FLEX 5000 Serial Modules

Catalog Numbers 5094-SERIAL, 5094-SERIALXT

Allen-Bradley
by ROCKWELL AUTOMATION

User Manual

Original Instructions

Important User Information

Read this document and the documents listed in the additional resources section about installation, configuration, and operation of this equipment before you install, configure, operate, or maintain this product. Users are required to familiarize themselves with installation and wiring instructions in addition to requirements of all applicable codes, laws, and standards.

Activities including installation, adjustments, putting into service, use, assembly, disassembly, and maintenance are required to be carried out by suitably trained personnel in accordance with applicable code of practice.

If this equipment is used in a manner not specified by the manufacturer, the protection provided by the equipment may be impaired.

In no event will Rockwell Automation, Inc. be responsible or liable for indirect or consequential damages resulting from the use or application of this equipment.

The examples and diagrams in this manual are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular installation, Rockwell Automation, Inc. cannot assume responsibility or liability for actual use based on the examples and diagrams.

No patent liability is assumed by Rockwell Automation, Inc. with respect to use of information, circuits, equipment, or software described in this manual.

Reproduction of the contents of this manual, in whole or in part, without written permission of Rockwell Automation, Inc., is prohibited.

Throughout this manual, when necessary, we use notes to make you aware of safety considerations.

WARNING: Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence.

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

Labels may also be on or inside the equipment to provide specific precautions.

SHOCK HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.

BURN HAZARD: Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.

ARC FLASH HAZARD: Labels may be on or inside the equipment, for example, a motor control center, to alert people to potential Arc Flash. Arc Flash will cause severe injury or death. Wear proper Personal Protective Equipment (PPE). Follow ALL Regulatory requirements for safe work practices and for Personal Protective Equipment (PPE).

Chapter 3

Configure FLEX 5000 Serial Modules

Chapter 4

Troubleshoot Your Module

Appendix A

Module Tags

Appendix B

Master Command List

Appendix C

Programming Example

Appendix D

ASCII Conversion Tables

Index

Notes:

About This Publication

This manual describes how to use FLEX 5000™ serial modules in Logix 5000™ control systems.

Make sure that you are familiar with the following:

- Use of a controller in a Logix 5000 control system
- Use of an EtherNet/IP™ network, if the serial modules are installed in a remote location from the controller that is accessible over an EtherNet/IP network
- Studio 5000 Logix Designer® application environment

Download Firmware, AOP, EDS, and Other Files

Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes from the Product Compatibility and Download Center at rok.auto/pcdc.

Additional Resources

These documents contain additional information concerning related products from Rockwell Automation.

Resource	Description
FLEX 5000 Modules Specifications Technical Data, publication 5094-TD001	Provides specifications for FLEX 5000 EtherNet/IP adapters, I/O modules, terminal bases, and accessories.
FLEX 5000 EtherNet/IP Adapters with RJ45 Ports Installation Instructions, publication 5094-IN001	Describes how to install and wire the FLEX 5000 EtherNet/IP adapters.
FLEX 5000 EtherNet/IP Adapters with SFP Support Installation Instructions, publication 5094-IN002	Describes how to install and wire the FLEX 5000 EtherNet/IP adapters with SFP support.
FLEX 5000 Digital 16-point Sinking Input Modules Installation Instructions, publication 5094-IN003	Describes how to install and wire the FLEX 5000 digital input modules.
FLEX 5000 FLEX 5000 Digital 16-point Sourcing Output Modules Installation Instructions, publication 5094-IN004	Describes how to install and wire the FLEX 5000 digital output modules.
FLEX 5000 Digital 8-point Isolated Relay Output Modules Installation Instructions, publication 5094-IN005	Describes how to install and wire the FLEX 5000 digital isolated relay output modules.
FLEX 5000 Analog 8-channel Current/Voltage Input Modules Installation Instructions, publication 5094-IN006	Describes how to install and wire the FLEX 5000 analog input modules.
FLEX 5000 Analog 8-channel Current/Voltage Output Modules Installation Instructions, publication 5094-IN007	Describes how to install and wire the FLEX 5000 analog output modules.
FLEX 5000 Analog 8-channel Current/Voltage/RTD/Thermocouple Input Modules Installation Instructions, publication 5094-IN008	Describes how to install and wire the FLEX 5000 analog RTD/Thermocouple input modules.
FLEX 5000 High-speed Counter Modules Installation Instructions, publication 5094-IN009	Describes how to install and wire the FLEX 5000 high-speed counter modules.
FLEX 5000 Terminal Base Assembly Modules Installation Instructions, publication 5094-IN010	Describes how to install the FLEX 5000 terminal base assemblies.
FLEX 5000 Interconnect Cables Installation Instructions, publication 5094-IN011	Describes how to install the FLEX 5000 interconnect cable.
FLEX 5000 Digital 16-point Sinking Safety Input Modules Installation Instructions, publication 5094-IN012	Describes how to install and wire the FLEX 5000 digital safety input modules.
FLEX 5000 Digital 16-point Sourcing Safety Output Modules Installation Instructions, publication 5094-IN013	Describes how to install and wire the FLEX 5000 digital safety output modules.
FLEX 5000 Relay 4-point Safety Output Modules Installation Instructions, publication 5094-IN015	Describes how to install and wire the FLEX 5000 relay 4-point safety output modules.
FLEX 5000 Analog 8-channel Isolated Current/Voltage/HART Input Modules Installation Instructions, publication 5094-IN020	Describes how to install and wire the FLEX 5000 analog isolated HART input modules.
FLEX 5000 Analog 8-channel Isolated Current/Voltage/HART Output Modules Installation Instructions, publication 5094-IN021	Describes how to install and wire the FLEX 5000 analog isolated HART output modules.

Resource	Description
FLEX 5000 Digital 32-point Sinking Input Modules Installation Instructions, publication 5094-IN022	Describes how to install and wire the FLEX 5000 digital input modules.
FLEX 5000 Digital 8-point High Current Output Modules Installation Instructions, publication 5094-IN023	Describes how to install and wire the FLEX 5000 digital high current output modules.
FLEX 5000 Digital 32-point Sourcing Output Modules Installation Instructions, publication 5094-IN024	Describes how to install and wire the FLEX 5000 digital output modules.
FLEX 5000 Digital 16-point 120V AC Input Modules Installation Instructions, publication 5094-IN025	Describes how to install and wire the FLEX 5000 digital AC input modules.
FLEX 5000 Digital 8-point 240V AC Input Modules Installation Instructions, publication 5094-IN026	Describes how to install and wire the FLEX 5000 digital AC input modules.
FLEX 5000 Digital 16-point 120/240V AC Output Modules Installation Instructions, publication 5094-IN027	Describes how to install and wire the FLEX 5000 digital AC output modules.
FLEX 5000 Serial Modules Installation Instructions, publication 5094-IN028	Describes how to install and wire the FLEX 5000 serial interface modules.
Replacement Guidelines: Logix 5000 Controllers Reference Manual, publication 1756-RM100	Provides guidelines on how to replace the following: – ControlLogix 5560/5570 controller with a ControlLogix 5580 controller – CompactLogix 5370 L3 controllers with a CompactLogix 5380 controller
FLEX 5000 Standard and Safety Digital I/O Modules User Manual, publication 5094-UM001	Provides information on how to configure and operate FLEX 5000 digital I/O modules (standard and safety).
FLEX 5000 Analog I/O Modules User Manual, publication 5094-UM002	Provides information on how to configure and operate FLEX 5000 analog I/O modules.
FLEX 5000 High-speed Counter Module User Manual, publication 5094-UM003	Provides information on how to configure and operate FLEX 5000 high-speed counter modules.
FLEX 5000 EtherNet/IP Adapter User Manual, publication 5094-UM005	Provides information on how to configure and operate FLEX 5000 EtherNet/IP adapters.
FLEX 5000 Analog 8-channel Isolated Current/Voltage/HART Input and Output Modules User Manual, publication 5094-UM007	Provides information on how to configure and operate FLEX 5000 analog isolated HART modules.
CompactLogix 5380 Controllers User Manual, publication 5069-UM001	Describes how to configure, operate, and troubleshoot CompactLogix 5380 controllers.
Integrated Architecture and CIP Sync Configuration Application Technique, publication IA-AT003	Provides information about CIP Sync™ technology and how to synchronize clocks within the Integrated Architecture® system.
Electronic Keying in Logix 5000 Control Systems Application Technique, publication LOGIX-AT001	Describes how to use electronic keying in Logix 5000 control system applications.
Logix 5000 Controllers Tasks, Programs, and Routines Programming Manual, publication 1756-PM005	Provides more information on event tasks and event task configuration.
Position-based Output Control with the MAOC Instruction, publication 1756-AT017	Describes how to configure time-scheduled output control with the MAOC instruction.
EtherNet/IP Network Devices User Manual, ENET-UM006	Describes how to configure and use EtherNet/IP devices to communicate on the EtherNet/IP network.
Ethernet Reference Manual, ENET-RM002	Describes basic Ethernet concepts, infrastructure components, and infrastructure features.
System Security Design Guidelines Reference Manual, SECURE-RM001	Provides guidance on how to conduct security assessments, implement Rockwell Automation products in a secure system, harden the control system, manage user access, and dispose of equipment.
Industrial Components Preventive Maintenance, Enclosures, and Contact Ratings Specifications, publication IC-TD002	Provides a quick reference tool for Allen-Bradley industrial automation controls and assemblies.
Safety Guidelines for the Application, Installation, and Maintenance of Solid-state Control, publication SGI-1.1	Designed to harmonize with NEMA Standards Publication No. ICS 1.1-1987 and provides general guidelines for the application, installation, and maintenance of solid-state control in the form of individual devices or packaged assemblies incorporating solid-state components.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, rok.auto/certifications	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at rok.auto/literature.

FLEX 5000 Serial Module Operation in a Logix 5000 Control System

IMPORTANT You cannot use FLEX 5000 I/O modules with all Logix 5000 controllers. For example, you can use FLEX 5000 I/O modules with CompactLogix™ 5380 and ControlLogix® 5580 controllers but not with CompactLogix 5370 and ControlLogix 5570 controllers.

You can use FLEX 5000 I/O modules with Logix 5000 controllers as remote I/O modules only.

Throughout this publication, the term **Logix 5000 controller** refers to the controllers with which you can use FLEX 5000 I/O modules in a given capacity. The term does not refer to all Logix 5000 controllers.

For the most current information on the Logix 5000 controllers with which you can use FLEX 5000 I/O modules, see the product description at https://www.ab.com.

Use Studio 5000 Logix Designer application Version 31 or greater. You must install an Add-on Profile to use the serial module. To find the Add-on Profile, go to rok.auto/pcdc.

Logix 5000 controllers use FLEX 5000 I/O modules to control devices in a control system. The controllers access the modules over an EtherNet/IP network.

