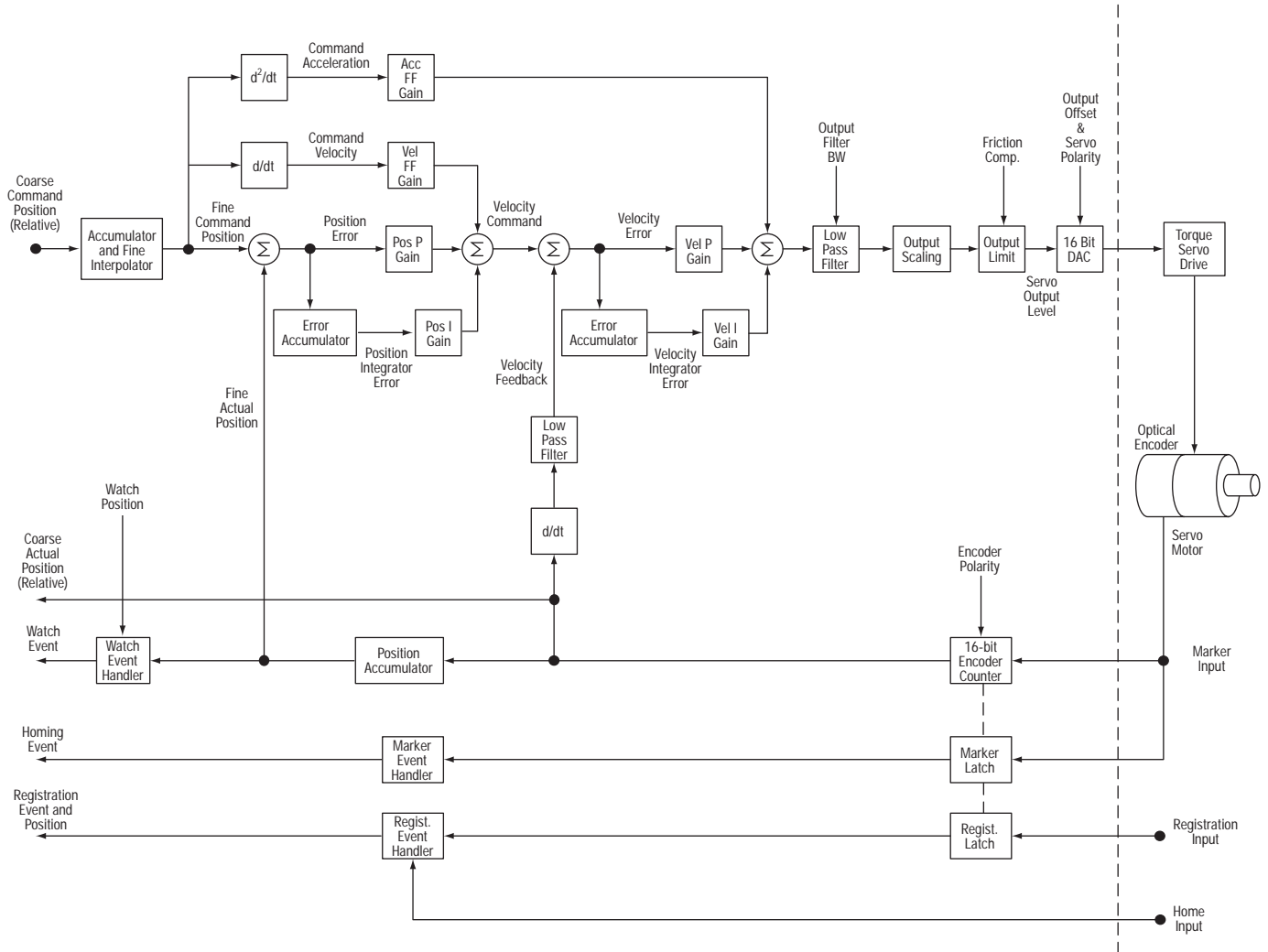
| For information about | See page |
|---|-----------------|
| Understanding Block Diagrams | B-2 |
| Using a 1756-M02AE Module With a Torque Servo Drive | B-3 |
| Using a 1756-M02AE Module With a Velocity Servo Drive | B-4 |
| Understanding Wiring Diagrams | B-5 |
| Wiring to a Servo Module RTB | B-5 |
| Wiring to an Ultra 100 Series Drive | B-6 |
| Wiring to an Ultra 200 Series Drive | B-7 |
| Wiring to a 1394 Servo Drive | B-8 |
| Wiring the 1394-SA15 Cable | B-9 |
| Wiring Registration Sensors | B-10 |
| Wiring the Home Limit Switch Input | B-11 |
| Wiring the OK Contacts | B-12 |

Understanding Block Diagrams

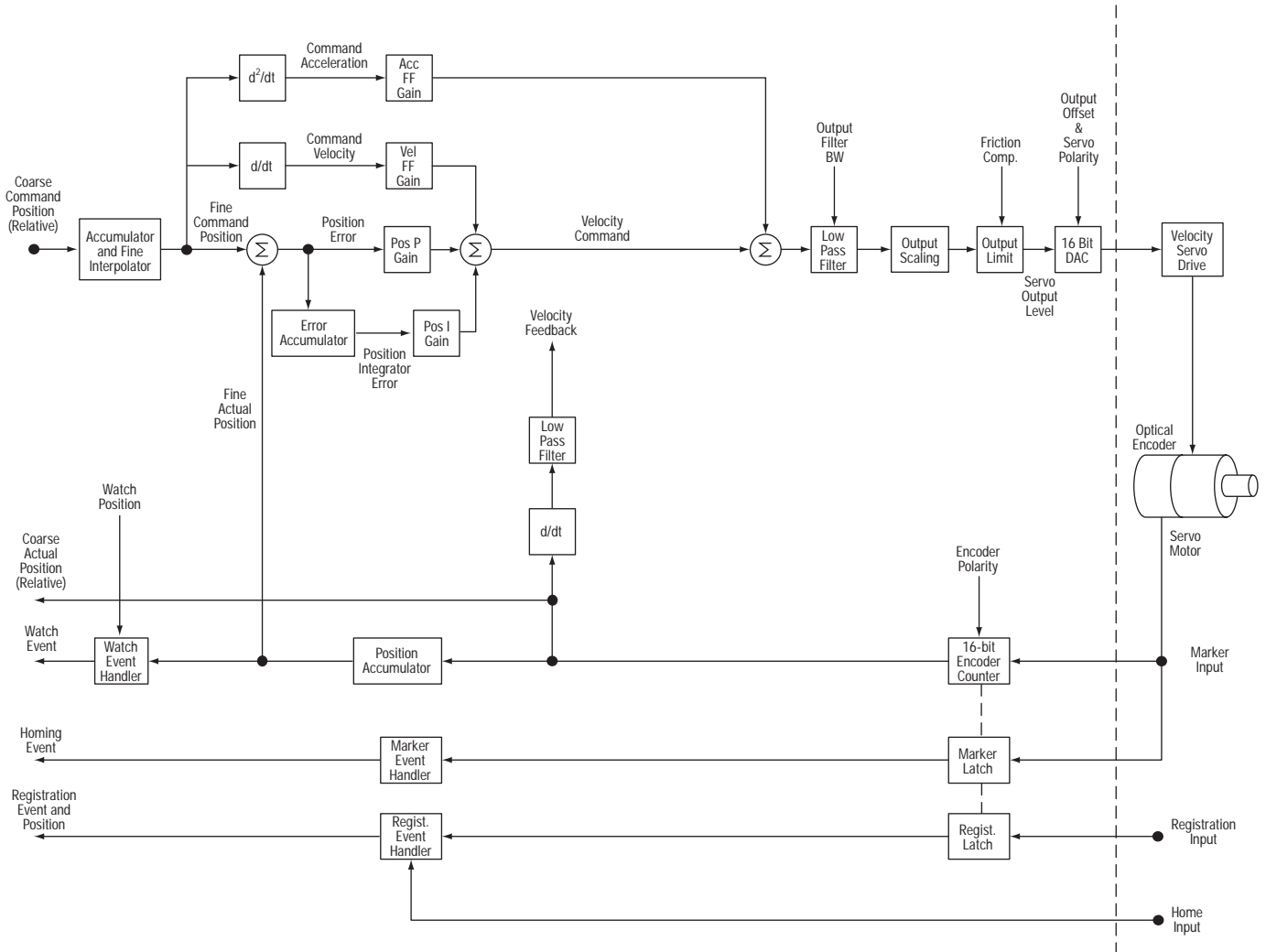
The control block diagrams in this section use the following terms for motion attributes.

| Diagram term | Motion attribute name (as used in the GSV and SSV instructions) |
|---------------------------|---|
| Acc FF Gain | AccelerationFeedforwardGain |
| Vel FF Gain | VelocityFeedforwardGain |
| Pos P Gain | PositionProportionalGain |
| Pos I Gain | PositionIntegralGain |
| Vel P Gain | VelocityProportionalGain |
| Vel I Gain | VelocityIntegralGain |
| Output Filter BW | OutputFilterBandwidth |
| Output Scaling | OutputScaling |
| Friction Comp | FrictionCompensation |
| Output Limit | OutputLimit |
| Output Offset | OutputOffset |
| Position Error | PositionError |
| Position Integrator Error | PositionIntegratorError |
| Velocity Error | VelocityError |
| Velocity Integrator Error | VelocityIntegratorError |
| Velocity Feedback | VelocityFeedback |
| Velocity Command | VelocityCommand |
| Servo Output Level | ServoOutputLevel |
| Registration Position | RegistrationPosition |
| Watch Position | WatchPosition |