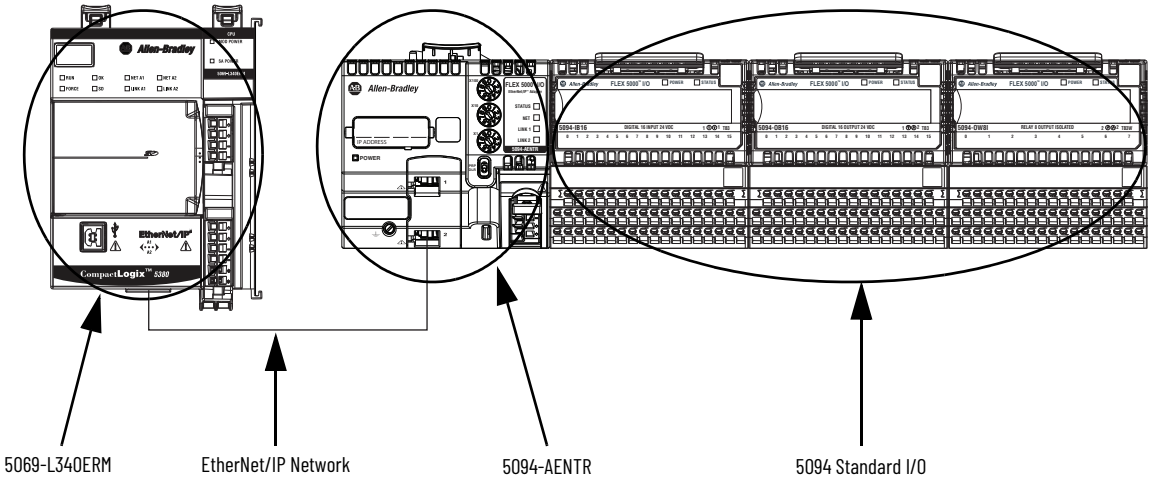
FLEX 5000 I/O modules use terminal base (TB) assemblies to connect field-side wiring. The modules are installed to the right of a FLEX 5000 EtherNet/IP adapter.

The FLEX 5000 serial module provides two independent channels that function as network interfaces to a wide variety of RS-232C, RS-422, and RS-485 devices. The channels can transmit data to and receive data from serial devices using the following communication mode:

- Generic ASCII
- Modbus RTU (Master/Slave)
- Modbus ASCII (Master/Slave)

Logix 5000 controllers can exchange data with the modules over the network. Figure 1 shows a standard controller with standard I/O modules. Standard controllers **do not** support safety I/O modules.

Figure 1 - FLEX 5000 Standard I/O Modules in a Logix 5000 Control System



Controller and Software Compatibility

Controller and programming software compatibility requirements apply when you use FLEX 5000 standard and safety I/O modules. A module type and how it is used affect which requirements apply.

You must also consider Logix Designer application version requirements when you design your system. For example, you can use FLEX 5000 safety I/O modules with only version 32 or greater of the Logix Designer application.

Controller Compatibility

Compatibility between Logix 5000 controllers and FLEX 5000 I/O modules varies based on module type, that is, whether the module is standard or safety.

While you must pair safety I/O with a safety controller, you can also pair standard I/O with a safety controller. For example, ControlLogix 5580 controllers are compatible with FLEX 5000 standard I/O modules. GuardLogix® 5580 controllers are compatible with FLEX 5000 standard and safety I/O modules.

Firmware and Software Compatibility

Table 1 describes the module compatibility requirements when you use FLEX 5000 serial modules with Logix 5000 controllers.

IMPORTANT You must use adapter firmware revision 3.011 or later with firmware revision 2.011 serial modules.

Table 1 - FLEX 5000 Serial Modules Controller and Software Compatibility Requirements

Modules	Controllers		Logix Designer Application Version
	System	Cat. Nos.	
5094-SERIAL, 5094-SERIALXT	CompactLogix 5380	5069-L320ER, 5069-L340ERM	31.00.00 or later
		5069-L306ER, 5069-L306ERM, 5069-L310ER, 5069-L310ERM, 5069-L310ER-NSE, 5069-L310ERS2, 5069-L320ERM, 5069-L330ER, 5069-L330ERM, 5069-L340ER	31.00.00 or later
		5069-L350ERM, 5069-L380ERM, 5069-L3100ERM	31.00.00 or later
	Compact GuardLogix 5380	5069-L306ERMS2, 5069-L306ERS2, 5069-L310ERS2, 5069-L310ERMS2, 5069-L320ERS2, 5069-L320ERMS2, 5069-L330ERS2, 5069-L330ERMS2, 5069-L340ERS2, 5069-L340ERMS2, 5069-L350ERS2, 5069-L350ERMS2, 5069-L380ERS2, 5069-L380ERMS2, 5069-L3100ERS2, 5069-L3100ERMS2	31.00.00 or later
	ControlLogix 5580	1756-L83E, 1756-L85E	31.00.00 or later
		1756-L81E, 1756-L82E, 1756-L84E	31.00.00 or later
	GuardLogix 5580	1756-L81ES, 1756-L82ES, 1756-L83ES, 1756-L84ES	31.00.00 or later

Secure Access to the System

To secure access to the [device] by authorized users only, consider these options:

- Password helps protect the source and execution of the control program
- Remove the key from the controller
- Deploy EtherNet/IP devices in accordance with recommended architectures and concepts. See the Deploying a Resilient Converged Plantwide Ethernet Architecture Design and Implementation Guide, publication ENET-TD001.
- Implement physical barriers, such as locked cabinets

To secure access to the system, consider these options:

- Follow industry best practices to harden your personal computers and servers, including anti-virus/anti-malware and application whitelisting solutions.
- The recommendations are published at the Rockwell Automation technical support center in Knowledgebase article *Rockwell Automation Customer Hardening Guidelines, #546987*. The technical support center is available at: rok.auto/knowledgebase.
- Develop and deploy backup and disaster recovery policies and procedures. Test backups on a regular schedule.
- Minimize network exposure for all control system devices and systems, and confirm that they are not accessible from the Internet.
- Locate control system networks and devices behind firewalls and isolate them from the business network.
- Subscribe to the Knowledgebase article *Industrial Security Advisory Index, #54102*, so you have access to information about security matters that affect Rockwell Automation products. The article is available at the technical support center at: rok.auto/knowledgebase.

Ownership

Every I/O module in a Logix 5000 control system must be owned by a controller, also known as the owner-controller. When the FLEX 5000 serial module is used in a Logix 5000 control system, the owner-controller performs the following:

- Stores configuration data for every module that it owns.
- Can reside in a location that differs from the FLEX 5000 I/O modules.
- Sends the I/O module configuration data to define module behavior and begin operation in the control system.

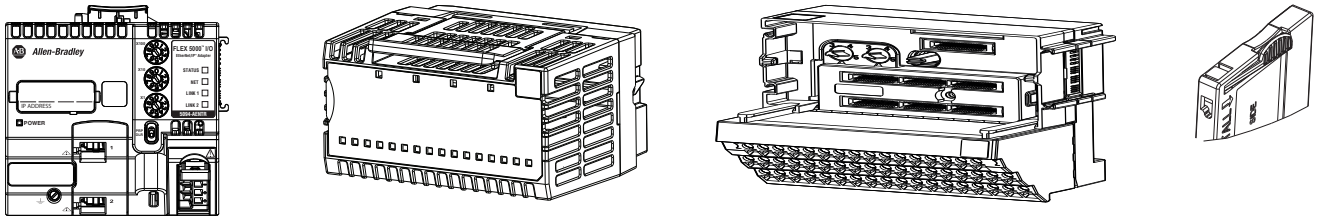
Each FLEX 5000 I/O module must continuously maintain communication with its owner-controller during normal operation.

Typically, each I/O module in a FLEX 5000 I/O system has only one owner-controller. Modules with output tags are limited to one owner-controller.

Construct a FLEX 5000 I/O System

FLEX 5000 I/O is a small, modular I/O system for distributed applications that performs all functions of rack-based I/O. The FLEX system contains the components that are pictured below.

Adapter I/O module Terminal base End cap



- Adapter – transfers read and write configuration data to and from the I/O module
- Terminal base – contains a terminal strip to terminate wiring for two- or three-wire devices
- I/O modules – contains the bus interface and circuitry that is needed to perform specific functions that are related to your application
- End cap – basically a dust cap for the last module in a rack

FLEX 5000 I/O System Power

ATTENTION: Power to this equipment and all connected I/O must be supplied from a source that is isolated from Mains power via an approved isolating transformer that is constructed with basic insulation.

FLEX 5000 I/O SA Field-Side Power

- FLEX 5000 I/O modules use terminal base (TB) assemblies to connect field-side wiring.
- SA field-side power source is connected to the terminal base (TB) assemblies via SA power terminals.
- If you are using DC voltage for SA power, you must limit the SA field-side power source to 10 A, max, at 18…32V DC.

- Confirm that the external module power supply is adequately sized for the total SA field-side power current draw in the module.

  For example, if the total module power current draw, including current inrush requirements, is 5 A, you can use a module power supply that is limited to 5 A.
- You must use SELV-listed power supplies for SA field-side power if there are ports on the DTE/DCE device that are accessible while the equipment is powered on.
- You must use SELV-listed power supplies for module power if:
  - there are Functional Safety modules that are connected to the FLEX 5000 I/O system.
  - there are ports on the equipment that are accessible while the equipment is powered on.
- Not all power supplies are certified for use in all applications, for example, nonhazardous and hazardous environments.

IMPORTANT We recommend that you use separate external power supplies for the adapter and the adjacent terminal base. This practice can prevent unintended consequences that can result if you use one supply.

For more information, see the publications that are listed in Additional Resources on page 7.

Before You Begin

Before you use your serial module, you must complete the following:

a. Install a FLEX 5000 EtherNet/IP adapter.
b. Install the FLEX 5000 I/O modules to the right of the adapter.
c. Install an EtherNet/IP network.
d. Install the Logix 5000 controller that accesses the FLEX 5000 I/O modules over an EtherNet/IP network.

Make sure that you have enough FLEX 5000 terminal base (TB) assemblies to satisfy your application needs. For more information, see the FLEX 5000 Terminal Base Assembly Modules Installation Instructions, publication 5094-IN010.

IMPORTANT Terminal bases are not included with your module and are not available for purchase. A terminal base consists of a mounting base (MB) and removable terminal block (RTB). You must purchase MBs and RTBs separately and assemble them together.

For adapter information, see the FLEX 5000 EtherNet/IP Adapters with RJ45 Ports Installation Instructions, publication 5094-IN001, and the FLEX 5000 EtherNet/IP Adapters with SFP Support Installation Instructions, publication 5094-IN002.

Module Overview

Figure 2 shows the parts of a FLEX 5000 serial module.