Using a 1756-M02AE Module With a Torque Servo Drive

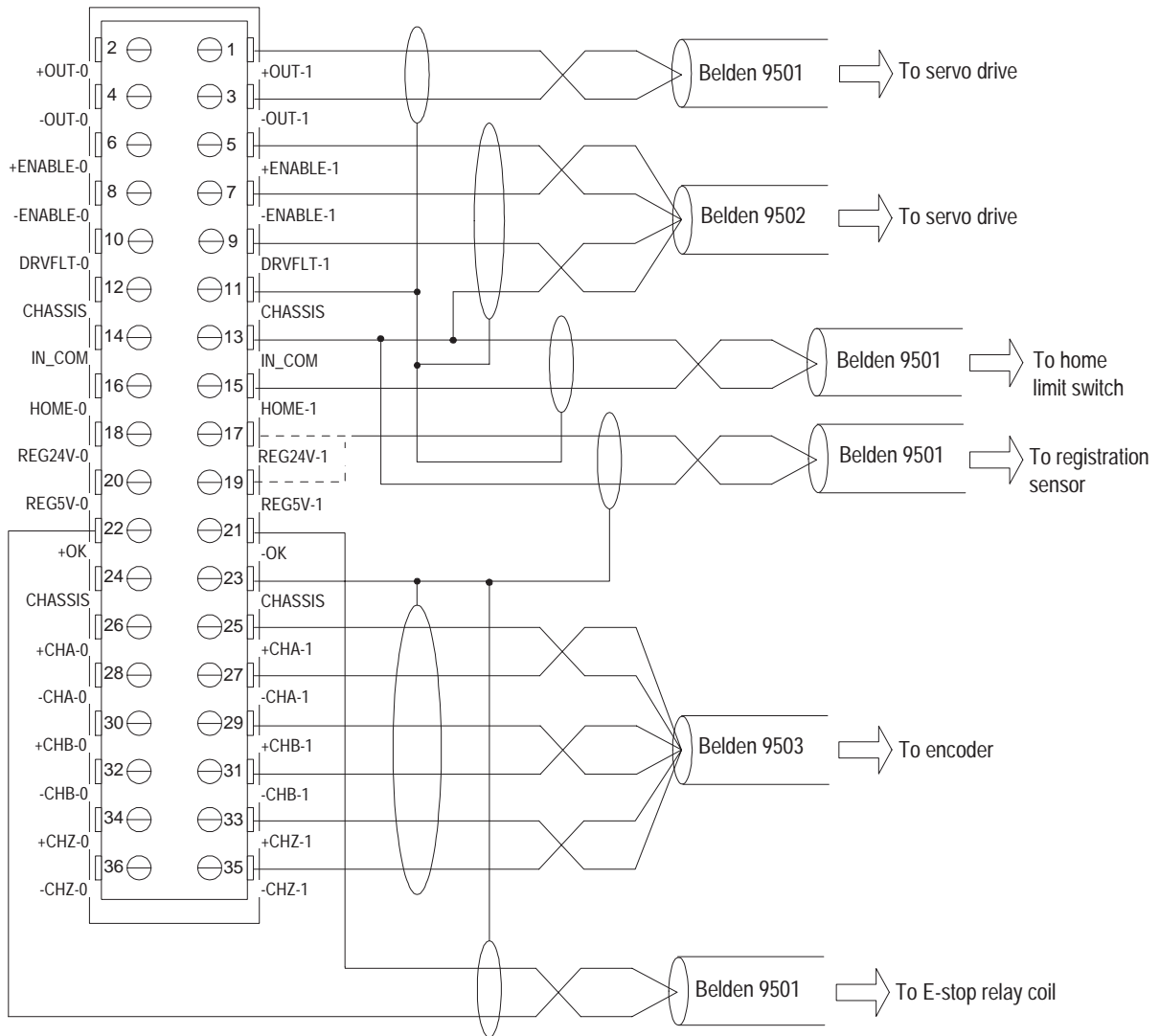


Using a 1756-M02AE Module With a Velocity Servo Drive



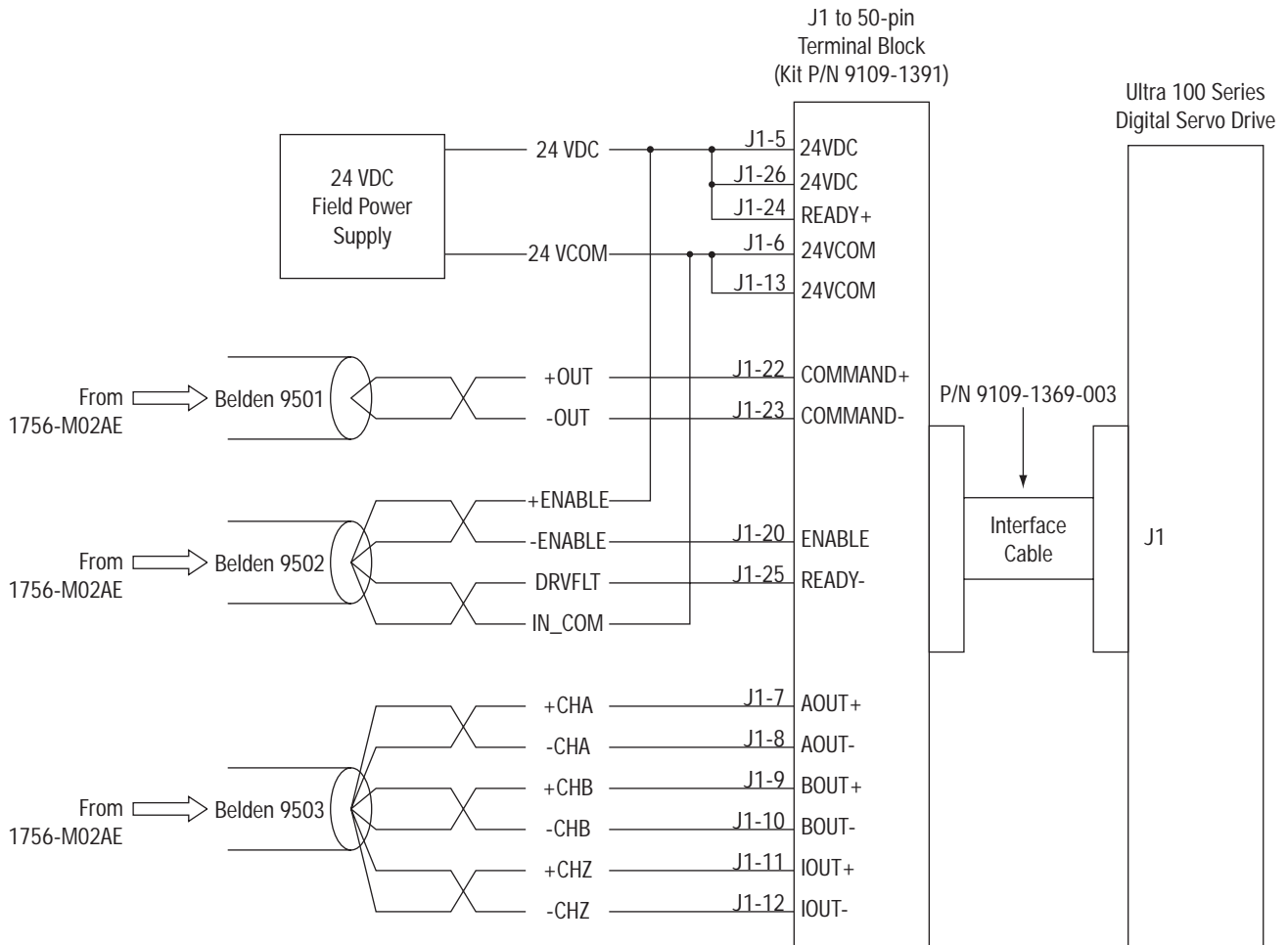
Understanding Wiring Diagrams

Wiring to a Servo Module RTB



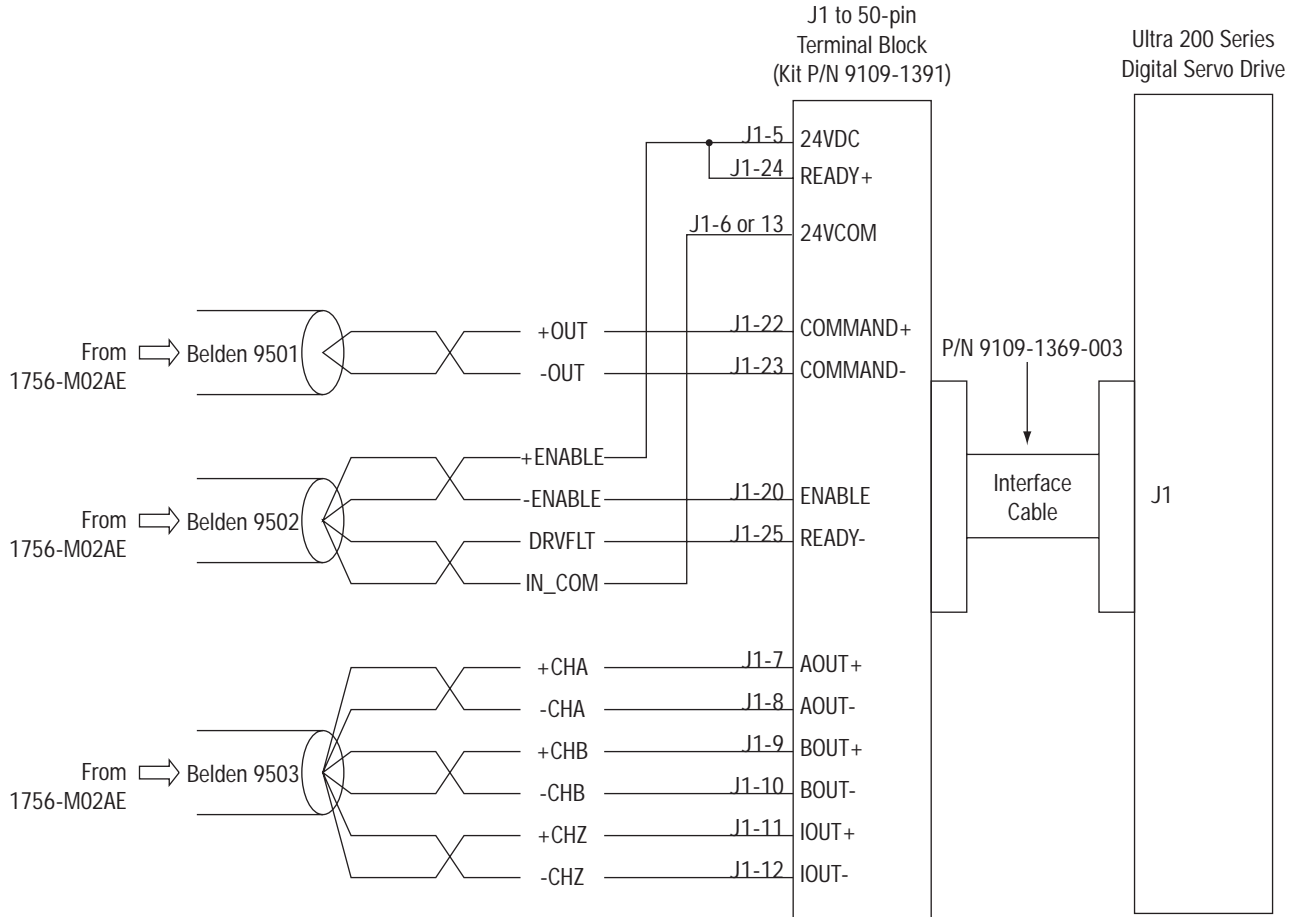
Note: This is a general wiring example illustrating Axis 1 wiring only. Other configurations are possible with Axis 0 wiring identical to Axis 1.

Wiring to an Ultra 100 Series Drive



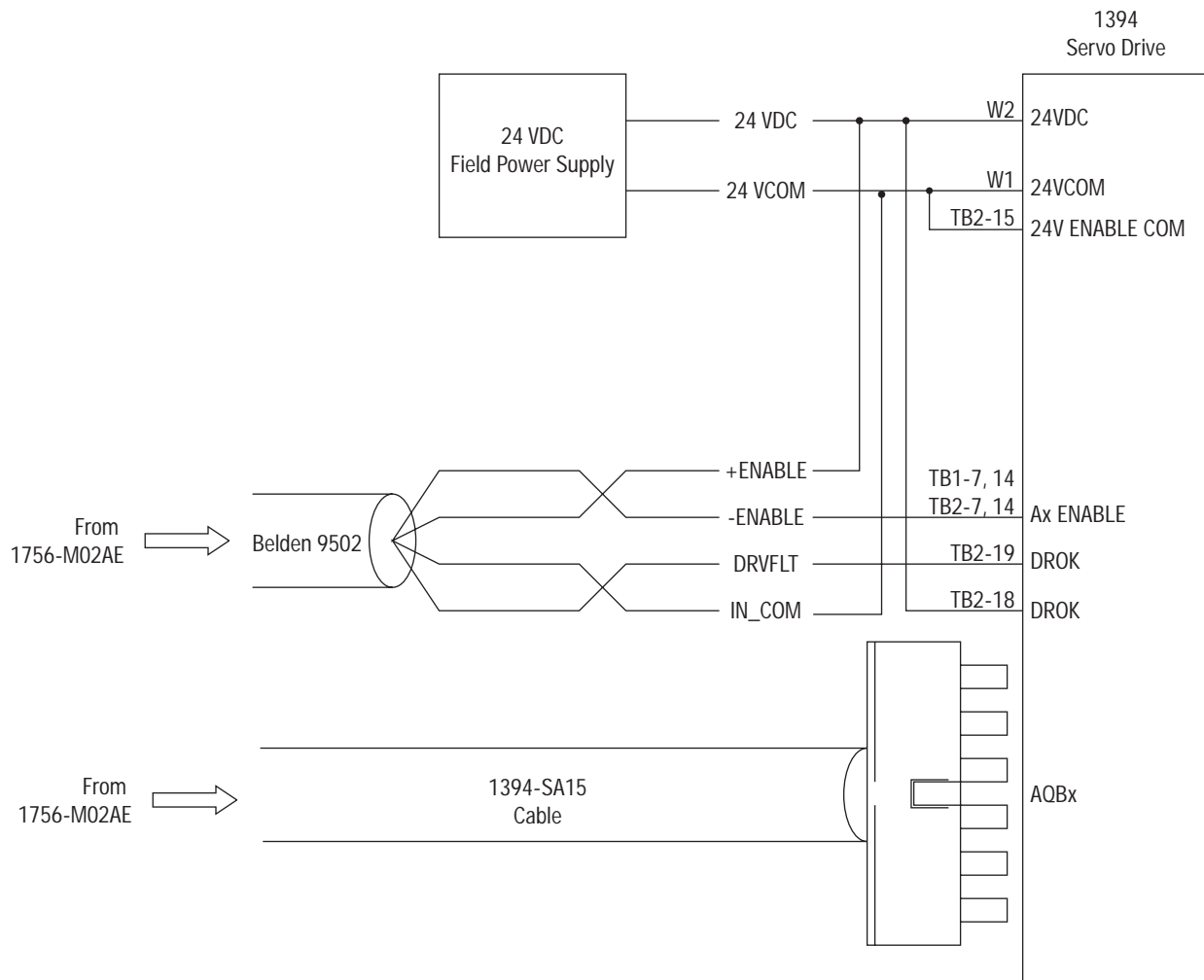
Note: This is a general wiring example only. Other configurations are possible. For more information, refer to the Ultra 100 Series Drive Installation Manual, publication number 1398-5.2.

Wiring to an Ultra 200 Series Drive



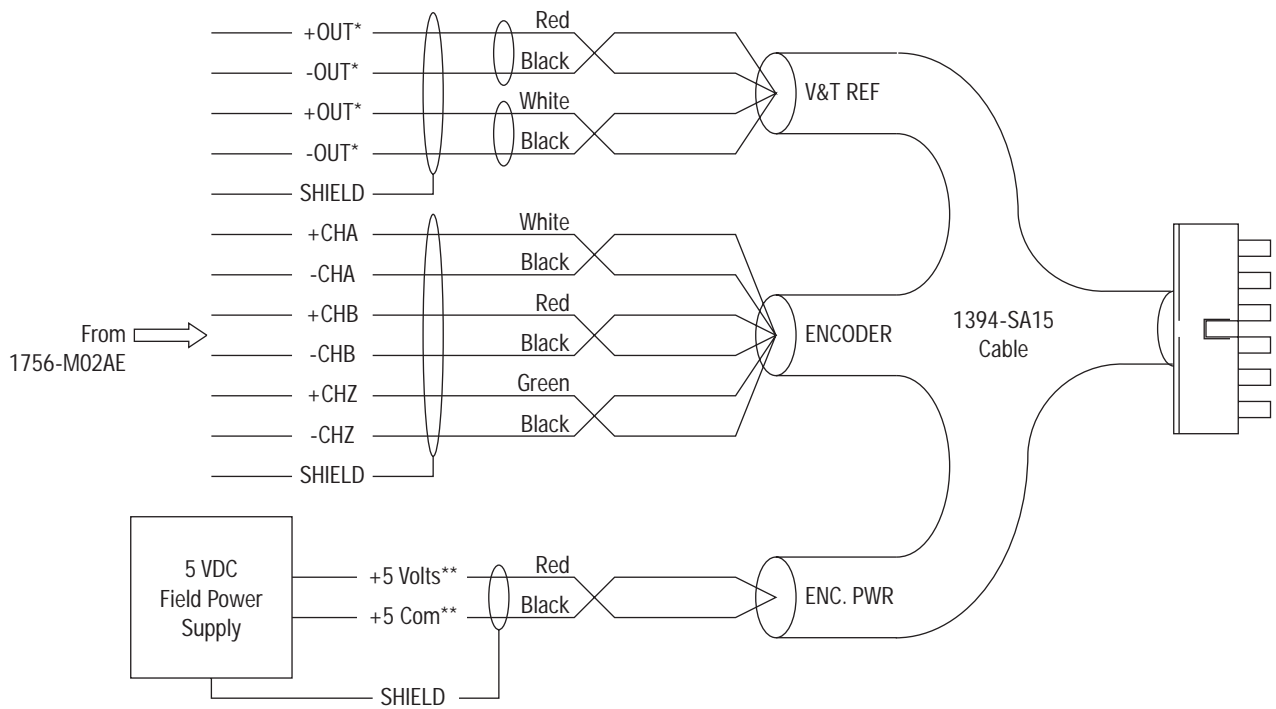
Note: This is a general wiring example only. Other configurations are possible. For more information, refer to the Ultra 200 Series Drive Installation Manual, publication number 1398-5.0.

Wiring to a 1394 Servo Drive



Note: This is a general wiring example only. Other configurations are possible. The x in the diagram is the 1394 axis reference number (0, 1, 2, or 3) specifying one of the four possible 1394 servo drive axes. For more information, refer to the 1394 Digital Multi-Axis Motion Control System User Manual, publication number 1394-5.0.

Wiring the 1394-SA15 Cable



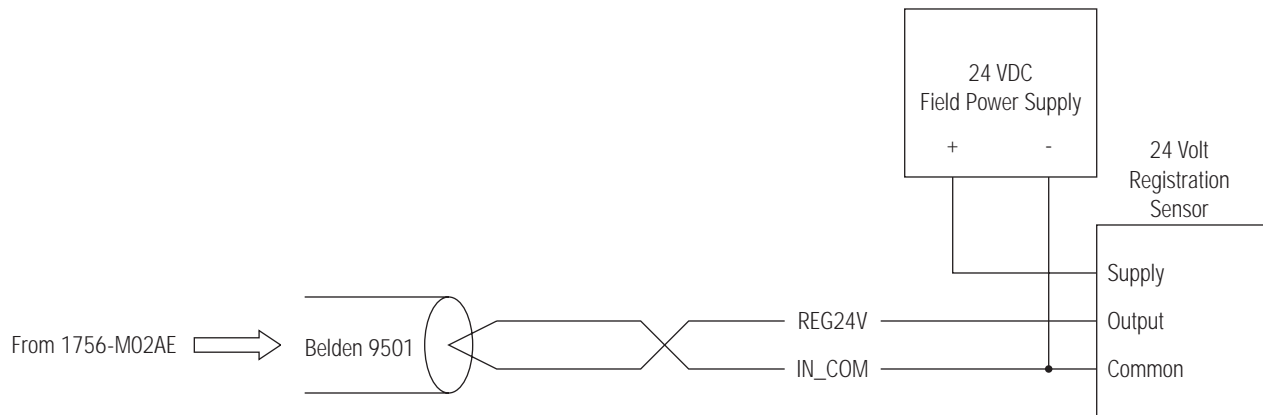
* Connect only one analog input pair to the output of the servo module. If you configured the 1394 axis as a velocity servo drive, then connect the VREF signals. If you configured the 1394 axis as a torque servo drive, then connect the TREF signals.

** An external +5V power supply is necessary to power the encoder driver circuit of the 1394 servo drive. The four axis encoder driver circuits share this power supply. Only one connection is needed to the +5V field supply.

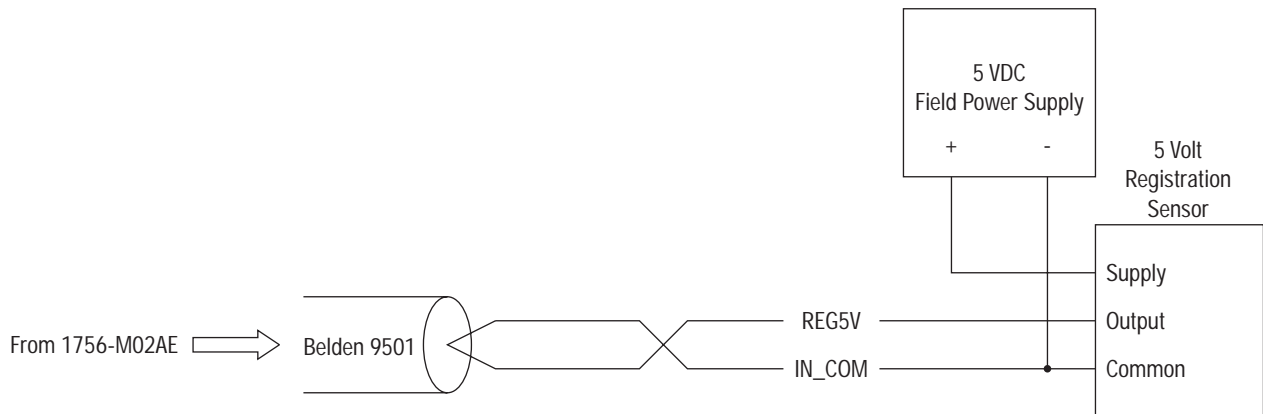
Wiring Registration Sensors

The registration inputs to the servo module can support 24V or 5V registration sensors. These inputs should be wired to receive source current from the sensor. Current sinking sensor configurations are not allowed because the registration input common (IN_COM) is shared with the other 24V servo module inputs.