Figure 2 - FLEX 5000 Serial Module

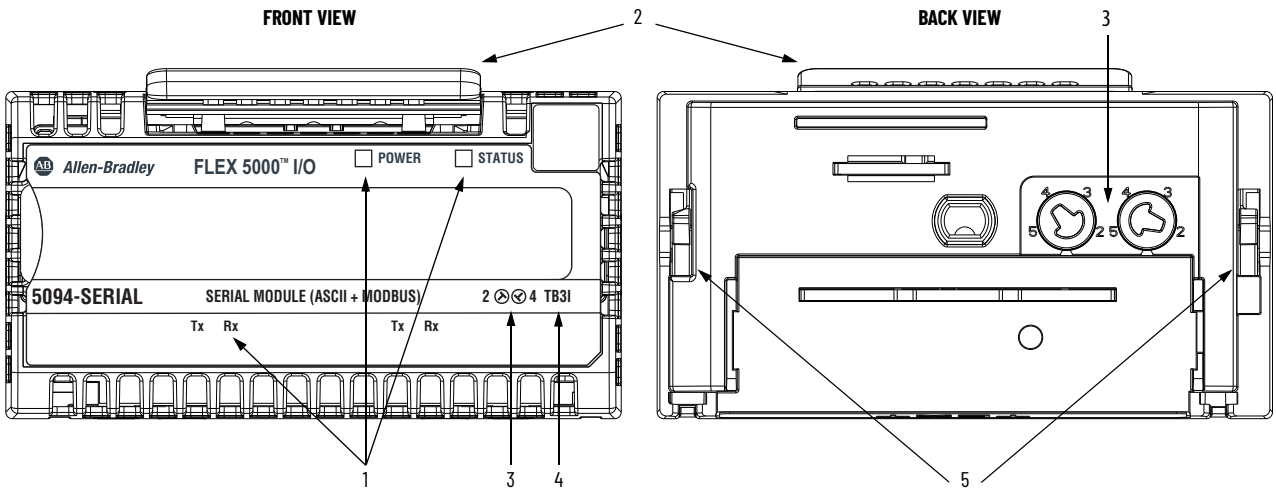


Table 2 - FLEX 5000 Serial Module Parts

Item	Description
1	**Status indicators** - Displays the status of communication, module health, and input/output devices. Indicators help with troubleshooting anomalies
2	**Release lever** - Disengages the latching hooks to allow removal of the module from the terminal base assembly
3	**Module keying** - Indicates the keying position the terminal base assembly must be configured to before installing the module
4	**Terminal base** - Indicates the type of terminal base assembly to use with the module
5	**Latching hooks** - Securely installs FLEX 5000 modules on the terminal base assembly

Configuration via Logix Designer Application

You must create a Logix Designer application project for the Logix 5000 controller that owns the FLEX 5000 serial modules. The project includes module configuration data for the FLEX 5000 serial modules.

The Logix Designer application transfers the project to the owner-controller during the program download. Data is then transferred to the FLEX 5000 serial modules over the EtherNet/IP network.

The FLEX 5000 serial modules can operate immediately after receiving the configuration data.

Connections

During module configuration, you must define the module. Among the Module Definition parameters, you must choose a connection type for the module. A connection is a real-time data transfer link between the owner-controller and the module that occupies the slot that the configuration references.

When you download module configuration to a controller, the controller attempts to establish a connection to each module in the configuration.

Because part of module configuration includes a slot in the FLEX 5000 I/O system, the owner-controller checks for the presence of a module there. If a module is detected, the owner-controller sends the configuration. One of the following occurs:

- If the configuration is appropriate to the module detected, a connection is made and operation begins.
- If the configuration is not appropriate to the module detected, the data is rejected and the Logix Designer application indicates that an error occurred.

  The configuration can be inappropriate for many reasons. For example, a mismatch in electronic keying that helps prevents normal operation.

The owner-controller monitors its connection with a module. Any break in the connection, for example, the loss of power to the FLEX 5000 I/O system, causes a fault. The Logix Designer application monitors the fault status tags to indicate when a fault occurs on a module.

Multiple Connections to One Serial Module

Unlike other FLEX 5000 I/O modules that make one connection to the owner-controller, the FLEX 5000 serial module can have multiple connections based on how the module channels are configured.

The serial module is required to use the Data connection type in the Module Definition dialog box. However, the module has two channels that you must configure independently of each other. You can disable a channel or choose a mode – Generic ASCII, Modbus Master, or Modbus Slave.

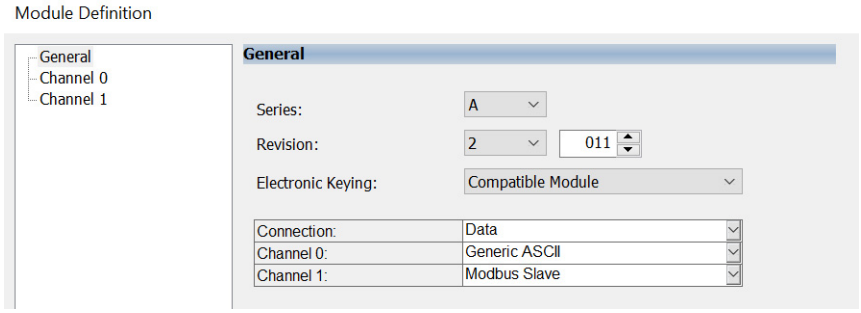
The combination of channel configuration choices determines the number of connections that are made between the owner-controller and the Serial module.

Table 3 - Modes for FLEX 5000 Serial Modules

Selected Protocol	Description
Generic ASII	The module returns the following to the owner-controller: • General fault data • Input Data • Output Data
Modbus Master	The module returns the following to the owner-controller: • Generic fault data • Last executed command data • Input/Output • Operation status
Modbus Slave	The module returns the following to the owner-controller: • Generic fault • Slave input data • Slave output data

Consider the following examples:

- Channel 0 = Generic ASCII, Channel 1 = Modbus Slave - Two connections are made between the owner-controller and the module.



- Channel 0 = Generic ASCII, Channel 1 = Modbus Master - Three connections are made between the owner-controller and the module

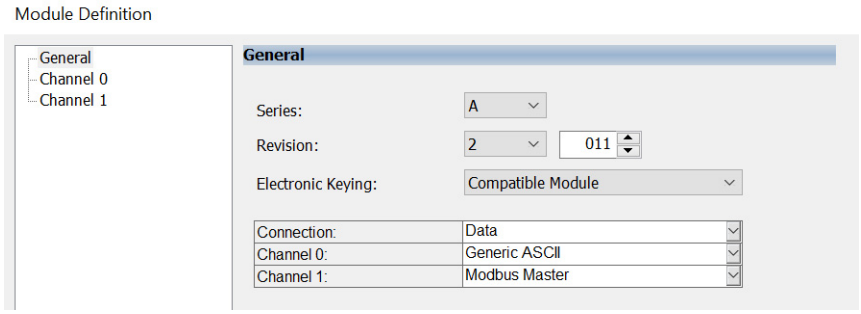


Table 4 shows the total number of connections for all protocol combinations.

Table 4 - Connections Available for Different Protocol Combinations

		CH0 Protocol Choice			
		Disabled	**Generic ASCII**	**Modbus Slave**	**Modbus Master**
CH1 Protocol Choice	**Disabled**	0 connection	1 connection	1 connection	2 connections
	Generic ASCII	1 connection	2 connections	2 connections	3 connections
	Modbus Slave	1 connection	2 connections	2 connections	3 connections
	Modbus Master	2 connections(1)	3 connections	3 connections	4 connections

(1) The Modbus Master can have 1 or 2 connections depending on the command list configuration.

Protected Operations

To maintain the secure operation of your FLEX 5000 serial module, operations that can disrupt module operation are restricted based on the module operating mode. Table 5 describes the restrictions.

Table 5 - Protected Operations on FLEX 5000 Serial Modules

Current Module Operation	Activity						
	Firmware Update Request	Module Reset Request	Connection Request	Configuration Change	Connection or Data Format Change	Electronic Keying Change	RPI Change
Connection not running	Accepted						
Connection running	Rejected		Rejected	Accepted(1)	Not allowed(2)	Accepted(3)	
Firmware update is in process	Rejected						

(1) Configuration changes are made in the Module Properties dialog box and you click Apply.
(2) The difference between Rejected and Not allowed is that rejected activities can be attempted in the Logix Designer application but do not take effect. The activities that are not allowed, that is, attempts to change the Connection or Data Format used, do not occur in the Logix Designer application.
For example, if you attempt to reset a module that is connected to the owner-controller, the Logix Designer application executes the request and alerts you that it was rejected. If you attempt to change the data format on a module that is connected to an owner-controller, the Logix Designer application does not execute the attempted change. The application only alerts you that the change is not allowed. In the case, if the change is attempted online, the Module Definition dialog box field that changes the data format is disabled.
(3) The change occurs after the connection is closed and reopened. You can close and reopen the connection in the following ways:
- Change the project while it is offline and download the updated project before going online again.
- Change the project while it is online and click Apply or OK in the Module Properties dialog box. In this case, a dialog box alerts you of the ramifications before the change is made.

Notes:

FLEX 5000 Serial Module Features

Purpose of the Module

The FLEX 5000 serial module features two different communication modes to connect to serial devices in various communication mediums. For example, the RS-232C, RS-422, or RS-485.

General Module Features

The FLEX 5000 serial module supports the following module-wide features:

- Software Configurable
- Requested Packet Interval
- Fault and Status Reporting
- Module Inhibiting
- Electronic Keying
- Status Indicators
- Module Firmware

Software Configurable

You use the Logix Designer application to configure the module, monitor system operation, and troubleshoot issues. You can also use the Logix Designer application to retrieve the following information from any module in the system:

- Serial number
- Firmware revision information
- Product code
- Vendor
- Error and fault information
- Diagnostic information

By minimizing the need for tasks, such as setting hardware switches and jumpers, the software makes module configuration easier and more reliable.

Requested Packet Interval

The Requested Packet Interval (RPI) is a configurable parameter that defines a specific rate at which data is exchanged between the owner-controller and the module.

You set the RPI value during initial module configuration and can adjust it as necessary after module operation has begun.

IMPORTANT If you change the RPI while the project is online, the connection to the module is closed and reopened in one of the following ways:

- You inhibit the connection to the module, change the RPI value, and uninhibit the connection.
- You change the RPI value. In this case, the connection is closed and reopened immediately after you apply the change to the module configuration.

Table 6 - RPI Range

Protocol	Setting Range of RPI
Generic ASCII	2...750 ms (by 0.1 ms increments)
Modbus Master	6...750 ms (by 0.1 ms increments)
Modbus Slave	6...750 ms (by 0.1 ms increments)

IMPORTANT If the RPI is not a multiple of 0.1 ms, round the number down to the closest multiple values of 0.1 ms. For example, if the RPI is 2.37 ms, round to 2.3 ms.