24V Registration Sensor

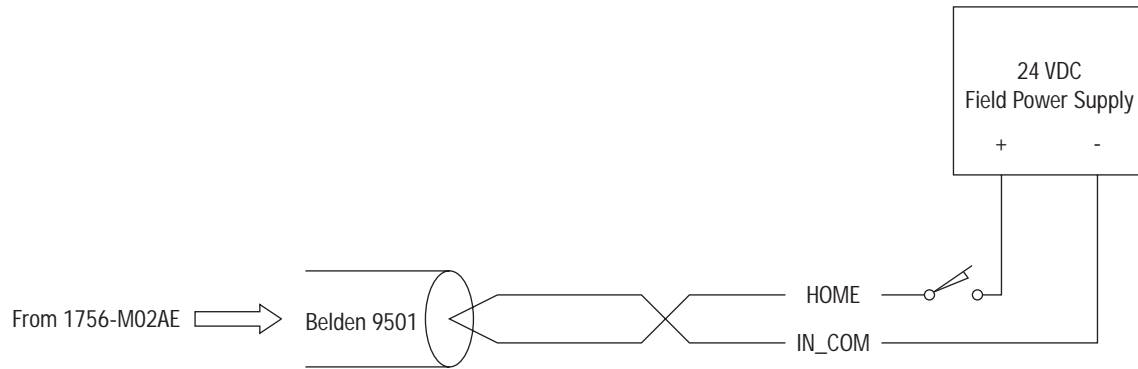


5V Registration Sensor



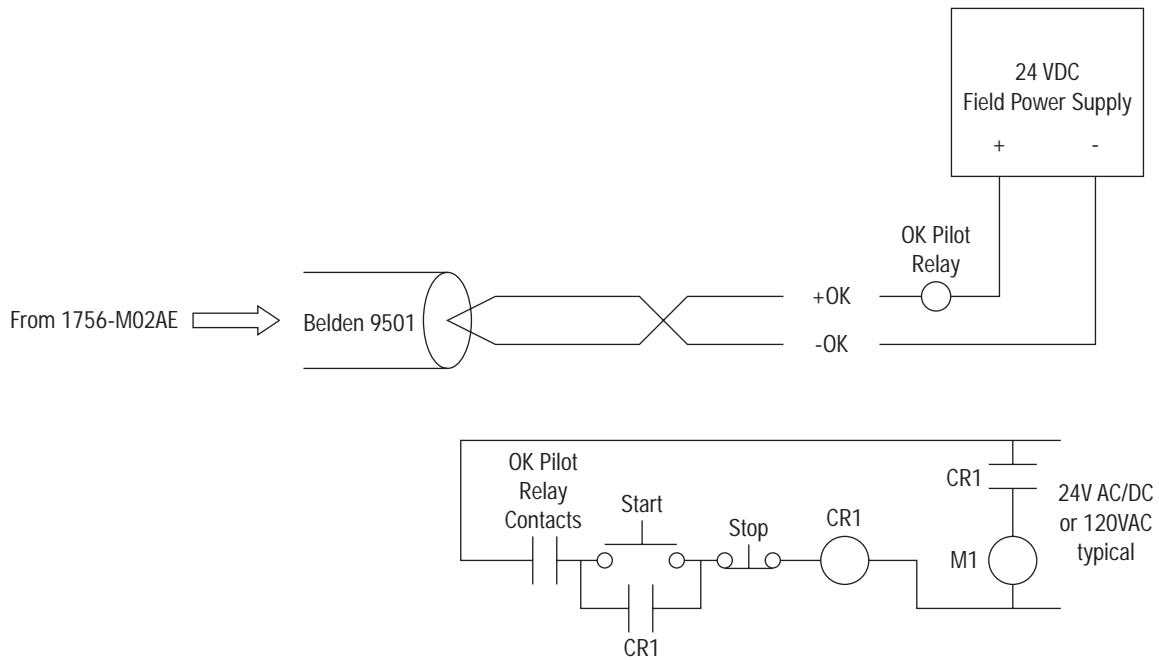
Wiring the Home Limit Switch Input

The home limit switch inputs to the servo module are designed for 24V nominal operation. These inputs should be wired for current sourcing operation.



Wiring the OK Contacts

A set of isolated solid-state OK relay contacts is provided for optional interface to an E-stop string, which controls power to the associated drives. The OK contacts are rated to drive an external 24V pilot relay (for example, Allen-Bradley 700-HA32Z24) whose contacts can be incorporated into the E-Stop string as shown below.



The Motion Control Structures

This appendix shows the structures for the AXIS, MOTION_GROUP, and MOTION_INSTRUCTION data tags. The following table shows the contents of this appendix:

| For information about | See page |
|--|-----------------|
| Understanding the AXIS Structure | C-2 |
| Understanding the MOTION_GROUP Structure | C-8 |
| Understanding the MOTION_INSTRUCTION Structure | C-11 |

Understanding the AXIS Structure

The AXIS structure contains status and configuration information for your motion axis. You can directly access this information in your application program. For example, if you want to use the `AccelStatus` attribute for `Axis_X`, you would use `Axis_X.AccelStatus` to gain access to the attribute.

| Variable | Data Type | Description |
|---------------------------------|-----------|---|
| <code>.AccelStatus</code> | BOOL | You can use this bit to determine if the axis has been commanded to accelerate. If neither this bit nor the <code>.DecelStatus</code> bit is set, the axis is running at the steady-state velocity or is at rest. |
| <code>.ACAsyncConnFault</code> | BOOL | You can use this bit to determine the status of asynchronous communication. When the controller detects that one of the servo module parameters failed to update because of an asynchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. |
| <code>.ACSyncConnFault</code> | BOOL | You can use this bit to determine the status of synchronous communication. When the controller detects that the servo module has missed several position updates in a row because of a synchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. |
| <code>.AxisHomedStatus</code> | BOOL | You can use this bit to determine the status of a homing sequence. During power-up or reconnection, the controller clears this bit. The Motion Axis Home (MAH) instruction sets this bit when a homing sequence completes successfully. After this bit sets, if the axis enters the shutdown state, the controller clears this bit. |
| <code>.Clutch Status</code> | BOOL | You can use this bit to determine if a clutch motion profile is in progress. If this bit is set, a clutch motion profile is currently in progress. This bit is clear when the clutch process is complete. |
| <code>.DecelStatus</code> | BOOL | You can use this bit to determine if the axis has been commanded to decelerate. If neither this bit nor the <code>.AccelStatus</code> bit is set, the axis is running at the steady-state velocity or is at rest. |
| <code>.DriveEnableStatus</code> | BOOL | You can use this bit to determine the status of the drive enable output. If this bit is set, you have activated the drive enable output for your axis. This bit is clear if you have deactivated the drive enable output for your axis. |
| <code>.DriveFault</code> | BOOL | You can use this bit to determine the status of the external drive. If this bit is set, the external drive detected a fault. This bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. |
| <code>.EncCHALossFault</code> | BOOL | You can use this bit to determine the status of the encoder channel A. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. |
| <code>.EncCHBLossFault</code> | BOOL | You can use this bit to determine the status of the encoder channel B. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. |
| <code>.EncCHZLossFault</code> | BOOL | You can use this bit to determine the status of the encoder channel Z. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. |
| <code>.EncNsFault</code> | BOOL | You can use this bit to determine the status of encoder channels A and B. If the servo module detects simultaneous transitions of channels A and B, this bit sets. This bit is clear after the controller executes a Motion Axis Fault Reset (MAFR) instruction. |

| Variable | Data Type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------------------|-----------|---|------------------------------|-----------|-------------------------------|-------------|-------------------|----|------|-------------------------------|------------------|----|------|------------------------------|-----------------|----|------|--------------------------|--------------|----|------|--------------------|------------------|----|------|------------------|---------------|----|------|------------|---------------|----|------|--------|------------------|----|------|--------------|
| .EventStatus | DINT | The servo event bits for your servo loop. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.WatchEvArmStatus</td> <td>00</td> <td>BOOL</td> <td>watch event armed</td> </tr> <tr> <td>.WatchEvStatus</td> <td>01</td> <td>BOOL</td> <td>watch event</td> </tr> <tr> <td>.RegEvArmStatus</td> <td>02</td> <td>BOOL</td> <td>registration event armed</td> </tr> <tr> <td>.RegEvStatus</td> <td>03</td> <td>BOOL</td> <td>registration event</td> </tr> <tr> <td>.HomeEvArmStatus</td> <td>04</td> <td>BOOL</td> <td>home event armed</td> </tr> <tr> <td>.HomeEvStatus</td> <td>05</td> <td>BOOL</td> <td>home event</td> </tr> </tbody> </table> | Bit | Number | Data type | Description | .WatchEvArmStatus | 00 | BOOL | watch event armed | .WatchEvStatus | 01 | BOOL | watch event | .RegEvArmStatus | 02 | BOOL | registration event armed | .RegEvStatus | 03 | BOOL | registration event | .HomeEvArmStatus | 04 | BOOL | home event armed | .HomeEvStatus | 05 | BOOL | home event | | | | | | | | |
| | | Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .WatchEvArmStatus | 00 | BOOL | watch event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .WatchEvStatus | 01 | BOOL | watch event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .RegEvArmStatus | 02 | BOOL | registration event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .RegEvStatus | 03 | BOOL | registration event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .HomeEvArmStatus | 04 | BOOL | home event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .HomeEvStatus | 05 | BOOL | home event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bits 06 through 31 are reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .GearingStatus | BOOL | You can use this bit to determine if electronic gearing is enabled. If this bit is set, the axis is currently gearing to another axis. This bit is clear when the gearing operation stops or when another motion operation supersedes the gearing operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .Hardfault | BOOL | You can use this bit to determine the status of the servo module. If this bit is set, the servo module detected a hardware problem that typically requires the replacement of the servo module. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .HomeEvArmStatus | BOOL | You can use this bit to determine the status of a homing event. If this bit is set, a Motion Axis Home (MAH) instruction has armed a home event. This bit clears when a home event occurs. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .HomeEvStatus | BOOL | You can use this bit to determine the status of a homing event. If this bit is set, a home event has occurred. This bit clears when another Motion Axis Home (MAH) instruction executes. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .HomingStatus | BOOL | You can use this bit to determine if a homing profile is in progress. If this bit is set, a homing profile is currently in progress. This bit is clear when the homing operation completes or when another motion operation supersedes the homing operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .JogStatus | BOOL | You can use this bit to determine if a jog profile is in progress. If this bit is set, a jog profile is currently in progress. This bit is clear when the jog completes or when another motion operation supersedes the jog operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .MotionFault | DINT | The motion fault bits for your axis. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.ACAsyncConnFault</td> <td>00</td> <td>BOOL</td> <td>asynchronous connection fault</td> </tr> <tr> <td>.ACSyncConnFault</td> <td>01</td> <td>BOOL</td> <td>synchronous connection fault</td> </tr> </tbody> </table> | Bit | Number | Data type | Description | .ACAsyncConnFault | 00 | BOOL | asynchronous connection fault | .ACSyncConnFault | 01 | BOOL | synchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .ACAsyncConnFault | 00 | BOOL | asynchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ACSyncConnFault | 01 | BOOL | synchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bits 02 through 31 are reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .MotionStatus | DINT | The motion status bits for your axis. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .MotionStatus | DINT | <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.AccelStatus</td> <td>00</td> <td>BOOL</td> <td>acceleration</td> </tr> <tr> <td>.DecelStatus</td> <td>01</td> <td>BOOL</td> <td>deceleration</td> </tr> <tr> <td>.MoveStatus</td> <td>02</td> <td>BOOL</td> <td>move</td> </tr> <tr> <td>.JogStatus</td> <td>03</td> <td>BOOL</td> <td>jog</td> </tr> <tr> <td>.GearingStatus</td> <td>04</td> <td>BOOL</td> <td>gear</td> </tr> <tr> <td>.HomingStatus</td> <td>05</td> <td>BOOL</td> <td>homing</td> </tr> <tr> <td>.ClutchStatus</td> <td>06</td> <td>BOOL</td> <td>clutch</td> </tr> <tr> <td>.AxisHomedStatus</td> <td>07</td> <td>BOOL</td> <td>homed status</td> </tr> </tbody> </table> | Bit | Number | Data type | Description | .AccelStatus | 00 | BOOL | acceleration | .DecelStatus | 01 | BOOL | deceleration | .MoveStatus | 02 | BOOL | move | .JogStatus | 03 | BOOL | jog | .GearingStatus | 04 | BOOL | gear | .HomingStatus | 05 | BOOL | homing | .ClutchStatus | 06 | BOOL | clutch | .AxisHomedStatus | 07 | BOOL | homed status |
| | | Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .AccelStatus | 00 | BOOL | acceleration | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .DecelStatus | 01 | BOOL | deceleration | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .MoveStatus | 02 | BOOL | move | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .JogStatus | 03 | BOOL | jog | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .GearingStatus | 04 | BOOL | gear | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .HomingStatus | 05 | BOOL | homing | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ClutchStatus | 06 | BOOL | clutch | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .AxisHomedStatus | 07 | BOOL | homed status | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bits 08 through 31 are reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Variable | Data Type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------|-----------|--|------------------------------|--------|-----------|-------------|--------------|----|------|---------------------------|--------------|----|------|---------------------------|----------------|----|------|----------------------|------------------|----|------|------------------------------|------------------|----|------|------------------------------|------------------|----|------|------------------------------|-------------|----|------|---------------------|-------------|----|------|-------------|----------------|----|------|------------------------------|------------|----|------|----------------------|
| .MoveStatus | BOOL | You can use this bit to determine if a move profile is in progress. If this bit is set, a move profile is currently in progress. This bit is clear when the move completes or when another motion operation supersedes the move operation. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .NOtrvIFault | BOOL | You can use this bit to determine the status of axis travel. If this bit is set, the axis has moved or has attempted to move beyond the MaximumNegativeOvertravel value. This bit is clear when the axis moves within the MaximumNegativeOvertravel values | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .OutLmtStatus | BOOL | You can use this bit to determine the status of servo loop output. If the magnitude of the servo loop output reaches or exceeds the OutputLimit value, this bit sets. This bit is clear when the magnitude of the servo loop output is within the OutputLimit value. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .PosErrorFault | BOOL | You can use this bit to determine the status of axis position error. If this bit is set, the servo module has detected that axis position error exceeds the PositionErrorTolerance value. This bit is clear when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .PosLockStatus | BOOL | You can use this bit to determine the status of the axis position error. If this bit is set, the magnitude of the axis position error is less than or equal to the PositionLockTolerance value. This bit is clear when the magnitude of the axis position error is greater than the PositionLockTolerance value. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .POtrvIFault | BOOL | You can use this bit to determine the status of axis travel. If this bit is set, the axis has moved or has attempted to move beyond the MaximumPositiveOvertravel value. This bit is clear when the axis moves within the MaximumPositiveOvertravel values | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .RegEvArmStatus | BOOL | You can use this bit to determine the status of a registration event. If this bit is set, the execution of a Motion Arm Registration (MAR) instruction has armed a registration event. This bit clears when a registration event occurs or the controller executes a Motion Disarm Registration (MDR) instruction. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .RegEvStatus | BOOL | You can use this bit to determine the status of a registration event. If this bit is set, a registration event has occurred. This bit clears when the controller executes another Motion Arm Registration (MAR) instruction or a Motion Disarm Registration (MDR) instruction. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ServoActStatus | BOOL | You can use this bit to determine if servo action is enabled for your axis. If this bit is set, servo action is currently enabled. This bit is clear when servo action is disabled. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ServoFault | DINT | <p>The servo fault bits for your servo loop.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.POtrvIFault</td> <td>00</td> <td>BOOL</td> <td>positive overtravel fault</td> </tr> <tr> <td>.NOtrvIFault</td> <td>01</td> <td>BOOL</td> <td>negative overtravel fault</td> </tr> <tr> <td>.PosErrorFault</td> <td>02</td> <td>BOOL</td> <td>position error fault</td> </tr> <tr> <td>.EncCHALossFault</td> <td>03</td> <td>BOOL</td> <td>encoder channel A loss fault</td> </tr> <tr> <td>.EncCHBLossFault</td> <td>04</td> <td>BOOL</td> <td>encoder channel B loss fault</td> </tr> <tr> <td>.EncCHZLossFault</td> <td>05</td> <td>BOOL</td> <td>encoder channel Z loss fault</td> </tr> <tr> <td>.EncNsFault</td> <td>06</td> <td>BOOL</td> <td>encoder noise fault</td> </tr> <tr> <td>.DriveFault</td> <td>07</td> <td>BOOL</td> <td>drive fault</td> </tr> <tr> <td>.SyncConnFault</td> <td>08</td> <td>BOOL</td> <td>synchronous connection fault</td> </tr> <tr> <td>.Hardfault</td> <td>09</td> <td>BOOL</td> <td>servo hardware fault</td> </tr> </tbody> </table> <p>Bits 10 through 31 are reserved.</p> | Bit | Number | Data type | Description | .POtrvIFault | 00 | BOOL | positive overtravel fault | .NOtrvIFault | 01 | BOOL | negative overtravel fault | .PosErrorFault | 02 | BOOL | position error fault | .EncCHALossFault | 03 | BOOL | encoder channel A loss fault | .EncCHBLossFault | 04 | BOOL | encoder channel B loss fault | .EncCHZLossFault | 05 | BOOL | encoder channel Z loss fault | .EncNsFault | 06 | BOOL | encoder noise fault | .DriveFault | 07 | BOOL | drive fault | .SyncConnFault | 08 | BOOL | synchronous connection fault | .Hardfault | 09 | BOOL | servo hardware fault |
| Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .POtrvIFault | 00 | BOOL | positive overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .NOtrvIFault | 01 | BOOL | negative overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .PosErrorFault | 02 | BOOL | position error fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .EncCHALossFault | 03 | BOOL | encoder channel A loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .EncCHBLossFault | 04 | BOOL | encoder channel B loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .EncCHZLossFault | 05 | BOOL | encoder channel Z loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .EncNsFault | 06 | BOOL | encoder noise fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .DriveFault | 07 | BOOL | drive fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .SyncConnFault | 08 | BOOL | synchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .Hardfault | 09 | BOOL | servo hardware fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Variable | Data Type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--------------------|----------------------------------|---|-----------------|---------|-----------|-----------------------|-----------------|----------------------------------|------|-----------------------|--------------------|----------------------------------|------|-------------------------|---------------|--------------------------|------|---------------------------|----------------|----|------|---------------|-------------|----|------|----------------|-------------|----|------|-----------------|-----------------|----|------|---------------|
| .ServoStatus | DINT | <p>The status bits for your servo loop.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.ServoActStatus</td> <td>00</td> <td>BOOL</td> <td>servo action</td> </tr> <tr> <td>.DriveEnableStatus</td> <td>01</td> <td>BOOL</td> <td>drive enable</td> </tr> <tr> <td>.OutLmtStatus</td> <td>02</td> <td>BOOL</td> <td>output limit</td> </tr> <tr> <td>.PosLockStatus</td> <td>03</td> <td>BOOL</td> <td>position lock</td> </tr> <tr> <td>.TuneStatus</td> <td>13</td> <td>BOOL</td> <td>tuning process</td> </tr> <tr> <td>.TestStatus</td> <td>14</td> <td>BOOL</td> <td>test diagnostic</td> </tr> <tr> <td>.ShutdownStatus</td> <td>15</td> <td>BOOL</td> <td>axis shutdown</td> </tr> </tbody> </table> <p>Bits 04 through 12 and bits 16 through 31 are reserved.</p> | Bit | Number | Data type | Description | .ServoActStatus | 00 | BOOL | servo action | .DriveEnableStatus | 01 | BOOL | drive enable | .OutLmtStatus | 02 | BOOL | output limit | .PosLockStatus | 03 | BOOL | position lock | .TuneStatus | 13 | BOOL | tuning process | .TestStatus | 14 | BOOL | test diagnostic | .ShutdownStatus | 15 | BOOL | axis shutdown |
| Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ServoActStatus | 00 | BOOL | servo action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .DriveEnableStatus | 01 | BOOL | drive enable | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .OutLmtStatus | 02 | BOOL | output limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .PosLockStatus | 03 | BOOL | position lock | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .TuneStatus | 13 | BOOL | tuning process | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .TestStatus | 14 | BOOL | test diagnostic | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ShutdownStatus | 15 | BOOL | axis shutdown | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .ShutdownStatus | BOOL | You can use this bit to determine if your axis is in the shutdown state. If this bit is set, the axis is in the shutdown state. This bit is clear when the axis transitions from the shutdown state to another state. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .SyncConnFault | BOOL | You can use this bit to determine the status of synchronous communication. When the servo module detects that it has missed several position updates in a row because of a synchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .TestStatus | BOOL | You can use this bit to determine the status of diagnostic tests. If this bit is set, a diagnostic test operation is in progress for the servo module. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .TuneStatus | BOOL | You can use this bit to determine the status of axis tuning. If this bit is set, an auto tuning operation is in progress for the servo module. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .UpdateStatus | DINT | <p>You can use these bits to determine the configuration parameters for your axis. If a bit is set, you configured the axis to update the associated attribute. For example, if the position error update bit is set, you configured your axis to update the PositionError attribute.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>00</td> <td>position error update</td> </tr> <tr> <td>01</td> <td>position integrator error update</td> </tr> <tr> <td>02</td> <td>velocity error update</td> </tr> <tr> <td>03</td> <td>velocity integrator error update</td> </tr> <tr> <td>04</td> <td>velocity command update</td> </tr> <tr> <td>05</td> <td>velocity feedback update</td> </tr> <tr> <td>06</td> <td>servo output level update</td> </tr> </tbody> </table> <p>Bits 07 through 31 are reserved.</p> | Bit | Meaning | 00 | position error update | 01 | position integrator error update | 02 | velocity error update | 03 | velocity integrator error update | 04 | velocity command update | 05 | velocity feedback update | 06 | servo output level update | | | | | | | | | | | | | | | | |
| Bit | Meaning | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 | position error update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | position integrator error update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 02 | velocity error update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 03 | velocity integrator error update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 04 | velocity command update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 05 | velocity feedback update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 06 | servo output level update | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .WatchEvArmStatus | BOOL | You can use this bit to determine the status of the watch event. If this bit is set, the execution of a Motion Arm Watch (MAW) instruction has armed a watch event. This bit clears when a watch event occurs or the controller executes a Motion Disarm Watch (MDW) instruction. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .WatchEvStatus | BOOL | You can use this bit to determine the status of a watch event. If this bit is set, a watch event has occurred. This bit clears when the controller executes another Motion Arm Watch (MAW) instruction or a Motion Disarm Watch (MDW) instruction. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Understanding Servo Configuration Update Status Bits attributes

You can use the servo configuration update status bits attributes to monitor the progress of servo configuration attribute updates, which are initiated by an SSV instruction in your application program.

When the SSV instruction initiates an update, the controller sets the update status bit associated with the attribute. The update status bit remains set until the servo module indicates that the data update was successful.

For example, if you use an SSV instruction to change the PositionProportionalGain attribute of an axis and follow it with logic based on the completion of the SSV instruction, you can check for the resetting of the .PosPGainStatus bit to ensure that the servo module attribute is updated.

The following is a list of the servo configuration update status bits attributes.

| Variable | Data Type | Description |
|-------------------------|-----------|--|
| .AccFfGainStatus | BOOL | The status of an update to the <i>AccelerationFeedforwardGain</i> attribute. |
| .AxisTypeStatus | BOOL | The status of an update to the <i>AxisType</i> attribute. |
| .DriveFaultActStatus | BOOL | The status of an update to the <i>DriveFaultAction</i> attribute. |
| .EncLossFaultActStatus | BOOL | The status of an update to the <i>EncoderLossFaultAction</i> attribute. |
| .EncNsFaultActStatus | BOOL | The status of an update to the <i>EncoderNoiseFaultAction</i> attribute. |
| .FricCompStatus | BOOL | The status of an update to the <i>FrictionCompensation</i> attribute. |
| .MaxNTrvlStatus | BOOL | The status of an update to the <i>MaximumNegativeTravel</i> attribute. |
| .MaxPTrvlStatus | BOOL | The status of an update to the <i>MaximumPositiveTravel</i> attribute. |
| .OutFiltBWStatus | BOOL | The status of an update to the <i>OutputFilterBandwidth</i> attribute. |
| .OutLimitStatus | BOOL | The status of an update to the <i>OutputLimit</i> attribute. |
| .OutOffsetStatus | BOOL | The status of an update to the <i>OutputOffset</i> attribute. |
| .OutScaleStatus | BOOL | The status of an update to the <i>OutputScaling</i> attribute. |
| .PosErrorFaultActStatus | BOOL | The status of an update to the <i>PositionErrorFaultAction</i> attribute. |
| .PosErrorTolStatus | BOOL | The status of an update to the <i>PositionErrorTolerance</i> attribute. |
| .PosIGainStatus | BOOL | The status of an update to the <i>PositionIntegralGain</i> attribute. |
| .PosLockTolStatus | BOOL | The status of an update to the <i>PositionLockTolerance</i> attribute. |
| .PosPGainStatus | BOOL | The status of an update to the <i>PositionProportionalGain</i> attribute. |
| .PosUnwindStatus | BOOL | The status of an update to the <i>PositionUnwind</i> attribute. |
| .POtrvlFactActStatus | BOOL | The status of an update to the <i>SoftOvertravelFaultAction</i> attribute. |
| .VelFfGainStatus | BOOL | The status of an update to the <i>VelocityFeedforwardGain</i> attribute. |
| .VelIGainStatus | BOOL | The status of an update to the <i>VelocityIntegralGain</i> attribute. |
| .VelPGainStatus | BOOL | The status of an update to the <i>VelocityProportionalGain</i> attribute. |

Understanding the MOTION_GROUP Structure

The MOTION_GROUP structure contains status and configuration information for your motion group. You can directly access this information in your motion control program. For example, if you want to use the DriveFault attribute for Motion_Group, you would use Motion_Group.DriveFault to gain access to the attribute.

The bits in the MOTION_GROUP structure are set when any axis in the group experiences the conditions required to set the bit. For example, if one axis in a group of ten axes developed the conditions to set the .POtrvIFault bit, the controller would set the .POtrvIFault bit in the MOTION_GROUP structure.

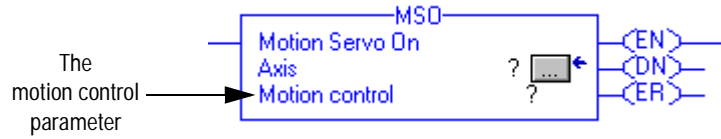
| Variable | Data Type | Description | | | | | | | | |
|--------------------|-----------|--|-------------|--------|-----------|-------------|-------------------|----|------|---------|
| .ACAsyncConnFault | BOOL | You can use this bit to determine the status of asynchronous communication. When the controller detects that one of the servo module parameters failed to update because of an asynchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. | | | | | | | | |
| .ACSyncConnFault | BOOL | You can use this bit to determine the status of synchronous communication. When the controller detects that the servo module has missed several position updates in a row because of a synchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. | | | | | | | | |
| .DriveFault | BOOL | You can use this bit to determine the status of the external drive. If this bit is set, the external drive detected a fault. This bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | |
| .EncCHALossFault | BOOL | You can use this bit to determine the status of the encoder channel A. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | |
| .EncCHBLossFault | BOOL | You can use this bit to determine the status of the encoder channel B. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | |
| .EncCHZLossFault | BOOL | You can use this bit to determine the status of the encoder channel Z. This bit sets if both of the differential signals are at the same level or if the servo module or encoder loses encoder power or common. The bit clears when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | |
| .EncNsFault | BOOL | You can use this bit to determine the status of encoder channels A and B. If the servo module detects simultaneous transitions of channels A and B, this bit sets. This bit is clear after the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | |
| .GroupFault | DINT | The fault bits for your motion group. <table border="1" data-bbox="576 1564 1299 1690"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>GroupOverlapFault</td> <td>00</td> <td>BOOL</td> <td>UNKNOWN</td> </tr> </tbody> </table> Bits 01 through 31 are reserved. | Bit | Number | Data type | Description | GroupOverlapFault | 00 | BOOL | UNKNOWN |
| Bit | Number | Data type | Description | | | | | | | |
| GroupOverlapFault | 00 | BOOL | UNKNOWN | | | | | | | |
| .GroupOverlapFault | BOOL | You can use this bit to determine the status of motion processing. This bit indicates that the motion processing in the controller does not have enough time to complete, and therefore requires an increased coarse update rate to function properly. If this bit is set, the controller requested motion processing, but it has not completed its previous two requests. You can reset this bit via direct access or by downloading to the controller. | | | | | | | | |

| Variable | Data Type | Description | | | | | | | | | | | | |
|-------------------|-----------|---|-------------------------------|--------|-----------|-------------|-------------------|----|------|-------------------------------|------------------|----|------|------------------------------|
| .GroupStatus | DINT | <p>The status bits for your motion group.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.InhibitStatus</td> <td>00</td> <td>BOOL</td> <td>UNKNOWN</td> </tr> <tr> <td>.GroupSynced</td> <td>01</td> <td>BOOL</td> <td>UNKNOWN</td> </tr> </tbody> </table> <p>Bits 02 through 31 are reserved.</p> | Bit | Number | Data type | Description | .InhibitStatus | 00 | BOOL | UNKNOWN | .GroupSynced | 01 | BOOL | UNKNOWN |
| Bit | Number | Data type | Description | | | | | | | | | | | |
| .InhibitStatus | 00 | BOOL | UNKNOWN | | | | | | | | | | | |
| .GroupSynced | 01 | BOOL | UNKNOWN | | | | | | | | | | | |
| .GroupSynced | BOOL | You can use this bit to determine the status of the group connection to the controller. This bit is set the first time all the axes in a group are connected and synchronized to the controller. This bit remains set until you download a new program, clear the controller memory, or powercycle the controller. | | | | | | | | | | | | |
| .Hardfault | BOOL | You can use this bit to determine the status of the servo module. If this bit is set, the servo module detected a hardware problem that typically requires the replacement of the servo module. | | | | | | | | | | | | |
| .InhibitStatus | BOOL | Not used by the controller. | | | | | | | | | | | | |
| .MotionFault | DINT | <p>The motion fault bits for your axis.</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.ACAsyncConnFault</td> <td>00</td> <td>BOOL</td> <td>asynchronous connection fault</td> </tr> <tr> <td>.ACSyncConnFault</td> <td>01</td> <td>BOOL</td> <td>synchronous connection fault</td> </tr> </tbody> </table> <p>Bits 02 through 31 are reserved.</p> | Bit | Number | Data type | Description | .ACAsyncConnFault | 00 | BOOL | asynchronous connection fault | .ACSyncConnFault | 01 | BOOL | synchronous connection fault |
| Bit | Number | Data type | Description | | | | | | | | | | | |
| .ACAsyncConnFault | 00 | BOOL | asynchronous connection fault | | | | | | | | | | | |
| .ACSyncConnFault | 01 | BOOL | synchronous connection fault | | | | | | | | | | | |
| .NOtrvlFault | BOOL | You can use this bit to determine the status of axis travel. If this bit is set, the axis has moved or has attempted to move beyond the MaximumNegativeOvertravel value. This bit is clear when the axis moves within the MaximumNegativeOvertravel values | | | | | | | | | | | | |
| .PosErrorFault | BOOL | You can use this bit to determine the status of axis position error. If this bit is set, the servo module has detected that axis position error exceeds the PositionErrorTolerance value. This bit is clear when the controller executes a Motion Axis Fault Reset (MAFR) instruction. | | | | | | | | | | | | |
| .POtrvlFault | BOOL | You can use this bit to determine the status of axis travel. If this bit is set, the axis has moved or has attempted to move beyond the MaximumPositiveOvertravel value. This bit is clear when the axis moves within the MaximumPositiveOvertravel values | | | | | | | | | | | | |

| Variable | Data Type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------|-----------|---|------------------------------|-----------|------------------------------|-------------|--------------|----|------|---------------------------|--------------|----|------|---------------------------|----------------|----|------|----------------------|------------------|----|------|------------------------------|------------------|----|------|------------------------------|------------------|----|------|------------------------------|-------------|----|------|---------------------|-------------|----|------|-------------|----------------|----|------|------------------------------|------------|----|------|----------------------|
| .ServoFault | DINT | The servo fault bits for your servo loop. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | <table border="1"> <thead> <tr> <th>Bit</th> <th>Number</th> <th>Data type</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>.POTrvlFault</td> <td>00</td> <td>BOOL</td> <td>positive overtravel fault</td> </tr> <tr> <td>.NOTrvlFault</td> <td>01</td> <td>BOOL</td> <td>negative overtravel fault</td> </tr> <tr> <td>.PosErrorFault</td> <td>02</td> <td>BOOL</td> <td>position error fault</td> </tr> <tr> <td>.EncCHALossFault</td> <td>03</td> <td>BOOL</td> <td>encoder channel A loss fault</td> </tr> <tr> <td>.EncCHBLossFault</td> <td>04</td> <td>BOOL</td> <td>encoder channel B loss fault</td> </tr> <tr> <td>.EncCHZLossFault</td> <td>05</td> <td>BOOL</td> <td>encoder channel Z loss fault</td> </tr> <tr> <td>.EncNsFault</td> <td>06</td> <td>BOOL</td> <td>encoder noise fault</td> </tr> <tr> <td>.DriveFault</td> <td>07</td> <td>BOOL</td> <td>drive fault</td> </tr> <tr> <td>.SyncConnFault</td> <td>08</td> <td>BOOL</td> <td>synchronous connection fault</td> </tr> <tr> <td>.Hardfault</td> <td>09</td> <td>BOOL</td> <td>servo hardware fault</td> </tr> </tbody> </table> | Bit | Number | Data type | Description | .POTrvlFault | 00 | BOOL | positive overtravel fault | .NOTrvlFault | 01 | BOOL | negative overtravel fault | .PosErrorFault | 02 | BOOL | position error fault | .EncCHALossFault | 03 | BOOL | encoder channel A loss fault | .EncCHBLossFault | 04 | BOOL | encoder channel B loss fault | .EncCHZLossFault | 05 | BOOL | encoder channel Z loss fault | .EncNsFault | 06 | BOOL | encoder noise fault | .DriveFault | 07 | BOOL | drive fault | .SyncConnFault | 08 | BOOL | synchronous connection fault | .Hardfault | 09 | BOOL | servo hardware fault |
| | | Bit | Number | Data type | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .POTrvlFault | 00 | BOOL | positive overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .NOTrvlFault | 01 | BOOL | negative overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .PosErrorFault | 02 | BOOL | position error fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .EncCHALossFault | 03 | BOOL | encoder channel A loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .EncCHBLossFault | 04 | BOOL | encoder channel B loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .EncCHZLossFault | 05 | BOOL | encoder channel Z loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .EncNsFault | 06 | BOOL | encoder noise fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | .DriveFault | 07 | BOOL | drive fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .SyncConnFault | 08 | BOOL | synchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .Hardfault | 09 | BOOL | servo hardware fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | Bits 10 through 31 are reserved. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| .SyncConnFault | BOOL | You can use this bit to determine the status of synchronous communication. When the servo module detects that it has missed several position updates in a row because of a synchronous communication failure, this bit sets. When you reestablish the connection, the bit clears. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Understanding the MOTION_INSTRUCTION Structure

The controller uses the MOTION_INSTRUCTION tag (structure) to store status information during the execution of motion instructions. Every motion instruction has a motion control parameter that requires a MOTION_INSTRUCTION tag for this purpose.



ATTENTION: Tags used for the motion control attribute of instructions should only be used once. Re-use of the motion control attribute in other instructions can cause unintended operation of the control variables.