For more information on guidelines for specifying RPI rates, see the Logix 5000 Controllers Design Considerations Reference Manual, publication 1756-RM094.

Fault and Status Reporting

The FLEX 5000 serial modules report fault and status data along with channel data. Fault and status data is reported in the following ways:

- Logix Designer application
- Module status indicators

For more information on fault reporting, see Chapter 5, Troubleshoot Your Module.

Module Inhibiting

Module inhibiting lets you indefinitely suspend a connection between an owner-controller and a serial module without removing the module from the configuration. This process lets you temporarily disable a module, such as to perform maintenance.

You can use module inhibiting in the following ways:

- You write a configuration for an I/O module but inhibit the module to help prevent it from communicating with the owner-controller. The owner does not establish a connection and the configuration is not sent to the module until the connection is uninhibited.
- In your application, a controller already owns a module, has downloaded the configuration to the module, and is exchanging data over the connection between the devices.

  In this case, you can inhibit the module and the connection to the module does not exist.

You can use module inhibiting in these instances:

- You want to update a serial module, for example, update the module firmware revision. Use the following procedure.
  a. Inhibit the module.
  b. Perform the update.
  c. Uninhibit the module.
- You use a program that includes a module that you do not physically possess yet. You do not want the controller to look for a module that does not yet exist. In this case, you can inhibit the module in your program until it physically resides in the proper slot.

To see where to inhibit a FLEX 5000 serial module, see page 46.

Electronic Keying

Electronic Keying reduces the possibility that you use the wrong device in a control system. It compares the device that is defined in your project to the installed device. If keying fails, a fault occurs. These attributes are compared.

Attribute	Description
Vendor	The device manufacturer.
Device Type	The general type of the product, for example, serial module.
Product Code	The specific type of the product. The Product Code maps to a catalog number.
Major Revision	A number that represents the functional capabilities of a device.
Minor Revision	A number that represents behavior changes in the device.

The following Electronic Keying options are available.

Keying Option	Description
Compatible Module	Lets the installed device accept the key of the device that is defined in the project when the installed device can emulate the defined device. With Compatible Module, you can typically replace a device with another device that has the following characteristics: • Same catalog number • Same or higher Major Revision • Minor Revision as follows: – If the Major Revision is the same, the Minor Revision must be the same or higher. – If the Major Revision is higher, the Minor Revision can be any number. • Non-XT and XT version as follows: – You can use an XT version of the module in place of a non-XT module. – You **cannot** use a non-XT version of the module in place of an XT module.
Disable Keying	Indicates that the keying attributes are not considered when attempting to communicate with a device. With Disable Keying, communication can occur with a device other than the type specified in the project. **ATTENTION**: Be extremely cautious when using Disable Keying; if used incorrectly, this option can lead to personal injury or death, property damage, or economic loss. We **strongly recommend** that you **do not use** Disable Keying. If you use Disable Keying, you must take full responsibility for understanding whether the device being used can fulfill the functional requirements of the application.
Exact Match	Indicates that all keying attributes must match to establish communication. If any attribute does not match precisely, communication with the device does not occur.

Carefully consider the implications of each keying option when selecting one.

IMPORTANT	Changing Electronic Keying parameters online interrupts connections to the device and any devices that are connected through the device. Connections from other controllers can also be broken. If an I/O connection to a device is interrupted, the result can be a loss of data.

For more detailed information on Electronic Keying, see Electronic Keying in Logix 5000 Control Systems Application Technique, publication LOGIX-AT001.

Status Indicators

Each FLEX 5000 serial module has a status indicator on the front of the module that lets you check the health and operational status of a module. The status indicator displays vary for each module.

For more information on status indicators, see Chapter 4, Troubleshoot Your Module.

Module Firmware

The FLEX 5000 serial modules are manufactured with module firmware installed. If updated module firmware revisions are available in the future, you can update the firmware.

Updated firmware revisions are made available for various reasons, for example, to correct an anomaly that existed in previous module firmware revisions.

You access updated firmware files at rok.auto/pcdc. At the Product Compatibility and Download Center (PCDC), use the module catalog number to check for firmware updates. If the catalog number is not available, no updates exist.

Verify that the firmware revision of the FLEX 5000 serial modules that you use is correct before commissioning the system.

IMPORTANT Only download firmware and access product release notes from the Rockwell Automation PCDC.
Do not download firmware from non-Rockwell Automation sites.

Common Module Functions

The module supports the following terms, definitions, and parameters. To configure these parameters, see page 48.

Table 7 - Common Module Functions

Function	Definition	Available Options
Baud Rate	The communication speed of each channel.	• 9600 • 1200 • 2400 • 4800 • 19200 (default) • 38400 • 57600 • 115200
Data Bits	The number of data bits are used to represent one character of data.	• 7 bit • 8 bit (default)
Parity	Sets the parity of transmitted data for error detection. It is created with data files and used to check data integrity and help with data recovery.	• None (default) • Even • Odd
Stop Bits	This parameter sets the number of stop bits for each data value sent.	• 1 (default) • 2
Serial Media	Type of media that is connected to the channels communication ports.	• RS-232C (default) • RS-422 • RS-485
Duplex	The type of communication that is used by each channel.	• No Handshake (default) • Full-duplex • Half-duplex
Continuous Carrier	A carrier frequency that is transmitted even when data is not being sent. Continuous carrier is selected if you want to use it with half-duplex communication. The checkbox is unavailable if you have chosen something other than half-duplex communication, or if you have chosen Master as your protocol. The default option is cleared when enabled.	• On • Off (default)
RTS Send Delay	Enter the time (x20 ms) to delay transmitting the first character of a message after turning on the RTS line. The default value is 0.	• 0…255 (default = 0)
RTS Off Delay	Enter the time (x20 ms) to delay turning off the RTS line after the last character has been transmitted. The default value is 0.	• 0…255 (default = 0)
DCD Wait Delay	The number of seconds to wait before lowering the DCD modem line. When DCD is high, the controller is in the middle of transmitting data. This delay may be needed because of the latency in the sending radio transmissions.	• 0…255 (default = 0)

Control Line Menu

When you are required to connect to a dial-up modem, see Table 8, which explains the duplex setting in the serial port.

Table 8 - Control Line Menu

Modem	Duplex Status	Controller	Your Function Choice	Continuous Carrier
Not using a modem	–		No handshaking	–
Using a modem	Modems in a point-to-point link are full-duplex.	–	Full-duplex	
	Master modem is a full-duplex while slave modem is half-duplex.	Master controller	Full-duplex	
	–	Slave controller	Half-duplex	Select the continuous carrier checkbox.
	All modems in a system are half-duplex.	–	Half-duplex	Clear the continuous carrier checkbox (default).

Data Exchange

Generic ASCII Data Exchange

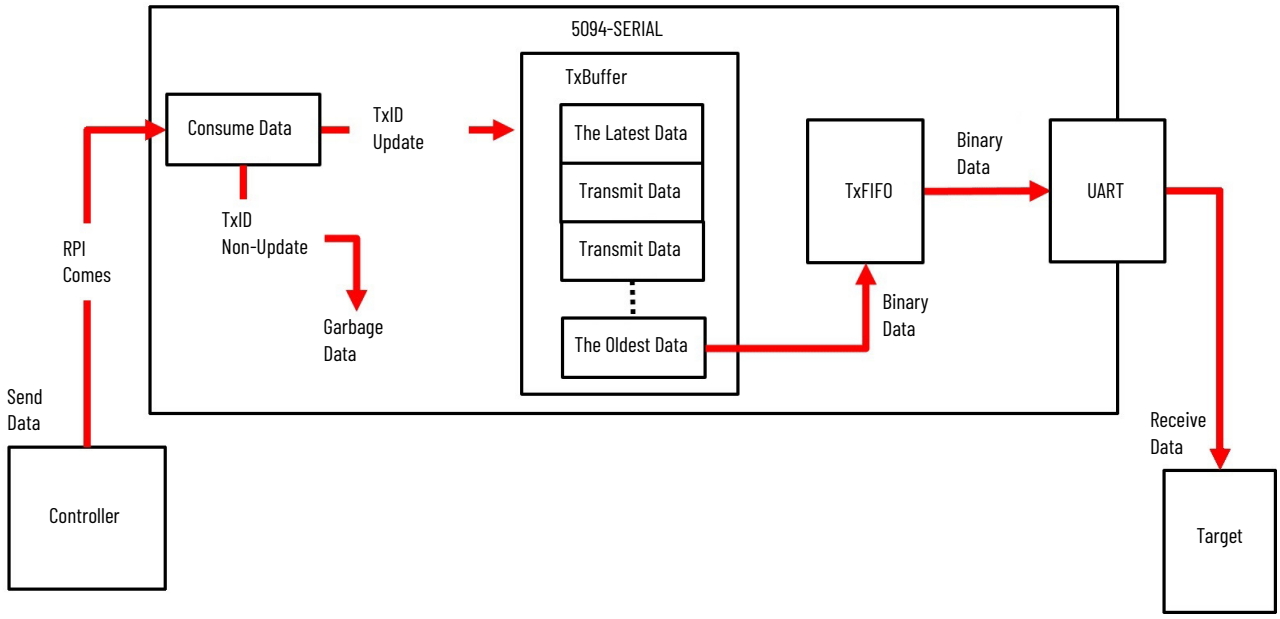
The following illustrations show the different data exchanges using the serial module.

Data Sent with the Serial Port

Data is sent out of the serial port using the following steps:

1. The controller sends out the consumed packet with an incremented TxID.
2. The serial module puts the packet into the TxBuffer.
3. The serial module takes the oldest data from the TxBuffer and sends it out to the target.

Figure 3 - Data Sent out to the Serial Port



Generic ASCII Transmit Functions

In Generic ASCII mode, you can define any kind of data to be transmitted to communicate to serial devices. Some examples are barcode scanners, dial up modems, serial printers, temperature controllers, and so on.

Table 9 - Generic ASCII Transmit Functions

Parameter	Definition	Available Options
Swap Mode	Select whether swapping will be done or what swapping method is to be used before the module sends output data to the Serial Port or after the module receives input data from the Serial Port.	• No Change (default) • Word Swap - After executing a word swap on 32-bit hex value 11112222, the result is 22221111. • Byte Swap - After executing a byte swap on 32-bit hex value 11223344, the result is 22114433. • Word and Byte Swap - After executing a word and byte swap on 32-bit hex value 11223344 the result is 44332211.
Termination Mode/Delimiter	Selecting the termination mode of each channel. Choose to ignore or include the delimiter.	• Ignore End Delimiter (default) - Transmits packet based on number of bytes specified only. • Exclude Delimiter - Determines end of data when it finds the Termination Delimiter characters, but doesn't transmit the Termination Delimiter bytes with the packet. • Include Delimiter - Determines end of data by Termination Delimiters and transmits them.
Termination Delimiter 1	Configure the channel's termination delimiter.	• 7 bit $00...$7F • 8 bit $00...$FF
Termination Delimiter 2	Configure the channel's termination delimiter.	• 7 bit $00...$7F • 8 bit $00...$FF ($FF = disabled)

Generic ASCII Transmit Methods

To transmit the ASCII packet based on the number of characters, follow these steps:

1. For this method, configure Termination Mode for Ignore End Delimiter.
2. After copying the characters into the ASCII.TxData output tag array, write the number of characters into the ASCII.TxDataLength output tag, then increment the ASCII.TxID output tag.