The structure of the motion instruction tag is shown below:

MOTION_INSTRUCTION structure

| | | | | | | | | | | | | |
|------------|-----------------------------|----|----|----|----|----|-----------------------------------|--|----|-----------------------------------|--|--|
| bit number | 31 | 30 | 29 | 28 | 27 | 26 | 16 | | 15 | 0 | | |
| | EN | DN | ER | IP | PC | | | | | | | |
| | error code (.ERR) (16 bits) | | | | | | message status (.STATUS) (8 bits) | | | execution state (.STATE) (8 bits) | | |

| Mnemonic | Data Type | Description |
|----------|-----------|---|
| .EN | BOOL | The enable bit indicates that the instruction is enabled. |
| .DN | BOOL | The done bit indicates that the operation is complete. |
| .ER | BOOL | The error bit indicates when the operation generates an error. |
| .IP | BOOL | The in process bit indicates that a process is being executed. |
| .PC | BOOL | The process complete bit indicates that the operation is complete. Note: The .DN bit sets after an instruction has completed execution. The .PC bit sets when the initiated process has completed. |
| .ERR | DINT | The error value contains the error code associated with a motion function. |
| .STATUS | DINT | The message status value indicates the status condition of any message associated with the motion function. |
| .STATE | DINT | The execution status value indicates the execution state of a function. Many motion functions have several steps and this value tracks these steps. |

Understanding Error Codes (.ERR)

| Error Code | Description |
|------------|---|
| 3 | The instruction tried to execute while another instance of this instruction was executing. This can occur when the controller executes a messaging instruction without checking the .DN bit of the preceding instruction. |
| 4 | The instruction tried to execute on an axis with a closed servo loop. |
| 5 | The instruction tried to execute on an axis with a servo loop that is not closed. |
| 6 | The axis drive is enabled. |
| 7 | The axis is in the shutdown state. |
| 8 | The axis is not configured as a servo axis type. |
| 9 | The instruction tried to execute in a direction that aggravates the current overtravel condition. |
| 10 | The master axis reference is the same as the slave axis reference. |
| 11 | The axis is not configured. |
| 12 | Messaging to the servo module failed. |
| 13 | The instruction tried to use a parameter that is outside the range limit. |
| 14 | The instruction cannot apply the tuning parameters because of an error in the run tuning instruction. |
| 15 | The instruction cannot apply the diagnostic parameters because of an error in the run diagnostic test instruction. |
| 16 | The instruction tried to execute with homing in progress. |
| 17 | The instruction tried to execute a rotary move on an axis that is not configured for rotary operation. |
| 18 | The axis type is configured as unused. |
| 19 | The motion group is not in the synchronized state. This could be caused by a missing servo module or a misconfiguration. |
| 20 | The axis is in the faulted state. |
| 21 | The group is in the faulted state. |
| 22 | An MSO (Motion Servo On) or MAH (Motion Axis Home) instruction was attempted while the axis was in motion. |
| 23 | An instruction attempted an illegal change of dynamics, such as a merging on an S-curve or changing the acceleration of an S-curve. |

Understanding Message Status (.STATUS)

| Message Status | Description |
|----------------|---|
| 0x0 | The message was successful. |
| 0x1 | The module is processing another message. |
| 0x2 | The module is waiting for a response to a previous message. |
| 0x3 | The response to a message failed. |
| 0x4 | The module is not ready for messaging. |

Understanding Execution Status (.STATE)

The execution status is always set to 0 when the controller sets the .EN bit for a motion instruction. Other execution states depend on the motion instruction.

The Motion Attributes

This appendix describes the motion attributes, their data types, and their access rules.

The Logix5550 controller stores motion status and configuration information in the AXIS and MOTION_GROUP objects. To directly access this information, you can select the object (AXIS or MOTION_GROUP) and select the attribute. You can also use the GSV and SSV instructions to access these objects. See *Input/Output Instructions* in the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1 for more information about the GSV and SSV instructions.

Motion Instance Variables

To use the motion instance variables, choose AXIS from the object list of the GSV and SSV instructions.

When an attribute is marked with an asterisk (*), it means that the attribute is located in both the ControlLogix controller and in the motion module. When you use an SSV instruction to write one of these values, the controller will automatically update the copy in the module. However, this process is not immediate. To be sure that the new value has been updated in the module, use an interlock mechanism using the boolean bits in the Servo Configuration Update Status Bits of the AXIS structure.

For example, if you perform an SSV instruction on the PositionLockTolerance, the PositionLockTolStatus of the Axis tag will be set until an update to the module is successful. Therefore, the logic following the SSV could wait on this bit resetting before continuing in the program.

| Variable | Data Type | Access | Description |
|-------------------------------|-----------|------------|--|
| * AccelerationFeedforwardGain | REAL | GSV SSV | The value used to provide the torque command output to generate the command acceleration. |
| ActualPosition | REAL | GSV | The actual position of your axis. |
| ActualVelocity | REAL | GSV | The actual velocity of your axis. The internal resolution limit of the actual velocity is 1 encoder count per coarse update. |
| AverageVelocity | REAL | GSV | The average velocity of your axis. |
| AverageVelocityTimebase | REAL | GSV SSV | The timebase of the average velocity of your axis. |
| AxisConfigurationState | SINT | GSV | The state of the axis configuration. |
| * AxisType | INT | GSV SSV | The type of axis that you are using. Value: Meaning: 0 unused axis 1 position-only axis 2 servo axis |
| CommandPosition | REAL | GSV | The command position of your axis. |
| CommandVelocity | REAL | GSV | The command velocity of your axis. The internal resolution limit on the command velocity is 0.00001 encoder counts per coarse update. |
| ConversionConstant | REAL | GSV SSV | The conversion factor used to convert from your units to feedback counts. |
| DampingFactor | REAL | GSV SSV | The value used in calculating the maximum position servo bandwidth during the execution of the Motion Run Axis Tuning (MRAT) instruction. |
| * DriveFaultAction | SINT | GSV SSV | The operation performed when a drive fault occurs. Value: Meaning: 0 shutdown the axis 1 disable the drive 2 stop the commanded motion 3 change the status bit only |
| EffectivInertia | REAL | GSV | The inertia value for the axis as calculated from the measurements the controller made during the last Motion Run Axis Tuning (MRAT) instruction. |
| * EncoderLossFaultAction | SINT | GSV SSV | The operation performed when an encoder loss fault occurs. Value: Meaning: 0 shutdown the axis 1 disable the drive 2 stop the commanded motion 3 change the status bit only |

| Variable | Data Type | Access | Description |
|---------------------------|-----------|------------|--|
| * EncoderNoiseFaultAction | SINT | GSV SSV | The operation performed when an encoder noise fault occurs. Value: 0 1 2 3 Meaning: shutdown the axis disable the drive stop the commanded motion change the status bit only |
| * FrictionCompensation | REAL | GSV SSV | The fixed output level used to compensate for static friction. |
| GroupInstance | DINT | GSV | The instance number of the motion group that contains your axis. |
| HomeMode | SINT | GSV SSV | The homing mode for your axis. Value: 0 1 Meaning: passive homing active homing (default) |
| HomePosition | REAL | GSV SSV | The homing position of your axis. |
| HomeReturnSpeed | REAL | GSV SSV | The homing return speed of your axis. |
| HomeSequenceType | SINT | GSV SSV | The homing sequence type for your axis. Value: 0 1 2 3 Meaning: immediate homing switch homing marker homing switch-marker homing (default) |
| HomeSpeed | REAL | GSV SSV | The homing speed of your axis. |
| INSTANCE | DINT | GSV | The instance number of the axis. |
| MapTableInstance | DINT | GSV | The I/O map instance of the servo module. This attribute can only be set if you did not assign the axis to a group or if you assigned it to a group in the group inhibit mode. |
| MaximumAcceleration | REAL | GSV SSV | The maximum acceleration of your axis. The controller automatically sets the maximum acceleration value to approximately 85% of the tuning acceleration determined by the Motion Apply Axis Tune (MAAT) instruction. |
| MaximumDeceleration | REAL | GSV SSV | The maximum deceleration of your axis. The controller automatically sets the maximum deceleration value to approximately 85% of the tuning deceleration determined by the Motion Apply Axis Tune (MAAT) instruction. |
| * MaximumNegativeTravel | REAL | GSV SSV | The maximum negative travel limit. This value is always less than the MaximumPositiveTravel value. |
| * MaximumPositiveTravel | REAL | GSV SSV | The maximum positive travel limit. This value is always greater than the MaximumNegativeTravel value. |

| Variable | Data Type | Access | Description |
|---------------------------|-----------|-------------------|--|
| MaximumSpeed | REAL | GSV SSV | The maximum speed of your axis. The controller automatically sets the maximum speed value to the tuning speed determined by the Motion Apply Axis Tune (MAAT) instruction. |
| ModuleChannel | SINT | GSV | The module channel of your servo module. This attribute can only be set if you did not assign the axis to a group or if you assigned it to a group in the group inhibit mode. |
| MotionConfigurationBits | DINT | GSV SSV | The motion configuration bits for your axis. Bit: Meaning: 0 home direction reverse 1 home switch normally closed 2 home marker edge negative |
| MotionFaultBits | DINT | AXIS structure | The motion fault bits for your axis. Bit: Bit Name: Meaning: 0 ACAsyncConnFault asynchronous connection fault 1 ACSyncConnFault synchronous connection fault |
| MotionStatusBits | DINT | AXIS structure | The motion status bits for your axis. Bit: Bit Name: Meaning: 0 AccelStatus acceleration 1 DecelStatus deceleration 2 MoveStatus move 3 JogStatus jog 4 GearingStatus gear 5 HomingStatus homing 6 ClutchStatus clutch 7 AxisHomedStatus homed status |
| MotorEncoderTestIncrement | REAL | GSV SSV | The amount of motion that is necessary to initiate the Motion Run Hookup Diagnostic (MRHD) test. |
| * OutputFilterBandwidth | REAL | GSV SSV | The bandwidth of the servo low-pass digital output filter. |
| * OutputLimit | REAL | GSV SSV | The value of the maximum servo output voltage of your axis. |
| * OutputOffset | REAL | GSV SSV | The value used to offset the effects of the cumulative offsets of the servo module DAC output and the servo drive input. |
| * OutputScaling | REAL | GSV SSV | The value used to convert the output of the servo loop into the equivalent voltage to the drive. For a velocity servo drive, the output scaling is: $\frac{10Volts}{Speedat10Volts} \times ConversionConstant$ For a torque servo drive, the output scaling is: $\frac{10Volts}{Accelerationat10Volts} \times ConversionConstant$ |
| PositionError | REAL | GSV | The difference between the actual and command position of an axis. You can use this value to drive the motor to where the actual position equals the command position. |

| Variable | Data Type | Access | Description |
|----------------------------|-----------|------------|--|
| * PositionErrorFaultAction | SINT | GSV SSV | The operation performed when a position error fault occurs. Value: 0 1 2 3 Meaning: shutdown the axis disable the drive stop the commanded motion change the status bit only |
| * PositionErrorTolerance | REAL | GSV SSV | The amount of position error that the servo tolerates before issuing a position error fault. |
| * PositionIntegralGain | REAL | GSV SSV | The value used to achieve accurate axis positioning despite disturbances such as static friction and gravity. |
| PositionIntegratorError | REAL | GSV | The sum of the position error for an axis. You can use this value to drive the motor to where the actual position equals the command position. |
| PositionLockTolerance | REAL | GSV SSV | The amount of position error that the servo module tolerates when giving a true position locked status indication. |
| * PositionProportionalGain | REAL | GSV SSV | The value the controller multiplies with the position error to correct for the position error. |
| PositionServoBandwidth | REAL | GSV SSV | The unity gain bandwidth that the controller uses to calculate the gains for a Motion Apply Axis Tuning (MAAT) instruction. |
| * PositionUnwind | DINT | GSV SSV | The value used to perform the automatic unwind of the rotary axis. |
| ProgrammedStopMode | SINT | GSV SSV | The type of stop to perform on your axis. Value: 0 1 2 Meaning: fast stop fast shutdown hard shutdown |
| RegistrationPosition | REAL | GSV | The registration position for your axis. You can use the following equation to determine the maximum registration position error based on your axis speed: $\text{MaximumSpeed} \left(\frac{\text{PositionUnits}}{\text{Seconds}} \right) = \frac{\text{Accuracy}(\text{PositionUnits})}{0.000001 \text{ Seconds}}$ |
| * ServoConfigurationBits | DINT | GSV SSV | The servo configuration bits for your servo loop. Bit: 0 1 2 3 4 5 6 7 8 9 Meaning: rotary axis external velocity servo drive encoder polarity negative servo polarity negative soft overtravel checking position error checking encoder loss fault checking encoder noise fault checking drive fault checking drive fault normally closed |

| Variable | Data Type | Access | Description | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|------------------------------|------------------------|-------------------------------|---|------|-----------|----------|---|------------------|---------------------------|---|----------------|---------------------------|---|----------------|--------------------------|---|-----------------|------------------------------|---|-------------------|------------------------------|---|------------------|------------------------------|---|----------------|----------------------------|---|----------------|------------------------|---|-----------------|------------------------------|---|-----------------|-------------------------------|----|----------------|----------------------------|----|----------------|------------------------|----|-----------------|-------------------------|----|----------------|----------------|----|----------------|--------------|----|-----------------|---------------|----|----------------|-----------------------|----|----------------------|------------------------------|----|------------------------|-----------------------------|----|-----------------------|---------------------------|----|---------------------|----------------------------|----|---------------------|--------------------|
| ServoConfigurationUpdateBits | DINT | AXIS structure | <p>The servo configuration status bits for your servo loop.</p> <table border="1"> <thead> <tr> <th>Bit:</th> <th>Bit Name:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr><td>0</td><td>AxisTypeStatus</td><td>axis type</td></tr> <tr><td>1</td><td>PosUnwndStatus</td><td>position unwind</td></tr> <tr><td>2</td><td>MaxPTrvlStatus</td><td>maximum positive travel</td></tr> <tr><td>3</td><td>MaxNTrvlStatus</td><td>maximum negative travel</td></tr> <tr><td>4</td><td>PosErrorTolStatus</td><td>position error tolerance</td></tr> <tr><td>5</td><td>PosLockTolStatus</td><td>position lock tolerance</td></tr> <tr><td>6</td><td>PosPGainStatus</td><td>position proportional gain</td></tr> <tr><td>7</td><td>PosIGainStatus</td><td>position integral gain</td></tr> <tr><td>8</td><td>VelFFGainStatus</td><td>velocity feedforward gain</td></tr> <tr><td>9</td><td>AccFFGainStatus</td><td>acceleration feedforward gain</td></tr> <tr><td>10</td><td>VelPGainStatus</td><td>velocity proportional gain</td></tr> <tr><td>11</td><td>VelIGainStatus</td><td>velocity integral gain</td></tr> <tr><td>12</td><td>OutFiltBwStatus</td><td>output filter bandwidth</td></tr> <tr><td>13</td><td>OutScaleStatus</td><td>output scaling</td></tr> <tr><td>14</td><td>OutLimitStatus</td><td>output limit</td></tr> <tr><td>15</td><td>OutOffsetStatus</td><td>output offset</td></tr> <tr><td>16</td><td>FricCompStatus</td><td>friction compensation</td></tr> <tr><td>17</td><td>POtrvlFaultActStatus</td><td>soft overtravel fault action</td></tr> <tr><td>18</td><td>PosErrorFaultActStatus</td><td>position error fault action</td></tr> <tr><td>19</td><td>EncLossFaultActStatus</td><td>encoder loss fault action</td></tr> <tr><td>20</td><td>EncNsFaultActStatus</td><td>encoder noise fault action</td></tr> <tr><td>21</td><td>DriveFaultActStatus</td><td>drive fault action</td></tr> </tbody> </table> | Bit: | Bit Name: | Meaning: | 0 | AxisTypeStatus | axis type | 1 | PosUnwndStatus | position unwind | 2 | MaxPTrvlStatus | maximum positive travel | 3 | MaxNTrvlStatus | maximum negative travel | 4 | PosErrorTolStatus | position error tolerance | 5 | PosLockTolStatus | position lock tolerance | 6 | PosPGainStatus | position proportional gain | 7 | PosIGainStatus | position integral gain | 8 | VelFFGainStatus | velocity feedforward gain | 9 | AccFFGainStatus | acceleration feedforward gain | 10 | VelPGainStatus | velocity proportional gain | 11 | VelIGainStatus | velocity integral gain | 12 | OutFiltBwStatus | output filter bandwidth | 13 | OutScaleStatus | output scaling | 14 | OutLimitStatus | output limit | 15 | OutOffsetStatus | output offset | 16 | FricCompStatus | friction compensation | 17 | POtrvlFaultActStatus | soft overtravel fault action | 18 | PosErrorFaultActStatus | position error fault action | 19 | EncLossFaultActStatus | encoder loss fault action | 20 | EncNsFaultActStatus | encoder noise fault action | 21 | DriveFaultActStatus | drive fault action |
| Bit: | Bit Name: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | AxisTypeStatus | axis type | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | PosUnwndStatus | position unwind | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | MaxPTrvlStatus | maximum positive travel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | MaxNTrvlStatus | maximum negative travel | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | PosErrorTolStatus | position error tolerance | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | PosLockTolStatus | position lock tolerance | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | PosPGainStatus | position proportional gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | PosIGainStatus | position integral gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | VelFFGainStatus | velocity feedforward gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | AccFFGainStatus | acceleration feedforward gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | VelPGainStatus | velocity proportional gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | VelIGainStatus | velocity integral gain | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | OutFiltBwStatus | output filter bandwidth | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | OutScaleStatus | output scaling | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | OutLimitStatus | output limit | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | OutOffsetStatus | output offset | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | FricCompStatus | friction compensation | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | POtrvlFaultActStatus | soft overtravel fault action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 18 | PosErrorFaultActStatus | position error fault action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 19 | EncLossFaultActStatus | encoder loss fault action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 20 | EncNsFaultActStatus | encoder noise fault action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 21 | DriveFaultActStatus | drive fault action | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ServoEventBits | DINT | AXIS structure | <p>The servo event bits for your servo loop.</p> <table border="1"> <thead> <tr> <th>Bit:</th> <th>Bit Name:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr><td>0</td><td>WatchEvArmStatus</td><td>watch event armed</td></tr> <tr><td>1</td><td>WatchEvStatus</td><td>watch event</td></tr> <tr><td>2</td><td>RegEvArmStatus</td><td>registration event armed</td></tr> <tr><td>3</td><td>RegEvStatus</td><td>registration event</td></tr> <tr><td>4</td><td>HomeEvArmStatus</td><td>home event armed</td></tr> <tr><td>5</td><td>HomeEvStatus</td><td>home event</td></tr> </tbody> </table> | Bit: | Bit Name: | Meaning: | 0 | WatchEvArmStatus | watch event armed | 1 | WatchEvStatus | watch event | 2 | RegEvArmStatus | registration event armed | 3 | RegEvStatus | registration event | 4 | HomeEvArmStatus | home event armed | 5 | HomeEvStatus | home event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit: | Bit Name: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | WatchEvArmStatus | watch event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | WatchEvStatus | watch event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | RegEvArmStatus | registration event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | RegEvStatus | registration event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | HomeEvArmStatus | home event armed | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | HomeEvStatus | home event | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ServoFaultBits | DINT | AXIS structure | <p>The servo fault bits for your servo loop.</p> <table border="1"> <thead> <tr> <th>Bit:</th> <th>Bit Name:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr><td>0</td><td>POtrvlFault</td><td>positive overtravel fault</td></tr> <tr><td>1</td><td>NOtrvlFault</td><td>negative overtravel fault</td></tr> <tr><td>2</td><td>PosErrorFault</td><td>position error fault</td></tr> <tr><td>3</td><td>EncCHALossFault</td><td>encoder channel A loss fault</td></tr> <tr><td>4</td><td>EncCHBLossFault</td><td>encoder channel B loss fault</td></tr> <tr><td>5</td><td>EncCHZLossFault</td><td>encoder channel Z loss fault</td></tr> <tr><td>6</td><td>EncNsFault</td><td>encoder noise fault</td></tr> <tr><td>7</td><td>DriveFault</td><td>drive fault</td></tr> <tr><td>8</td><td>SyncConnFault</td><td>synchronous connection fault</td></tr> <tr><td>9</td><td>HardFault</td><td>servo hardware fault</td></tr> </tbody> </table> | Bit: | Bit Name: | Meaning: | 0 | POtrvlFault | positive overtravel fault | 1 | NOtrvlFault | negative overtravel fault | 2 | PosErrorFault | position error fault | 3 | EncCHALossFault | encoder channel A loss fault | 4 | EncCHBLossFault | encoder channel B loss fault | 5 | EncCHZLossFault | encoder channel Z loss fault | 6 | EncNsFault | encoder noise fault | 7 | DriveFault | drive fault | 8 | SyncConnFault | synchronous connection fault | 9 | HardFault | servo hardware fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| Bit: | Bit Name: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | POtrvlFault | positive overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | NOtrvlFault | negative overtravel fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | PosErrorFault | position error fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | EncCHALossFault | encoder channel A loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | EncCHBLossFault | encoder channel B loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | EncCHZLossFault | encoder channel Z loss fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | EncNsFault | encoder noise fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | DriveFault | drive fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | SyncConnFault | synchronous connection fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | HardFault | servo hardware fault | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| ServoOutputLevel | REAL | GSV | The output voltage level for your axis servo loop. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Variable | Data Type | Access | Description | | | | | | | | | | | | | | | | | | | | | | | | |
|-----------------------------|--|-------------------|---|--------|-----------|----------|------------------------------|----------------|----------------------------------|---|----------------------------------|--------------|----------------------------------|--------------|--|---|-----------------------------|---------------|---------------------------|------------|----------------|----|------------|-----------------|----|----------------|---------------|
| ServoStatusBits | DINT | AXIS structure | <p>The status bits for your servo loop.</p> <table border="0"> <thead> <tr> <th>Bit:</th> <th>Bit Name:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>ServoActStatus</td> <td>servo action</td> </tr> <tr> <td>1</td> <td>DriveEnableStatus</td> <td>drive enable</td> </tr> <tr> <td>2</td> <td>OutLmtStatus</td> <td>output limit</td> </tr> <tr> <td>3</td> <td>PosLockStatus</td> <td>position lock</td> </tr> <tr> <td>13</td> <td>TuneStatus</td> <td>tuning process</td> </tr> <tr> <td>14</td> <td>TestStatus</td> <td>test diagnostic</td> </tr> <tr> <td>15</td> <td>ShutdownStatus</td> <td>axis shutdown</td> </tr> </tbody> </table> | Bit: | Bit Name: | Meaning: | 0 | ServoActStatus | servo action | 1 | DriveEnableStatus | drive enable | 2 | OutLmtStatus | output limit | 3 | PosLockStatus | position lock | 13 | TuneStatus | tuning process | 14 | TestStatus | test diagnostic | 15 | ShutdownStatus | axis shutdown |
| Bit: | Bit Name: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | ServoActStatus | servo action | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | DriveEnableStatus | drive enable | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | OutLmtStatus | output limit | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | PosLockStatus | position lock | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | TuneStatus | tuning process | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | TestStatus | test diagnostic | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | ShutdownStatus | axis shutdown | | | | | | | | | | | | | | | | | | | | | | | | | |
| ServoStatusUpdateBits | DINT | GSV SSV | <p>The servo status update bits for your axis.</p> <table border="0"> <thead> <tr> <th>Bit:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>position error update</td> </tr> <tr> <td>1</td> <td>position integrator error update</td> </tr> <tr> <td>2</td> <td>velocity error update</td> </tr> <tr> <td>3</td> <td>velocity integrator error update</td> </tr> <tr> <td>4</td> <td>velocity command update</td> </tr> <tr> <td>5</td> <td>velocity feedback update</td> </tr> <tr> <td>6</td> <td>servo output level update</td> </tr> </tbody> </table> | Bit: | Meaning: | 0 | position error update | 1 | position integrator error update | 2 | velocity error update | 3 | velocity integrator error update | 4 | velocity command update | 5 | velocity feedback update | 6 | servo output level update | | | | | | | | |
| Bit: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | position error update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | position integrator error update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | velocity error update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | velocity integrator error update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | velocity command update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | velocity feedback update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | servo output level update | | | | | | | | | | | | | | | | | | | | | | | | | | |
| * SoftOvertravelFaultAction | SINT | GSV SSV | <p>The operation performed when a soft overtravel fault occurs.</p> <table border="0"> <thead> <tr> <th>Value:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>shutdown the axis</td> </tr> <tr> <td>1</td> <td>disable the drive</td> </tr> <tr> <td>2</td> <td>stop the commanded motion</td> </tr> <tr> <td>3</td> <td>change the status bit only</td> </tr> </tbody> </table> | Value: | Meaning: | 0 | shutdown the axis | 1 | disable the drive | 2 | stop the commanded motion | 3 | change the status bit only | | | | | | | | | | | | | | |
| Value: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | shutdown the axis | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | disable the drive | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | stop the commanded motion | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | change the status bit only | | | | | | | | | | | | | | | | | | | | | | | | | | |
| StartActualPosition | REAL | GSV | <p>The actual position of your axis when new commanded motion starts for the axis.</p> <p>You can use this value to correct for any motion occurring between the detection of an event and the action initiated by the event.</p> | | | | | | | | | | | | | | | | | | | | | | | | |
| StartCommandPosition | REAL | GSV | <p>The command position of your axis when new commanded motion starts for the axis.</p> <p>You can use this value to correct for any motion occurring between the detection of an event and the action initiated by the event.</p> | | | | | | | | | | | | | | | | | | | | | | | | |
| StrobeActualPosition | REAL | GSV | <p>The actual position of an axis when the Motion Group Strobe Position (MGSP) instruction executes.</p> | | | | | | | | | | | | | | | | | | | | | | | | |
| StrobeCommandPosition | REAL | GSV | <p>The command position of an axis when the Motion Group Strobe Position (MGSP) instruction executes.</p> | | | | | | | | | | | | | | | | | | | | | | | | |
| TestDirectionForward | BOOL | GSV | <p>The direction of axis travel during the Motion Run Hookup Diagnostic (MRHD) instruction as seen by the servo module.</p> <table border="0"> <thead> <tr> <th>Value:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>negative (reverse) direction</td> </tr> <tr> <td>1</td> <td>positive (forward) direction</td> </tr> </tbody> </table> | Value: | Meaning: | 0 | negative (reverse) direction | 1 | positive (forward) direction | | | | | | | | | | | | | | | | | | |
| Value: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | negative (reverse) direction | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | positive (forward) direction | | | | | | | | | | | | | | | | | | | | | | | | | | |
| TestStatus | UINT16 | GSV | <p>The status of the last Motion Run Hookup Diagnostic (MRHD) instruction.</p> <table border="0"> <thead> <tr> <th>Value:</th> <th>Meaning:</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>test process successful</td> </tr> <tr> <td>1</td> <td>test in progress</td> </tr> <tr> <td>2</td> <td>test process aborted by the user</td> </tr> <tr> <td>3</td> <td>test exceeded 2-second time-out</td> </tr> <tr> <td>4</td> <td>test process failed due to servo fault</td> </tr> <tr> <td>5</td> <td>insufficient test increment</td> </tr> </tbody> </table> | Value: | Meaning: | 0 | test process successful | 1 | test in progress | 2 | test process aborted by the user | 3 | test exceeded 2-second time-out | 4 | test process failed due to servo fault | 5 | insufficient test increment | | | | | | | | | | |
| Value: | Meaning: | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 0 | test process successful | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | test in progress | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | test process aborted by the user | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | test exceeded 2-second time-out | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | test process failed due to servo fault | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | insufficient test increment | | | | | | | | | | | | | | | | | | | | | | | | | | |

| Variable | Data Type | Access | Description |
|-------------------------|-----------|------------|--|
| TuneAcceleration | REAL | GSV | The acceleration value measured during the last Motion Run Axis Tuning (MRAT) instruction. |
| TuneAccelerationTime | REAL | GSV | The acceleration time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction. |
| TuneDeceleration | REAL | GSV | The deceleration value measured during the last Motion Run Axis Tuning (MRAT) instruction. |
| TuneDecelerationTime | REAL | GSV | The deceleration time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction. |
| TuneRiseTime | REAL | GSV | The axis rise time in seconds measured during the last Motion Run Axis Tuning (MRAT) instruction. This value only applies to axes that you configure to work with an external velocity servo drive. |
| TuneSpeedScaling | REAL | GSV | The axis drive scaling factor measured during the last Motion Run Axis Tuning (MRAT) instruction. This value only applies to axes that you configure to work with an external velocity servo drive. |
| TuneStatus | UINT16 | GSV | The status of the last Motion Run Axis Tuning (MRAT) instruction. Value: 0 1 2 3 4 5 6 7 Meaning: tune process successful tuning in progress tune process aborted by user tune exceeded 2-second time-out tune process failed due to servo fault axis reached tuning travel limit axis polarity set incorrectly tune speed is too small to make measurements |
| TuneVelocityBandwidth | REAL | GSV | The bandwidth of the drive as calculated from the measurements made during the last Motion Run Axis Tuning (MRAT) instruction. |
| TuningConfigurationBits | DINT | GSV SSV | The tuning configuration bits for your axis. Bit: 0 1 2 3 4 5 Meaning: tuning direction (0=forward, 1=reverse) tune position error integrator tune velocity error integrator tune velocity feedforward acceleration feedforward tune velocity low-pass filter |
| TuningSpeed | REAL | GSV SSV | The maximum speed reached by the Motion Run Axis Tuning (MRAT) instruction. |
| TuningTravelLimit | REAL | GSV SSV | The travel limit used by the Motion Run Axis Tuning (MRAT) instruction to limit the action of the axis during tuning. |
| VelocityCommand | REAL | GSV | The current velocity reference to the velocity servo loop for an axis. |
| VelocityError | REAL | GSV | The difference between the commanded and actual velocity of a servo axis. You can use this value to drive the motor to where the velocity feedback equals the velocity command. |

| Variable | Data Type | Access | Description |
|----------------------------|-----------|------------|---|
| VelocityFeedback | REAL | GSV | The actual velocity of your axis as estimated by the servo module. To estimate the velocity, the servo module applies a 1 kHz low-pass filter to the change in actual position in one update interval. |
| * VelocityFeedforwardGain | REAL | GSV SSV | The value used to provide the velocity command output to generate the command velocity. |
| * VelocityIntegralGain | REAL | GSV SSV | The value that the controller multiplies with the VelocityIntegratorError value to correct the velocity error. |
| VelocityIntegratorError | REAL | GSV | The sum of the velocity error for a specified axis. You can use this value to drive the motor to where the velocity feedback equals the velocity command. |
| * VelocityProportionalGain | REAL | GSV SSV | The value that the controller multiplies with the VelocityError to correct the velocity error. |
| WatchPosition | REAL | GSV | The watch position of your axis. |

Instruction Timing

This appendix describes motion instruction timing types. The following table shows the contents of this appendix:

| For information about | See page |
|---|----------|
| Understanding Immediate Type Instructions | E-1 |
| Understanding Message Type Instructions | E-3 |
| Understanding Process Type Instructions | E-5 |

Motion instructions use three types of timing sequences

| Timing type | Description |
|-------------|--|
| Immediate | The instruction completes in one scan. |
| Message | The instruction completes over several scans because the instruction sends messages to the servo module. |
| Process | The instruction could take an indefinite amount of time to complete. |

Understanding Immediate Type Instructions

Immediate type motion instructions execute to completion in one scan. If the controller detects an error during the execution of these instructions, the error status bit sets and the operation ends.

Examples of immediate type instructions include the:

- Motion Change Dynamics (MCD) instruction
- Motion Group Strobe Position (MGSP) instruction

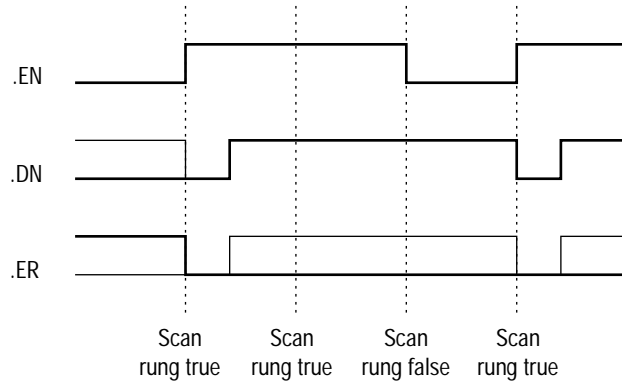
Immediate instructions work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
 - Sets the enable (EN) bit.
 - Clears the done (DN) bit.
 - Clears the error (ER) bit.
2. The controller executes the instruction completely.

3.

| If the controller | Then |
|--|--|
| Does not detect an error when the instruction executes | The controller sets the .DN bit. |
| Detects an error when the instruction executes | The controller sets the .ER bit and stores an error code in the control structure. |

4. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.
5. The controller can execute the instruction again when the rung becomes true.



Understanding Message Type Instructions

Message type motion instructions send one or more messages to the servo module.

Examples of message type instructions include the:

- Motion Direct Drive On (MDO) instruction
- Motion Redefine Position (MRP) instruction

Message type instructions work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
 - Sets the enable (EN) bit.
 - Clears the done (DN) bit.
 - Clears the error (ER) bit.

2. The controller begins to execute the instruction by setting up a message request to the servo module.

Note: The remainder of the instruction executes in parallel to the program scan.

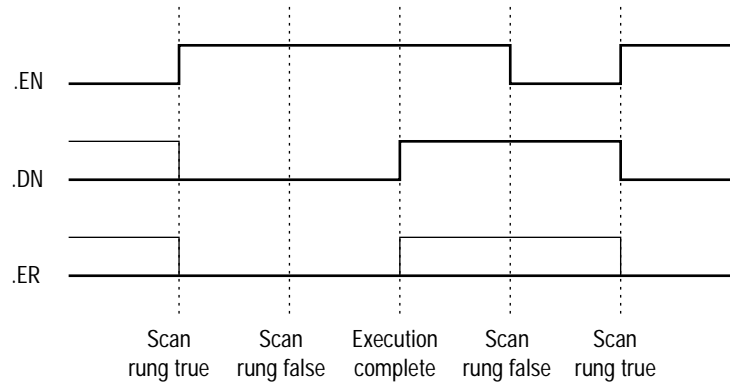
3. The controller checks if the servo module is ready to receive a new message.
4. The controller places the results of the check in the message status word of the control structure.
5. When the module is ready, the controller constructs and transmits the message to the module.