To transmit the ASCII packet based on the termination delimiter characters, follow these steps:

1. For this method, configure Termination Mode for either Include or Exclude.
2. After copying the characters into the ASCII.TxData output tag array, copy the two configured termination delimiter characters as the next two characters in the array, then increment the ASCII.TxID output tag. While doing this, keep the TxDataLength output tag at 0.

Include transmits the packet with termination delimiter characters, and exclude does not.

Generic ASCII Receive Functions

Table 10 - Generic ASCII Receive Functions

Parameter	Definition	Available Options
Swap Mode	Select whether swapping is done or what swapping method is to be used before the module sends output data to the Serial Port or after the module receives input data from the Serial Port.	• No Change (default) • Word Swap - After executing a word swap on 32-bit hex value 11112222, the result is 22221111. • Byte Swap - After executing a byte swap on 32-bit hex value 11223344, the result is 22114433. • Word and Byte Swap - After executing a word and byte swap on 32-bit hex value 11223344 the result is 44332211.
Handshake Mode	Determines how the serial module passes the received data from the serial port to the controller.	• Master/Slave (default) - User logic must increment the ASCII.RxID output tag to receive the next packet of data into the ASCII.RxData input tag. • Immediate - ASCII.RxID input tag increments automatically indicating that the next packet of received data is available in the ASCII.RxData input tag.
Message Timeout	The timer resets every time that the module receives a new byte from the Serial Port. If a Timeout occurs, the Non-Delimited Flag is set, and Received Data is regarded as a new record to produce.	• 0 = Disabled (default) • 1...32,767 ms
Pad Character	Character that is used to fill the remainder of the ASCII.RxData array after the end of the received packet characters. Padding range is I.RxData[RxDataLength] to I.RxData[Read Buffer Size].	• 7 bit $00...$7F • 8 bit $00...$FF
Start Mode/Delimiter	Select the usage of the Start Delimiter in the communication frame. Choose to ignore, exclude, or include the delimiter.	• Ignore Start Delimiter (default) - Start of received packet not based on the Start Delimiter. • Exclude - Start of packet is determined based on the Start Delimiter character, but this byte is not included in the ASCII.RxData input tag. • Include - Start Delimiter is always the first byte in the ASCII.RxData input tag.
Start Delimiter	Beginning of the message.	• 7 bit $00...$7F – Recommendation: When Delete Mode is enabled, do not configure Start Delimiter to DEL character. • 8 bit $00...$FF
Termination Mode/Delimiter	Selecting the termination mode of each channel. Choose to ignore or include the delimiter.	• Ignore End Delimiter (default) - Transmits packet based on number of bytes specified only. • Exclude Delimiter - Determines end of data when it finds the Termination Delimiter characters, but doesn't transmit the Termination Delimiter bytes with the packet. • Include Delimiter - Determines end of data by Termination Delimiters and transmits them.
Termination Delimiter 1	Configure the channel's termination delimiter.	• 7 bit $00...$7F – Recommendation: When Delete Mode is enabled, do not configure Start Delimiter to DEL character. • 8 bit $00...$FF
Termination Delimiter 2	Configure the channel's termination delimiter.	• 7 bit $00...$7F • 8 bit $00...$FF ($FF = disabled)
XON/XOFF	Selecting the flow control of each channel. Enables software handshaking.	• 0 = Disable (default) • 1 = Enable
Echo Mode	The module sends all bytes received from Serial Port immediately to the serial port by 1 byte, and sends produced data to the controller. Enables retransmission of all received characters.	• 0 = Disable (default) • 1 = Enable
Delete Mode	If the mode is ignored, it is handled as regular ASCII data. Choosing CRT means that the module does not send previous data and is replaced by three characters. Choosing printer means that the module does not send previous data and is replaced by one character.	• 0 = Ignore (default) - echoes DEL character same as any other character • 1 = CRT - receives DEL character, echoes backspace, space, backspace • 2 = Printer - receives DEL character, echoes '/' followed by previous character
Read Buffer Size	Max buffer length supported.	• 1...256 (default = 256)

General ASCII Receive Methods

When receiving the ASCII packet based on a fixed number of characters, the number of characters is configured in the Read Buffer Size parameter.

- For this method, configure Termination Mode for Ignore End Delimiter.

When receiving the ASCII packed based on timeout since the last character was received, the timeout in milliseconds is configured in the Message Timeout parameter.

- For this method, configure Termination Mode for Ignore End Delimiter.

When receiving the ASCII packed based on Termination delimiters at the end of the packet, if termination mode is “Include”, then the termination bytes remain appended to the end of the data copied into the ASCII.RxData input tag. If the termination mode is “Exclude”, then the termination bytes are stripped off.

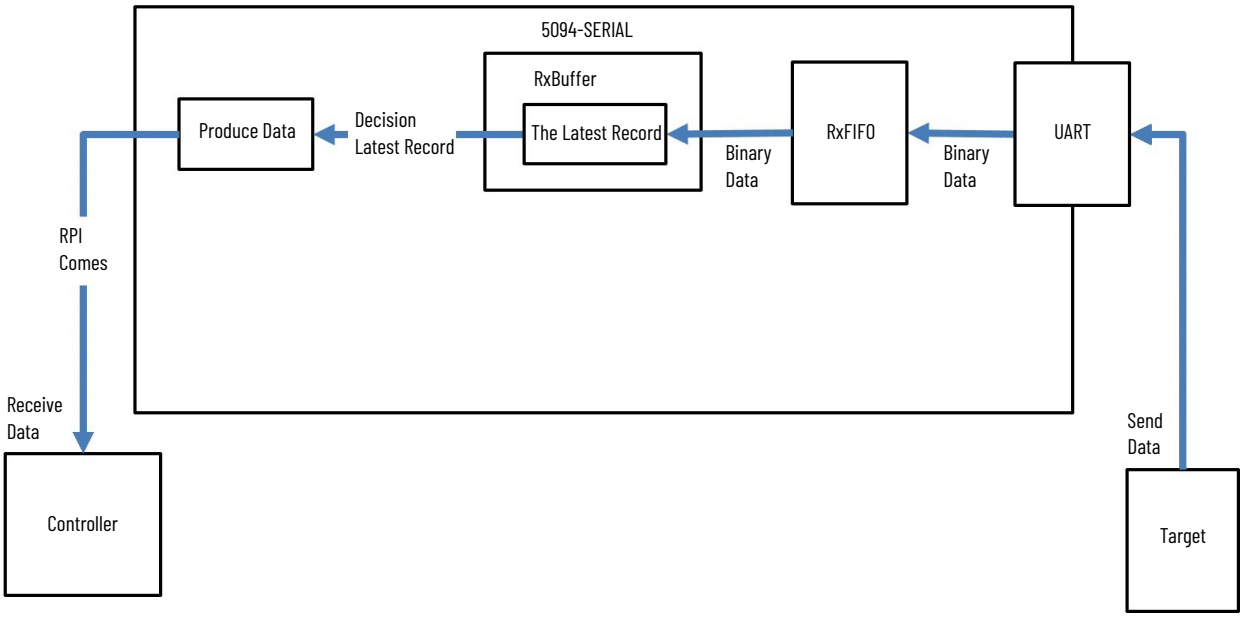
For Generic ASCII Sample Code, see page 80.

Data Received from Serial Port in Immediate Mode

After the serial port receives data, it processes the data using the following steps.

1. The serial module receives the packet.
2. If any of the following conditions occur, the serial module creates a record.
   a. Message Timeout timer expires.
   b. The number of received bytes equals the configured Read Buffer Size.
   c. Termination Delimiter bytes were received.
3. The serial module copies the data into the ASCII.RxData input tag, copies the number of characters that are received into the ASCII.RxDataLength input tag, increments the ASCII.RxID input tag and sends it to the controller.

Figure 4 - Data Received from the Serial Port in Immediate Mode

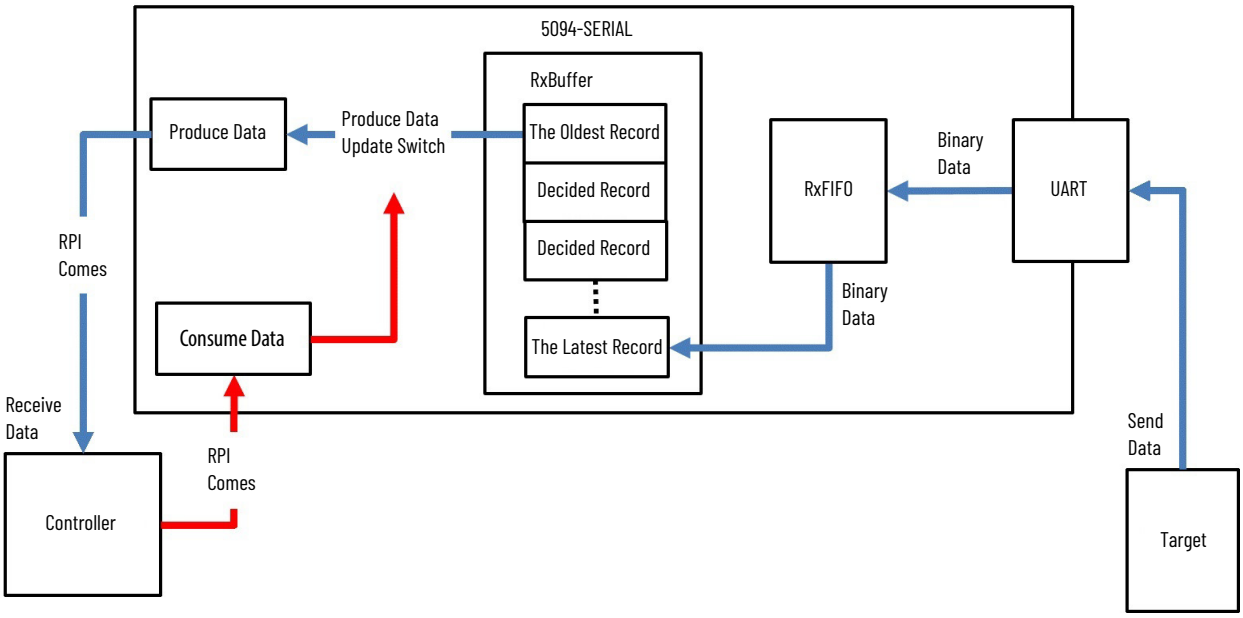


Data Received from Serial Port in Master/Slave Handshake Mode

The serial port works in a handshake mode using the following steps.