Note: This process may repeat several times if the instruction requires multiple messages.

- 6.

| If the controller | Then |
|--|--|
| Does not detect an error when the instruction executes | The controller sets the .DN bit. |
| Detects an error when the instruction executes | The controller sets the .ER bit and stores an error code in the control structure. |

7. The next time the rung becomes false after either the .DN or .ER bit sets, the controller clears the .EN bit.
8. When the rung becomes true, the controller can execute the instruction again.



Understanding Process Type Instructions

Process type motion instructions initiate motion processes that can take an indefinite amount of time to complete.

Examples of process type instructions include the:

- Motion Arm Watch Position (MAW) instruction
- Motion Axis Move (MAM) instruction

Process type instructions work as follows:

1. When the rung that contains the motion instruction becomes true, the controller:
 - Sets the enable (.EN) bit.
 - Clears the done (.DN) bit.
 - Clears the error (.ER) bit.
 - Clears the process complete (.PC) bit.
2. The controller initiates the motion process.

3.

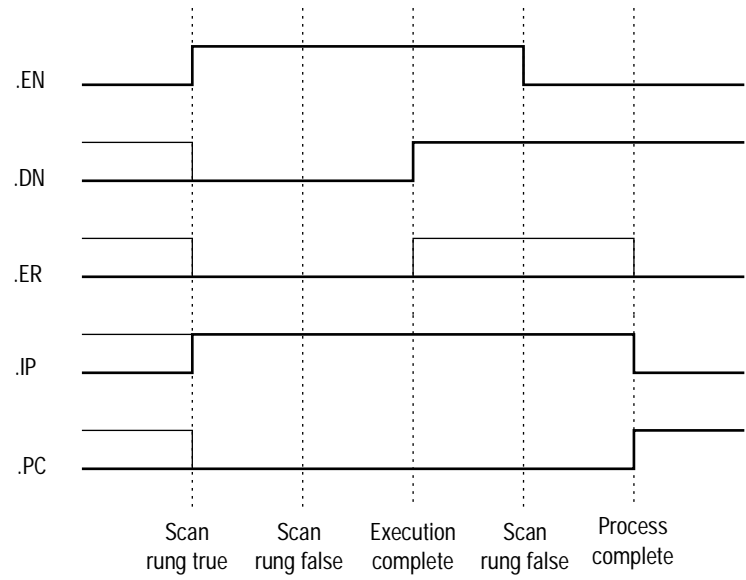
| If | Then the controller |
|--|--|
| The controller does not detect an error when the instruction executes | <ul style="list-style-type: none"> • Sets the .DN bit. • Sets the in process (.IP) bit. |
| The controller detects an error when the instruction executes | <ul style="list-style-type: none"> • Sets the .ER bit. • Stores an error code in the control structure. • Does not change the .IP and .PC bits. |
| The controller detects another instance of the motion instruction | Clears the .IP bit for that instance. |
| The motion process reaches the point where the instruction can be executed again | Sets the .DN bit. Note: For some process type instructions, like MAM, this will occur on the first scan. For others, like MAH, the .DN bit will not be set until the entire homing process is complete. |
| One of the following occurs during the motion process: <ul style="list-style-type: none"> • The motion process completes • Another instance of the instruction executes • Another instruction stops the motion process • A motion fault stops the motion process | <ul style="list-style-type: none"> • Sets the .DN bit. • Sets the .PC bit. • Clears the .IP bit. |

4. Once the initiation of the motion process completes, the program scan can continue.

Note: The remainder of the instruction and the control process continue in parallel with the program scan.

5. The next time the rung becomes false after either the .DN bit or the .ER bit sets, the controller clears the .EN bit.

6. When the rung becomes true, the instruction can execute again.



Fault Handling

This appendix describes motion errors and faults. The following table shows the contents of this appendix:

| For information about | See page |
|----------------------------------|----------|
| Handling Motion Faults | F-1 |
| Understanding Errors | F-1 |
| Understanding Minor/Major Faults | F-2 |

Handling Motion Faults

Two types of motion faults exist.

| Type | Description | Example |
|-------------|--|--|
| Errors | <ul style="list-style-type: none"> Do not impact controller operation Should be corrected to optimize execution time and ensure program accuracy | A Motion Axis Move (MAM) instruction with a parameter out of range |
| Minor/Major | <ul style="list-style-type: none"> Caused by a problem with the servo loop Can shutdown the controller if you do not correct the fault condition | The application exceeded the PositionErrorTolerance value |

Understanding Errors

Executing a motion instruction within an application program can generate errors. The MOTION_INSTRUCTION tag has a field that contains the error code (any number from 1 to 23 depending on the error). For more information on error codes for individual instructions, refer to the motion instruction chapters in the Logix5550 Controller Instruction Set Reference Manual, publication 1756-6.4.1.

Understanding Minor/Major Faults

Several faults can occur that are not caused by motion instructions. For example, a loss of encoder feedback or actual position exceeding an overtravel limit will cause faults. The motion faults are considered Type 11 faults with error codes from 1 to 32. For more information about motion error codes, refer to *Handling Controller Faults* in the Logix5550 Controller User Manual, publication 1756-6.5.12.

Note: You can configure a fault as either minor (non major) or major by using the Axis Wizard-Group window.

For more information about handling faults, see *Handling Controller Faults* in the Logix5550 Controller User Manual, publication 1756-6.5.12.

The terms in this glossary are specific to the ControlLogix line. For a complete guide to Rockwell Automation technical terms, refer to the Industrial Automation Glossary, publication AG-7.1.

A

- Active homing** A homing mode that allows you to choose a specific homing sequence. The active homing mode uses the trapezoidal velocity profile to perform the homing operation. See *Home, Passive homing*.
- Actual position** The current position of a physical or virtual axis as measured by the encoder or other feedback devices. See *Command position*.
- Alias tag** A tag that references another tag. An alias tag can refer to:
- Another alias tag or a base tag.
 - Memory within another tag by referencing a member of a structure, an array element, or a bit within a tag or member.
- See *Base tag, Tag*.
- Application** The combination of routines, programs, tasks, and I/O configuration used to define the operation of a single controller. See *Project*.
- Application program** See *Program*.
- Array** A numerically indexed sequence of elements, each of the same data type. In ControlLogix, an index starts at 0 and extends to the number of elements minus 1 (zero based). An array can have as many as three dimensions, unless it is a member of a structure where it can have only one dimension. An array tag occupies a contiguous block of memory in the controller, each element in sequence. See *Atomic data type, Structure*.
- Atomic data type** The basic definition used to allocate bits, bytes, or words of memory and to define their numeric interpretation, including BOOL, SINT, INT, DINT, and REAL data types. See *Application, Structure*.

Axis faulted operating state

An axis operating state in which a servo fault is present. The status of the drive enable output, the action of the servo, and the condition of the OK contact depend on the faults and fault actions that are present.

Axis ready operating state

An axis operating state that is the normal power-up state of an axis. In this operating state:

- The servo module drive enable output is inactive.
- Servo action is disabled.
- No servo faults are present.

B

Base tag

A tag that defines the memory where a data element is stored. See *Alias tag, Tag*.

Binary

Integer values displayed and entered in base 2 (each digit represents a single bit). Binary numbers are:

- Prefixed with 2#.
- Padded out to the length of the boolean or integer (1, 8, 16, or 32 bits).

When a binary number is displayed, every group of four digits is separated by an underscore for legibility. See *Decimal, Hexadecimal, Octal*.

Bit

Binary digit. The smallest unit of memory, which is represented by the digits 0 (cleared) and 1 (set).

BOOL

An atomic data type that stores the state of a single bit (0 or 1).

Byte

A unit of memory consisting of 8 bits.

C

| | |
|--------------------------------------|--|
| Command position | The position of the servo as generated by motion instructions. See <i>Actual position</i> . |
| Compatible module | <p>An electronic keying protection mode. To establish a connection with a module in this mode, you must match the following attributes of the physical module and the module configured using the programming software:</p> <ul style="list-style-type: none">• Vendor• Catalog number• Major revision <p>See <i>Disable keying</i>, <i>Exact match</i>.</p> |
| Continuous task | A task that runs continuously, restarting the execution of its programs when the last program finishes. If your application requires a continuous task, you can use only one continuous task. See <i>Periodic task</i> . |
| ControlBus | The backplane used by the 1756 chassis. |
| Controller scope | <p>Data accessible anywhere in the controller. Each controller contains a collection of tags that can be accessed by:</p> <ul style="list-style-type: none">• Routines in any program.• Alias tags in any program.• Other aliases in the controller scope. <p>See <i>Program scope</i>.</p> |
| Coordinated system time (CST) | A synchronized time value for all the modules within a single ControlBus chassis. To determine the relative time between data samples, you can compare samples that are timestamped with CST data from modules within a single ControlBus chassis. |

D

| | |
|---|--|
| Damping factor | An attribute that controls the dynamic response of a servo axis. The controller uses the damping factor attribute to calculate the maximum position servo bandwidth attribute. |
| Data type | A definition of the memory size and the layout of memory that the controller allocates when you use a tag of a specific data type. Data types can be atomic, structures, or arrays. |
| Decimal | Integer values displayed and entered in base 10. Decimal values: <ul style="list-style-type: none">• Do not use a prefix.• Are not padded to the length of the integer. See <i>Binary, Hexadecimal, Octal</i> . |
| Description | A field that allows to enter a brief description of a tag or module. In a description, you can use any printable character, including carriage returns, tabs, and spaces. <ul style="list-style-type: none">• Descriptions for tags are a maximum of 128 characters long.• Descriptions for modules are a maximum of 120 characters long. |
| Dimension | Specification of the size of an array. Arrays can have a maximum of three dimensions. |
| DINT | An atomic data type that stores a 32-bit signed integer value (-2,147,483,648 to +2,147,483,647). |
| Direct drive control operating state | An axis operating state that allows the servo module DAC to directly control an external drive. In this operating state: <ul style="list-style-type: none">• The servo module drive enable output is active.• Servo action is disabled.• No servo faults are present. |

Disable keying An electronic keying protection mode. To establish a connection with a module in this mode, you do not have to match any of the attributes of the physical module and the module configured in the programming software. See *Compatible module, Exact match*.

Download The process of transferring the contents of a project on a workstation into a controller. See *Upload*.

E

Electronic keying A feature of the 1756 I/O line where modules perform an electronic check to insure that the physical module is consistent with what you configured using the programming software. By using this feature, you can prevent the use of incorrect modules or incorrect revisions of modules. See *Compatible module, Disable keying, Exact match*.

Element An addressable unit of data that is a sub-unit of a larger unit of data. An element is a single unit in an array. See *Array*.

Error A malfunction caused by a motion instruction. For example, a Motion Axis Move (MAM) instruction with a parameter out of range would cause an error. This malfunction does not impact controller operation. To optimize execution time and ensure program accuracy, you should correct errors. See *Fault*.

Exact match An electronic keying protection mode. To establish a connection with a module in this mode, you must match the following attributes of the physical module and the module configured using the programming software:

- Vendor
- Catalog number
- Major revision
- Minor revision

See *Compatible module, Disable keying*.

Execution time

The total time required for the execution of an instruction.

Exponential

Real values displayed and entered in scientific or exponential format. An exponential number is always displayed with one digit to the left of the decimal point, followed by the decimal portion, and then by the exponent.

F

Fault

Any malfunction that interferes with normal system operation. See *Error*.

Float

Real values displayed and entered in floating point format. The number of digits to the left of the decimal point varies according to the magnitude of the number.

G

Gear

A type of axis motion that allows the controller to synchronize any axis to the actual or command position of another axis.

H

Hexadecimal

Integer values displayed and entered in base 16 (each digit represents four bits). Hexadecimal numbers are:

- Prefixed with 16#.
- Padded out to the length of the boolean or integer (1, 8, 16, or 32 bits).

When a hexadecimal number is displayed, every group of four digits is separated by an underscore for legibility. See *Binary*, *Decimal*, *Octal*.

| | |
|------------------------------|--|
| Home | A type of axis motion that calibrates the actual position of an axis. |
| Hookup diagnostics | <p>A set of three tests you can use to check encoder and marker connections.</p> <ul style="list-style-type: none">• The motor/encoder test verifies the motor/encoder hookup for an axis.• The encoder test verifies the encoder hookup for an axis.• The marker test verifies the marker hookup for an axis. |
| I | |
| <hr/> | |
| Immediate type timing | A type of instruction timing in which the instruction completes in one scan. See <i>Message type timing</i> , <i>Process type timing</i> . |
| Immediate value | A 32-bit signed integer value (-2,147,483,648 to +2,147,483,647). An immediate tag does not store a value. |
| Index | A reference used to specify an element within an array. |
| INT | An atomic data type that stores a 16-bit integer value (-32,768 to +32,767). |
| J | |
| <hr/> | |
| Jog | A type of axis motion that continuously moves (jogs) an axis. |
| K | |
| <hr/> | |
| Keying | A process of marking two devices with equivalent marking connectors. When you key devices, you can identify which devices connect to each other. For example, you can use keying pattern to identify which removable terminal block belongs with each module. |

M

Major fault

A malfunction, either hardware or instruction, that sets a major fault bit and processes fault logic to try to clear the fault condition. If the fault logic cannot clear the fault, then:

- Logic execution stops.
- The controller shuts down.
- The outputs go to their configured shutdown state.

See *Minor (non-major) fault*.

Major revision

A revision indicator in the 1756 line of modules. The major revision is updated any time there is a functional change to the module. See *Minor revision*.

Master axis

An axis that controls the slave axis during a gearing operation. See *Slave axis*.

Master controller (CST)

A controller designated as the Coordinated System Time (CST) master. All other modules in the system synchronize their CST values to this master. Within a single chassis, one and only one controller must be designated as the master.

Master coordinated system time

See Coordinated system time (CST).

Member

An element of a structure that has its own data type and name. Members can be structures as well, creating nested structure data types. Each member within a structure can be a different data type. See *Structure*.

Message type timing

A type of instruction timing in which the instruction completes over several scans because the instruction sends messages to the servo module. See *Immediate type timing*, *Process type timing*.

Minor (non-major) fault

A malfunction, either hardware or instruction, that sets a minor fault bit, but allows the logic scan to continue. See *Major fault*.

Minor revision

A revision indicator in the 1756 line of modules. The minor revision is updated any time there is a change to a module that does not affect its function or interface. See *Major revision*.

Move

A type of axis motion that moves a physical axis to a new position.

N

Name

A title that identifies tags and modules. The naming conventions are IEC-1131-3 compliant. A name:

- must begin with an alphabetic character (A-Z or a-z) or an underscore (_).
- can contain only alphabetic characters, numeric characters, and underscores.
- can have as many as 40 characters.
- must not have consecutive or trailing underscore characters (_).

O

Object

A structure of data that stores status information. When you enter a GSV/SSV instruction, you specify the object and its attribute that you want to access. In some cases, you might also have to specify the object name because more than one instance of the same type of object exists. For example, your application can contain several tasks, and each task has its own TASK object that you access by the task name.

Octal

Integer values displayed and entered in base 8 (each digit represents three bits). Octal numbers are:

- Prefixed with 8#.
- Padded out to the length of the boolean or integer (1, 8, 16, or 32 bits).

When an octal number is displayed, every group of three digits is separated by an underscore for legibility. See *Binary*, *Decimal*, *Hexadecimal*.

P

| | |
|----------------------------|--|
| Passive homing | A homing mode that redefines the current absolute position on the next occurrence of the encoder marker. See <i>Active homing</i> . |
| Periodic task | A task that is triggered at a specific time interval. Whenever the time interval expires, the task is triggered and its programs are executed. Each controller can have as many as 32 periodic tasks. See <i>Continuous task</i> . |
| Position error | The difference between the actual position and the command position of an axis. |
| Position-only axis | A type of axis in which you use the axis to monitor axis position. See <i>Servo axis</i> . |
| Process type timing | A type of instruction timing in which the instruction could take an indefinite amount of time to complete. See <i>Immediate type timing</i> , <i>Message type timing</i> . |
| Program | A set of related routines and a collection of tags. When a program is executed by a task, execution of logic starts at the configured main routine. That main routine can execute subroutines using the JSR instruction. If a program fault occurs, execution jumps to a configured fault routine for the program. Any of the routines in a program can access the program tags, but routines in other programs cannot access these tags. See <i>Routine</i> , <i>Task</i> . |
| Program scope | Data accessible only within the current program. Each program contains a collection of tags that can only be referenced by the routines and alias tags in that program. See <i>Controller scope</i> . |
| Project | The file that the programming software uses to store a controller's logic and configuration. See <i>Application</i> . |

R

- REAL** An atomic data type that stores a 32-bit IEEE floating-point value.
- Removal and insertion under power (RIUP)** A ControlLogix feature that allows a user to install and remove a module while chassis power is applied.
- Routine** A set of logic instructions in a single programming language, such as a ladder diagram. Routines provide the executable code for the project in a controller. A routine is similar to a program file in a PLC or SLC processor. See *Program, Task*.