1. The serial module receives the packet.
2. If any of the following conditions occur, the serial module creates a record.
   a. The Message Timeout timer expires.
   b. The number of received bytes equals the configured Read Buffer Size.
   c. Termination Delimiter bytes were received.
3. The record is added into the RxBuffer.
4. Once the RxID (Consume Tag) is incremented by the user logic, the serial module takes the oldest record from RxBuffer, copies the data into the ASCII.RxData input tag, copies the number of characters received into the ASCII.RxDataLength input tag, and sends it to the controller.

Figure 5 - Serial Port Handshake Mode



IMPORTANT If Master/Slave Handshake is selected, dispose additional receiving data from the serial port for saved data in the receiving buffer of the firmware.

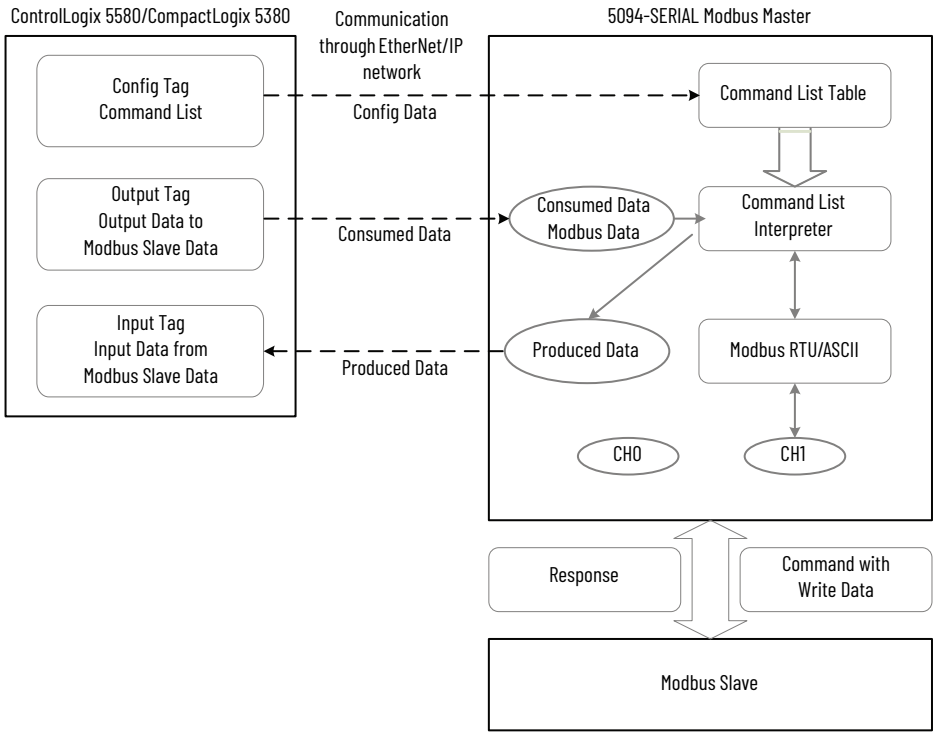
Modbus Master Data Exchange

For the Modbus Master data exchange, the following definitions apply:

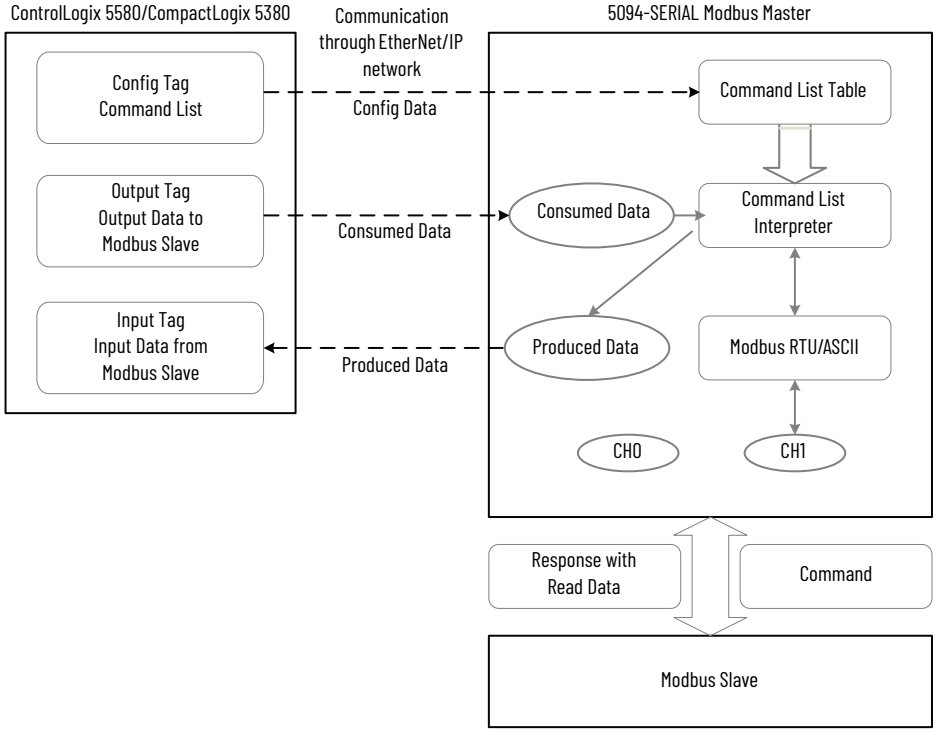
- Write – the Modbus Master writes data to the Modbus Slave.
- Read – the Modbus Master reads data from the Modbus Slave.

The FLEX 5000 serial module can get Modbus Data from Produced/Consumed Data command, every RPI.

Modbus Master Write Command



Modbus Master Read Command

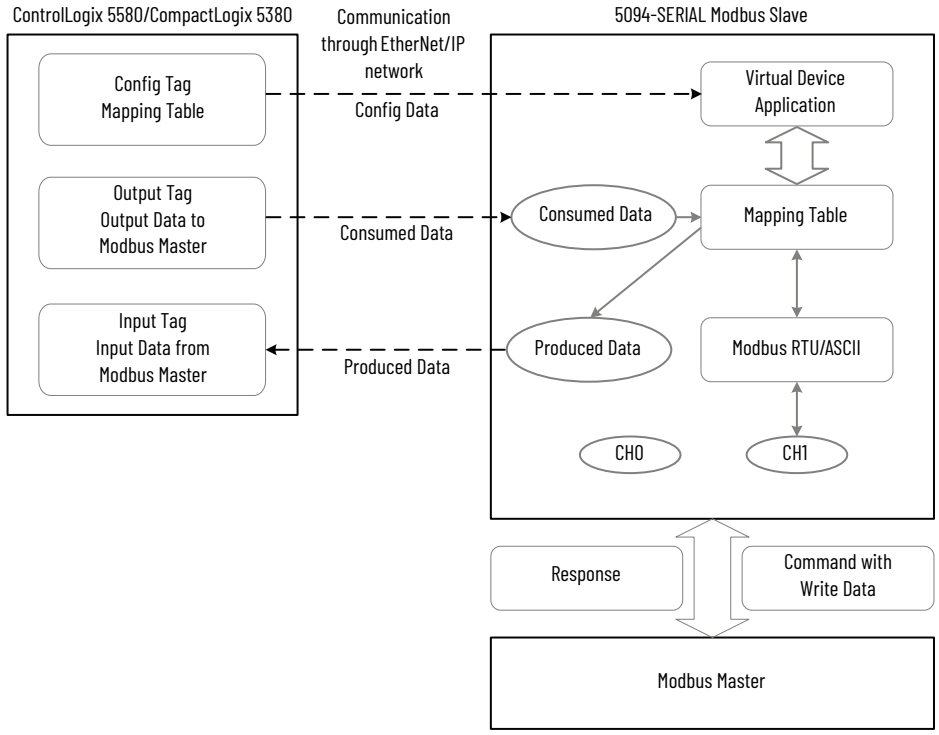


Modbus Slave Data Exchange

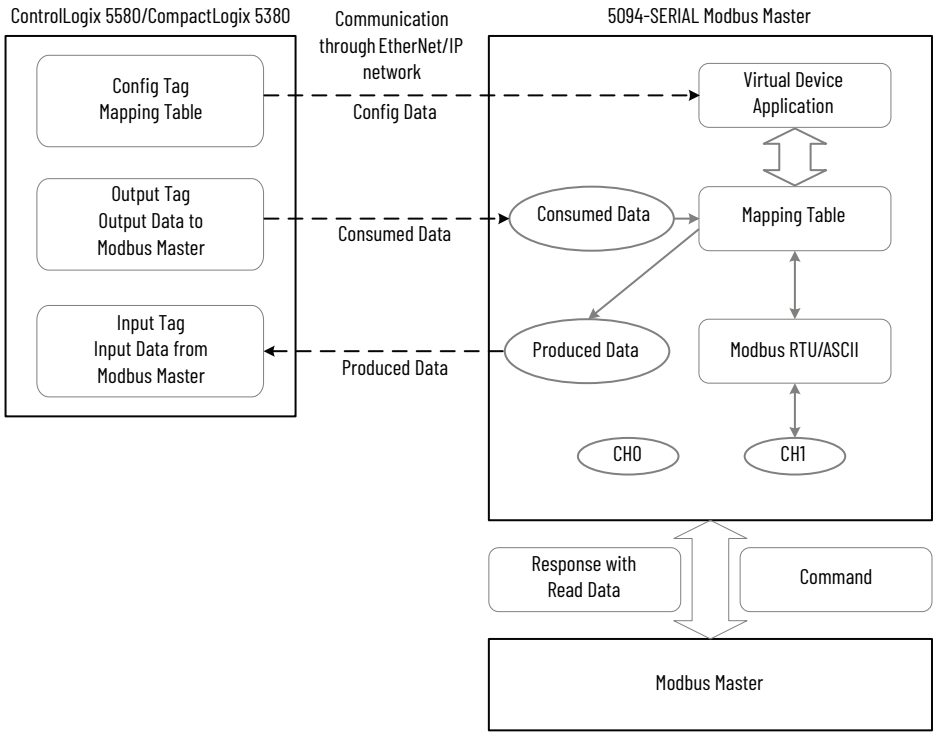
For the Modbus Slave data exchange, the following definitions apply:

- Write – the Controller and Modbus Master can write data to the Modbus Slave.
- Read – the Modbus Master reads data from the Modbus Slave.