S

- S-curve profile** A motion profile in which uses a controlled jerk to perform motion. The s-curve motion profile produces the least motor stress. See *Trapezoidal profile*.
- Scope** Defines where you can access a particular set of tags. See *Controller scope, Program scope*.
- Servo axis** A type of axis in which you can use the axis as a full closed-loop servo. See *Position-only axis*.
- Servo control operating state** An axis operating state that allows the servo module to perform closed loop motion. In this operating state:
- The servo module drive enable output is active.
 - Servo action is enabled.
 - No servo faults are present.
 - The axis is forced to maintain the commanded servo position.
- Shutdown operating state** An axis operating state that allows the OK relay contacts to open a set of contacts in the E-stop string of the drive power supply. In this operating state:
- The servo module drive enable output is inactive.
 - Servo action is disabled.
 - The OK contact is open.

| | |
|----------------------------|--|
| SINT | An atomic data type that stores an 8-bit signed integer value (-128 to +127). |
| Slave axis | An axis that follows the master axis during a gearing operation. See <i>Master axis</i> . |
| Structure | An object that stores a group of data, each of which can be a different data type. The controller and each I/O module you configure have their own predefined structures. You can also create specialized user-defined structures, using any combination of individual tags and most other structures. See <i>Member</i> , <i>User-defined structure</i> . |
| T <hr/> | |
| Tag | A named area of the controller's memory where it stores data. Tags are the basic mechanism for allocating memory, referencing data from logic, and monitoring data. See <i>Alias tag</i> , <i>Base tag</i> . |
| Task | <p>A scheduling mechanism for executing a program. A task can be configured to run as a continuous task or a periodic task.</p> <ul style="list-style-type: none">• You can create a maximum of 32 tasks to schedule programs.• You can execute a maximum of 32 programs when a task is triggered. <p>See <i>Continuous task</i>, <i>Periodic task</i>.</p> |
| Timestamp | A relative time reference that a ControlLogix process records when a change in input data occurs. |
| Trapezoidal profile | A motion profile in which the velocity-vs.-time profile resembles a trapezoid. This profile is characterized by constant acceleration, constant velocity, and constant deceleration. If you want the fastest acceleration and deceleration times, use the trapezoidal motion profile. See <i>S-curve profile</i> . |

U

Upload

The process of transferring the contents of the controller into a project file on a workstation. See *Download*.

User-defined structure

A single named entity that groups different types of data. A user-defined structure contains one or more data definitions called members. Creating a member in a user-defined structure is just like creating an individual tag. The data type for each member determines the amount of memory allocated for the member. The data type for each member can be:

- An atomic data type
- A product-defined structure
- A user-defined structure
- A single dimension array of an atomic data type
- A single dimension array of a product-defined structure
- A single dimension array of a user-defined structure

Numerics

- 1756-M02AE servo module 1-1
 - Adding to a program 4-1, 4-5
 - Adding an axis 4-7
 - Additional modules and axes 4-27
 - Auto tuning 4-28
 - Block diagrams
 - Torque servo drive B-3
 - Velocity servo drive B-4
 - Coarse update rate calculations A-4
 - Components 2-2
 - Configuring a motion axis 4-1, 4-8
 - Features 1-2
 - Getting started 3-1 to 3-36
 - Hookup diagnostics 4-28
 - Installing 2-1 to 2-16
 - Before installing your module 2-1
 - Removable terminal block (RTB) 2-12
 - Keying your module 2-8
 - LED indicators 2-14
 - Loop and interconnect diagrams B-1
 - Removable terminal block (RTB) 2-3
 - Removing
 - Module from the chassis 2-16
 - Removable terminal block (RTB) 2-15
 - Specifications A-1
 - Troubleshooting 6-1
 - Wiring diagrams
 - 1394 drive B-8
 - 1394-SA15 cable B-9
 - 24V registration sensor B-10
 - 5V registration sensor B-10
 - Home limit switch B-11
 - OK contacts B-12
 - Servo module RTB B-5
 - Ultra 100 drive B-6
 - Ultra 200 drive B-7

A

- Adding to a program
 - A ladder rung 3-29
 - A motion module 3-6, 4-1 to 4-39
 - An MSO instruction 3-30
 - An XIC instruction 3-29
 - Assigning an axis 4-7
- Application program
 - Creating 3-29, 3-34 to 3-35
 - Developing 1-4, 4-1, 4-38
 - Downloading 4-29
 - Entering ladder logic 3-29 to 3-35
 - Example 3-29, 4-39
 - Main routine 3-19, 3-29
- Assigning in an application program
 - Additional modules and axes 4-27
 - Master controller 4-2
- Auto tuning 3-19 to 3-28, 4-28 to 4-37
 - Starting 3-25, 4-34
 - Tune bandwidth window 3-26, 4-34
- AXIS control structure C-2
- Axis Properties window 3-21, 3-23, 3-24, 3-27, 4-30
 - Dynamics 3-28, 4-36
 - Gains 3-27, 4-36
 - Hookup 3-22, 4-30, 4-32
 - Tune servo 3-25, 4-33
- Axis Wizard
 - Dynamics 3-17, 4-26
 - Fault action 3-15, 4-18
 - Feedback 3-12, 4-13
 - Gains 3-17, 4-24
 - General 3-9, 4-8
 - Group 3-9, 3-11, 4-9, 4-12
 - Axis assignment 3-10, 4-10
 - Update rates 3-11, 4-11
 - Homing 3-13, 4-15

- Hookup 3-16, 4-22
- Overtravels 3-14, 4-17
- Positioning 3-13, 4-14
- Servo 3-14, 4-17
- Tune 3-16, 4-23
- Units 3-12, 4-12

B

- Block diagrams for a 1756-M02AE module B-2
 - With a torque servo drive B-3
 - With a velocity servo drive B-4

C

- Cage clamp RTB, wiring 2-11
- Configuring a motion axis 3-9 to 3-18, 4-8 to 4-27
- Control structures C-1
 - AXIS C-2
 - MOTION_GROUP C-8
 - MOTION_INSTRUCTION C-11
 - Error codes C-12
 - Execution status C-13
 - Message status C-13
- Controller Properties window 3-4, 4-3
- ControlLogix motion control 1-1
 - Components 1-2
 - Features 1-2
- Conventions used in this manual P-3
- Course update rate calculations A-4
 - Action timing A-5
 - Baseline task time A-4
 - Calculation worksheet A-6
 - Sample calculation A-7, A-9

D

- Diagnostic tests
 - Marker test 3-24
 - Motor/encoder test 3-22

- Diagrams
 - Block B-2
 - Wiring B-5
- Documentation P-3
- Downloading a program 3-20, 4-28, 4-29
- DRIVE LED indicator 6-3

E

- Electrostatic discharge, preventing 2-4
- Errors F-1
- European Union Directive
 - Compliance 2-5
 - EMC directive 2-5
 - Low voltage directive 2-5

F

- Fault handling F-1
 - Errors F-1
 - Minor/major faults F-2
 - Motion faults F-1
- Faults F-1
 - Types 1-6
- FDBK LED indicator 6-2

G

- Getting started with your motion module 3-1 to 3-36
 - Adding a 1756-M02AE module 3-6
 - Configuring your axis 3-9
 - Control system components 3-2
 - Entering an application program 3-29
 - Naming an axis 3-8
 - Running hookup diagnostics and auto tuning 3-19
 - Setting master system time 3-3
 - Steps before beginning 3-1
 - Tasks 3-2
- Greyed-out fields P-3, 4-8

-
- GSV instruction D-1
 - Reading status and configuration parameters 1-6
 - H
 - Hookup diagnostics 3-19 to 3-28, 4-28 to 4-37
 - I
 - Immediate instruction timing E-1
 - Installing your motion module 2-1 to 2-16
 - K
 - Keying
 - Module 2-8
 - Pattern 2-8
 - Removable terminal block 2-8, 2-9
 - L
 - LED indicators 2-14
 - Logix5550 controller 1-1
 - Features 1-2
 - M
 - Main routine 3-19, 3-29
 - Major faults F-2
 - Manual
 - Conventions P-3
 - Purpose P-2
 - Who should use P-1
 - Marker test
 - Starting 3-24, 4-32
 - Master coordinated system time 4-2
 - Message instruction timing E-3
 - Minor faults F-2
 - Module Properties window 3-18, 3-21, 4-27, 4-29
 - Motion attributes D-1
 - Changing configuration parameters 1-6
 - Motion instance variables D-1
 - Understanding status and configuration parameters 1-6
 - Motion configuration instructions 5-6
 - Motion Apply Axis Tuning (MAAT) 5-6
 - Motion Apply Hookup Diagnostic (MAHD) 5-6
 - Motion Run Axis Tuning (MRAT) 5-6
 - Motion Run Hookup Diagnostic (MRHD) 5-6
 - Motion event instructions 5-5
 - Motion Arm Registration (MAR) 5-5
 - Motion Arm Watch Position (MAW) 5-5
 - Motion Disarm Registration (MDR) 5-5
 - Motion Disarm Watch Position (MDW) 5-5
 - Motion group instructions 5-4
 - Motion Group Programmed Stop (MGPS) 5-4
 - Motion Group Shutdown (MGSD) 5-4
 - Motion Group Shutdown Reset (MGSR) 5-4
 - Motion Group Stop (MGS) 5-4
 - Motion Group Strobe Position (MGSP) 5-4
 - Motion instance variables D-1
 - Motion instructions 5-1
 - Motion Apply Axis Tuning (MAAT) 5-6
 - Motion Apply Hookup Diagnostic (MAHD) 5-6
 - Motion Arm Registration (MAR) 5-5
 - Motion Arm Watch Position (MAW) 5-5
 - Motion Axis Fault Reset (MAFR) 5-2
 - Motion Axis Gear (MAG) 5-3
 - Motion Axis Home (MAH) 5-3
 - Motion Axis Jog (MAJ) 5-3
 - Motion Axis Move (MAM) 5-3
 - Motion Axis Shutdown (MASD) 5-2
 - Motion Axis Shutdown Reset (MASR) 5-2
 - Motion Axis Stop (MAS) 5-3
 - Motion Change Dynamics (MCD) 5-3
 - Motion configuration instructions 5-6
 - Motion Direct Drive Off (MDF) 5-2
 - Motion Direct Drive On (MDO) 5-2
 - Motion Disarm Registration (MDR) 5-5
 - Motion Disarm Watch Position (MDW) 5-5
 - Motion event instructions 5-5
 - Motion group instructions 5-4

- Motion Group Programmed Stop (MGPS) 5-4
 - Motion Group Shutdown (MGSD) 5-4
 - Motion Group Shutdown Reset (MGSR) 5-4
 - Motion Group Stop (MGS) 5-4
 - Motion Group Strobe Position (MGSP) 5-4
 - Motion move instructions 5-3
 - Motion Redefine Position (MRP) 5-3
 - Motion Run Axis Tuning (MRAT) 5-6
 - Motion Run Hookup Diagnostic (MRHD) 5-6
 - Motion Servo Off (MSF) 5-2
 - Motion Servo On (MSO) 5-2
 - Motion state instructions 5-2
 - Timing E-1
 - Immediate E-1
 - Message E-3
 - Process E-5
 - Motion module. See 1756-M02AE servo module.
 - Motion move instructions 5-3
 - Motion Axis Gear (MAG) 5-3
 - Motion Axis Home (MAH) 5-3
 - Motion Axis Jog (MAJ) 5-3
 - Motion Axis Move (MAM) 5-3
 - Motion Axis Stop (MAS) 5-3
 - Motion Change Dynamics (MCD) 5-3
 - Motion Redefine Position (MRP) 5-3
 - Motion state instructions 5-2
 - Motion Axis Fault Reset (MAFR) 5-2
 - Motion Axis Shutdown (MASD) 5-2
 - Motion Axis Shutdown Reset (MASR) 5-2
 - Motion Direct Drive Off (MDF) 5-2
 - Motion Direct Drive On (MDO) 5-2
 - Motion Servo Off (MSF) 5-2
 - Motion Servo On (MSO) 5-2
 - MOTION_GROUP control structure C-8
 - MOTION_INSTRUCTION control structure C-11
 - Error codes C-12
 - Execution status C-13
 - Message status C-13
 - Understanding 1-5
 - Motor/encoder test
 - Starting 3-22, 4-30
- N**
- New Module window 3-7, 4-6
 - New Tag window 3-31
 - AXIS 3-8
 - MOTION_GROUP 3-10, 4-9
- O**
- Offline 3-29
 - OK LED indicator 6-1
- P**
- Performance guidelines A-1
 - Power requirements, determining 2-3
 - Process instruction timing E-5
 - Program. See Application program.
 - Purpose of this manual P-2
- R**
- Related documentation P-3
 - Removable terminal block (RTB) 2-3
 - Assembling 2-12
 - Cage clamp 2-11
 - Installing into module 2-12
 - Keying 2-8, 2-9
 - Removing from module 2-15
 - Spring clamp 2-10
 - Wiring 2-10
 - Removing and Inserting Under Power (RIUP) 2-4
 - Rockwell Automation support P-4
 - Local product support P-4
 - Technical product assistance P-4
 - RSLogix 5000 programming software 1-1
 - Adding a motion module 4-1
 - Adding a rung 3-29
 - Adding an MSO instruction 3-30

- Adding an XIC instruction 3-29
 - Application program
 - Developing 4-1, 4-38
 - Example 4-39
 - Main routine 3-19
 - Configuring a motion module 4-1
 - Control structures C-1
 - Creating an application program 3-34 to 3-35
 - Creating tags 3-31 to 3-33
 - Downloading an application program 3-20, 4-28, 4-29
 - Entering a variable 3-29
 - Example program 3-29
 - Fault handling F-1
 - Features 1-3
 - Going offline 3-29
 - GSV/SSV instructions D-1
 - Instruction timing E-1
 - Main routine 3-29
 - Motion attributes D-1
 - Motion instructions 5-1
- RUIP. See Removing and Inserting Under Power (RUIP).
- S
- Sample program 3-29
 - Select Module Type window 3-6, 4-5
 - Setting the master coordinated system time 3-3 to 3-5, 4-2 to 4-4
 - Specifications A-1
 - Spring clamp RTB, wiring 2-10
 - SSV instruction D-1
 - Changing configuration parameters 1-6
- T
- Tags
 - Creating 3-31 to 3-33
 - New Tag window
 - AXIS 3-8
 - MOTION_GROUP 3-10, 4-9
- Troubleshooting 6-1
 - DRIVE LED indicator 6-3
 - FDBK LED indicator 6-2
 - OK LED indicator 6-1
 - Tune Bandwidth window 3-26
 - Tuning. See Auto tuning.
- U
- Unavailable fields P-3, 4-8
- V
- Variables, using 3-29
- W
- Who should use this manual P-1
- Windows
- Axis properties 3-21, 3-23, 3-24, 3-27, 4-30
 - Dynamics 3-28
 - Gains 3-27
 - Hookup 3-22
 - Tune servo 3-25
 - Axis Wizard-Dynamics 3-17, 4-26
 - Axis Wizard-Fault action 3-15, 4-18
 - Axis Wizard-Feedback 3-12, 4-13
 - Axis Wizard-Gains 3-17, 4-24
 - Axis Wizard-General 3-9, 4-8
 - Axis Wizard-Group 3-9, 3-11, 4-9, 4-12
 - Axis assignment 3-10, 4-10
 - Update rates 3-11, 4-11
 - Axis Wizard-Homing 3-13, 4-15
 - Axis Wizard-Hookup 3-16, 4-22
 - Axis Wizard-Overtravels 3-14, 4-17
 - Axis Wizard-Positioning 3-13, 4-14
 - Axis Wizard-Servo 3-14, 4-17
 - Axis Wizard-Tune 3-16, 4-23
 - Axis Wizard-Units 3-12, 4-12
 - Controller properties 3-4, 4-3

Module properties 3-18, 3-21, 4-27, 4-29

New module 3-7, 4-6

New tag 3-8, 3-10, 3-31, 4-9

Select module type 3-6, 4-5

Tune bandwidth 3-26, 4-34

Wiring diagrams B-5

1394 drive B-8

1394-SA15 cable B-9

24V registration sensor B-10

5V registration sensor B-10

Home limit switch B-11

OK contacts B-12

Servo module RTB B-5

Ultra 100 drive B-6

Ultra 200 drive B-7



Rockwell Automation helps its customers receive a superior return on their investment by bringing together leading brands in industrial automation, creating a broad spectrum of easy-to-integrate products. These are supported by local technical resources available worldwide, a global network of system solutions providers, and the advanced technology resources of Rockwell.

Worldwide representation.



Argentina • Australia • Austria • Bahrain • Belgium • Bolivia • Brazil • Bulgaria • Canada • Chile • China, People's Republic of • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark • Dominican Republic • Ecuador • Egypt • El Salvador • Finland • France • Germany • Ghana • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Iran • Ireland • Israel • Italy • Jamaica • Japan • Jordan • Korea • Kuwait • Lebanon • Macau • Malaysia • Malta • Mexico • Morocco • The Netherlands • New Zealand • Nigeria • Norway • Oman • Pakistan • Panama • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania • Russia • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic of • Spain • Sweden • Switzerland • Taiwan • Thailand • Trinidad • Tunisia • Turkey • United Arab Emirates • United Kingdom • United States • Uruguay • Venezuela

Rockwell Automation Headquarters, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444

Rockwell Automation European Headquarters SA/NV, avenue Hermann Debrouxlaan, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40

Rockwell Automation Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846