Modbus Slave Write Command



Modbus Slave Read Command



Modbus Master Functions

In Modbus mode, the serial module supports both Modbus ASCII and Modbus RTU to connect to Modbus enabled devices like flowmeters, power meters or remote RTU talking on Modbus RTU/ASCII.

Table 11 - Generic ASCII Receive Functions

Parameter	Definition	Available Options
Modbus Format	Select communication method of Modbus of each channel.	• 0 = RTU (default) • 1 = ASCII – Intervals of up to one second may elapse between characters within the message. Unless the user has configured a longer timeout, an interval greater than one second means that an error has occurred.
Response Timeout	The Maximum Delay time of each channel until it receives a response for Modbus Master Command from Modbus Slave. When the maximum time has passed, the Modbus Master determines the Modbus Slave did not respond.	• 0...3600000 (x1 ms) (default = 200)
Broadcast Pause	The Wait time of each channel until the Modbus Slave finishes processing according to broadcasted command from the Modbus Master. The Modbus Master does not send the next command until this time has passed.	• 0...3600000 (x1 ms) (default = 200)
Inter-frame Timeout	Maximum delay time to receive Data of each channel.	• 0 is not a valid value. Use Table 12 and Table 13 to determine the minimum value.
Retry Count	This parameter specifies the number of times a command is retried if it fails. If the Master Port does not receive a response after the last retry, the Slave devices communication is suspended on the port for Error Delay Counter scans.	• 0...127 (default = 0)

Table 12 - Inter-frame Timeout Minimum Values (10 Bit)

Baud Rate	Default Value	1 Byte (10 Bit)	3.5t (10 Bit)	Legal Range (10 Bit)
1200	58000	8333.333	29166.667	29000…65535000 (µs)
2400	28000	4166.667	14583.333	14000…65535000 (µs)
4800	14400	2083.333	7291.667	7200…65535000 (µs)
9600	7200	1041.667	3645.833	3600…65535000 (µs)
19200	3500	520.833	1822.917	1750…65535000 (µs)
38400	3500	260.417	911.458	1750…65535000 (µs)
57600	3500	173.611	607.639	1750…65535000 (µs)
115200	3500	86.806	303.819	1750…65535000 (µs)

Table 13 - Inter-frame Timeout Minimum Values (11 Bit)

Baud Rate	Default Value	1 Byte (11 Bit)	3.5t (11 Bit)	Legal Range (11 Bit)
1200	64000	9166.667	32083.333	29000…65535000 (µs)
2400	32000	4583.333	16041.667	14000…65535000 (µs)
4800	16000	2291.667	8020.833	7200…65535000 (µs)
9600	8000	1145.833	4010.417	3600…65535000 (µs)
19200	3500	572.917	2005.208	1750…65535000 (µs)
38400	3500	286.458	1002.604	1750…65535000 (µs)
57600	3500	190.972	668.403	1750…65535000 (µs)
115200	3500	95.486	334.201	1750…65535000 (µs)

Modbus Slave Functions

Table 14 - Modbus Slave Configuration Parameters

Parameter	Definition	Available Options
Modbus Format	Select communication method of Modbus of each channel.	• 0 = RTU (default) • 1 = ASCII – Intervals of up to one second may elapse between characters within the message. Unless the user has configured a longer timeout, an interval greater than 1 second means that an error has occurred.
Node Address	Numbers to identify all modules that are connected to each channel. You must set a number not equal to 0.	• 0…247 (default = 1)
Inter-frame Timeout	Maximum delay time to receive Data of each channel.	• 0 is not a valid value. Use Table 12 and Table 13 to determine the minimum value.
Retry Count	This parameter specifies the number of times a command is retried if it fails. If the Master Port does not receive a response after the last retry, the Slave devices communication is suspended on the port for Error Delay Counter scans.	• 0…127 (default = 0)

For Modbus Master and Modbus Slave sample code, see page 82 and page 82.

Notes:

Configure FLEX 5000 Serial Modules

This chapter describes how to configure your FLEX 5000 serial module in a Logix Designer application project. You can use the default module configuration or edit the module configuration.

IMPORTANT Consider the following:

- You must use the Studio 5000 Logix Designer application, version 31 or later, to configure the FLEX 5000 serial module.
- You must install an Add-on Profile to use the serial module. To find the Add-on Profile, go to rok.auto/pcdc.

When a controller establishes a connection to a 5094-SERIAL module, it uses a class 3 connection. We recommend that you reserve one class 3 connection on the FELX 5000 EtherNet/IP adapter to establish a connection to the module. Otherwise, you can encounter a "Connection Request Error: Module connection limit exceeded" error.

Before You Begin

You must complete the following tasks before you can configure the module:

1. Create a Logix Designer application project.
2. Add a FLEX 5000 EtherNet/IP adapter to the project.

   For more information on how to add a FLEX 5000 EtherNet/IP adapter to a Logix Designer application project, see the FLEX 5000 EtherNet/IP Adapter User Manual, publication 5094-UM005.

Create a New Module

After you create a Logix Designer application project and add a FLEX 5000 EtherNet/IP adapter to the project, you can use the following methods to add modules to the project.

- Discover Modules
- New Module

Discover Modules

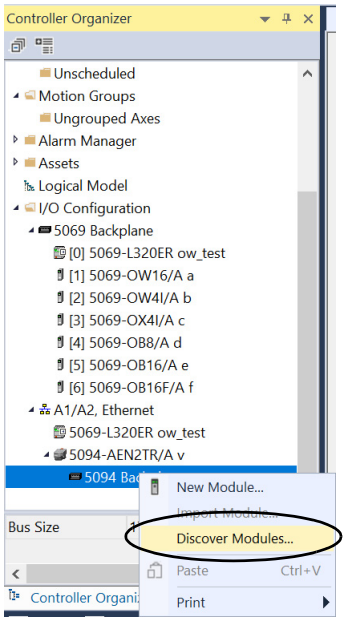
To use the Discover Modules method with FLEX 5000 I/O modules, complete these steps.

1. Go online with your Logix Designer application.

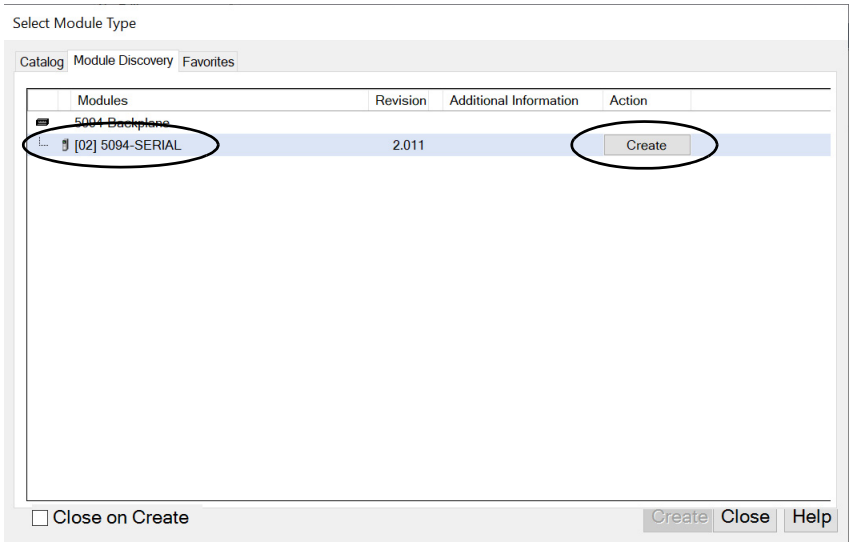
   The project must include a FLEX 5000 EtherNet/IP adapter.

2. Right-click the FLEX 5000 EtherNet/IP adapter and choose Discover Modules.

   The Logix Designer application automatically detects available modules that are connected to the backplane.

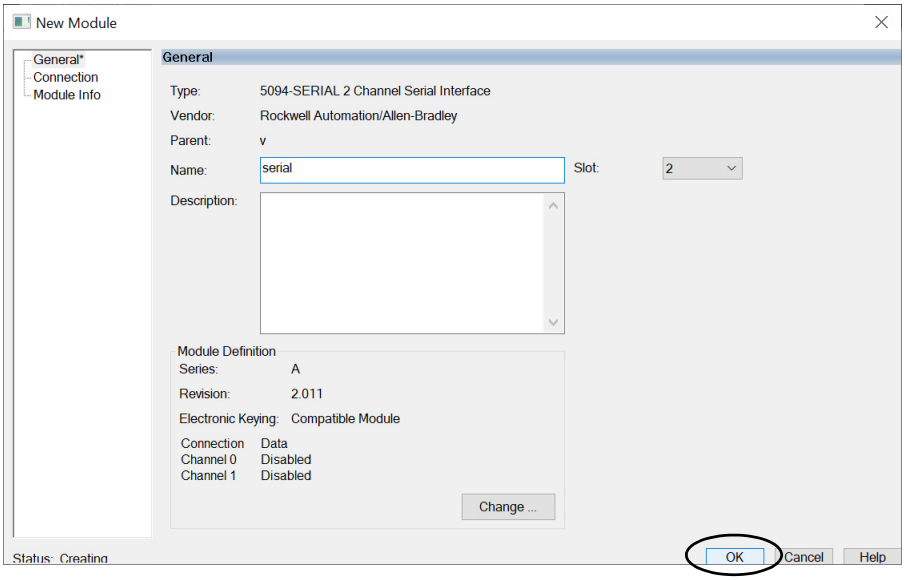


3. At the Select Module Type window, click Create to add the discovered module to your project.

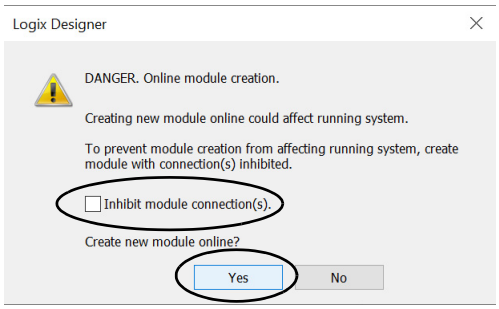


   The New Module dialog box appears with a list of categories on the left side. The number and type of categories varies by module type.

4. You can click OK to use the default configuration as shown or edit the module configuration. The rest of this chapter describes how to edit module configuration categories.



5. At the warning dialog box, make sure that Inhibit module connection is selected and click Yes.



6. Close the Select Module Type dialog box.

To add additional I/O modules with this method, complete one of the following:

- If you cleared the Close on Create checkbox when you created the first I/O module, repeat steps 3...6.
- If you did not clear the Close on Create checkbox when you created the first I/O module, repeat steps 2...6.

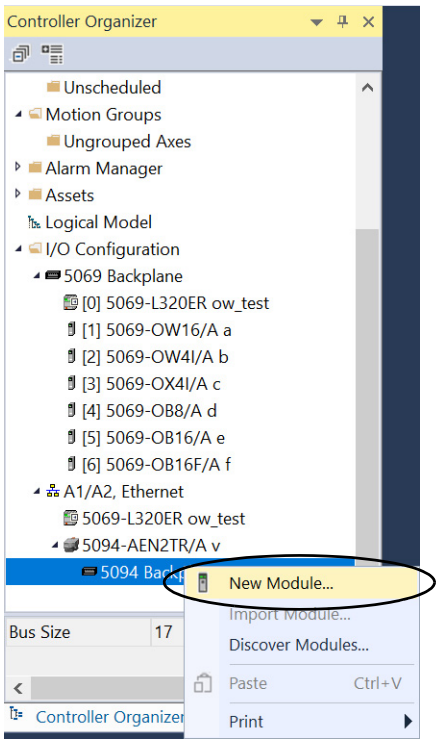
New Module

To use the New Module method with FLEX 5000 I/O modules, complete these steps.

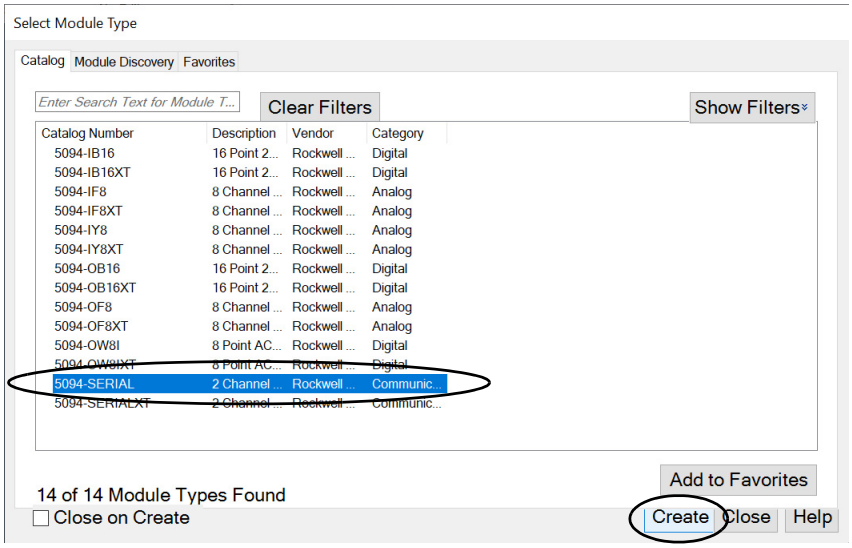
This example shows how to add an I/O module when the Logix Designer application project is offline.

You can add new modules when the project is online, if desired. In this case, the steps are similar to the steps described in Discover Modules on page 36. One exception is that, in step 1, you choose New Module instead of Discover Modules.

1. Right-click the FLEX 5000 EtherNet/IP adapter and choose New Module.

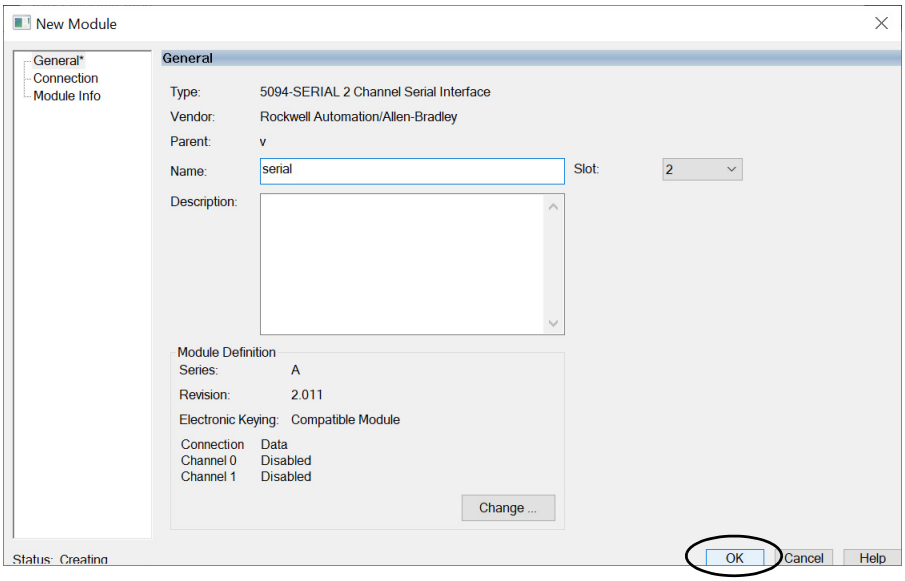


2. Select the module and click Create.



The New Module dialog box appears with a list of categories on the left side. The number and type of categories varies by module type.

3. You can click OK to use the default configuration as shown or edit the module configuration. The rest of this chapter describes how to edit module configuration categories.



To add additional remote I/O modules with this method, complete one of the following:

- If you cleared the Close on Create checkbox when you created the first I/O module, repeat steps 2...3.
- If you did not clear the Close on Create checkbox when you created the first I/O module, repeat steps 1...3.

Edit the Module Configuration

This section describes how to edit the default module configuration when you add the module to the project. You can also change the module configuration after you add it to the project.

To change module configuration after you add it to the project, double-click the catalog number in the I/O Configuration tree or right-click on the catalog number and choose Properties.

The following categories are available:

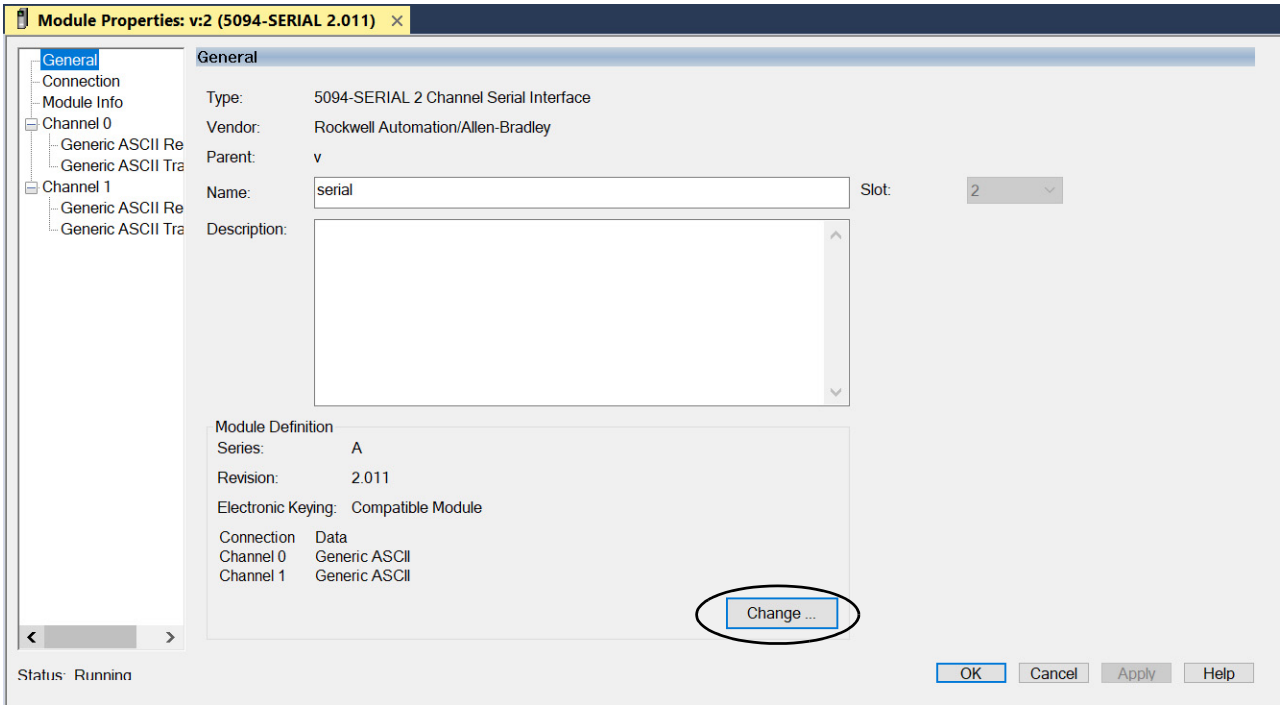
- General Category
- Connection Category
- Module Info Category
- Channel Category

General Category

The General category appears first when you create a module. You use this category to complete the following tasks:

- Name the module
- Assign a slot number (required)
- Describe the module
- Access the Module Definition

Figure 6 - General Module Properties



Module Definition

To access the Module Definition parameters, click Change on the General page of the Module Properties.

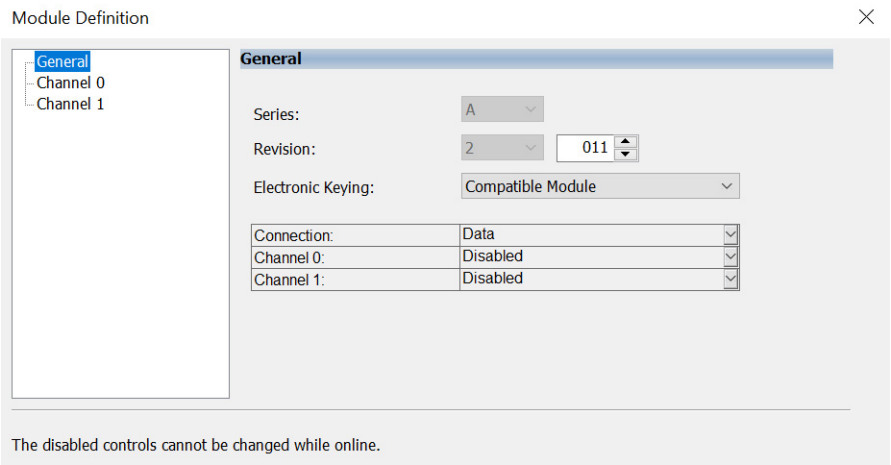


Table 15 describes the parameters that are available on the Module Definition dialog box for FLEX 5000 serial modules.

Table 15 - Module Definition Parameters

Parameter	Definition	Available Choices[1]
Series	Module hardware series	Module-specific
Revision	Module firmware revision, including major and minor revision levels	Module-specific
Electronic Keying	Software method by which you reduce the possibility of using the wrong device in a control system. For more information, see the following: • View the Module Tags on page 52 • Electronic Keying in Logix 5000 Control Systems Application Technique, publication LOGIX-AT001	Exact Match Compatible Module Disable Keying
Connection	Definition establishes a connection between the controller and the module.	Data
Channel 0/1	Determines the protocol that is used on Channel 0 or 1.	Disabled Generic ASCII Modbus Master Modbus Slave For more information, see Table 16.

(1) The choices that are available vary by module type and catalog number.

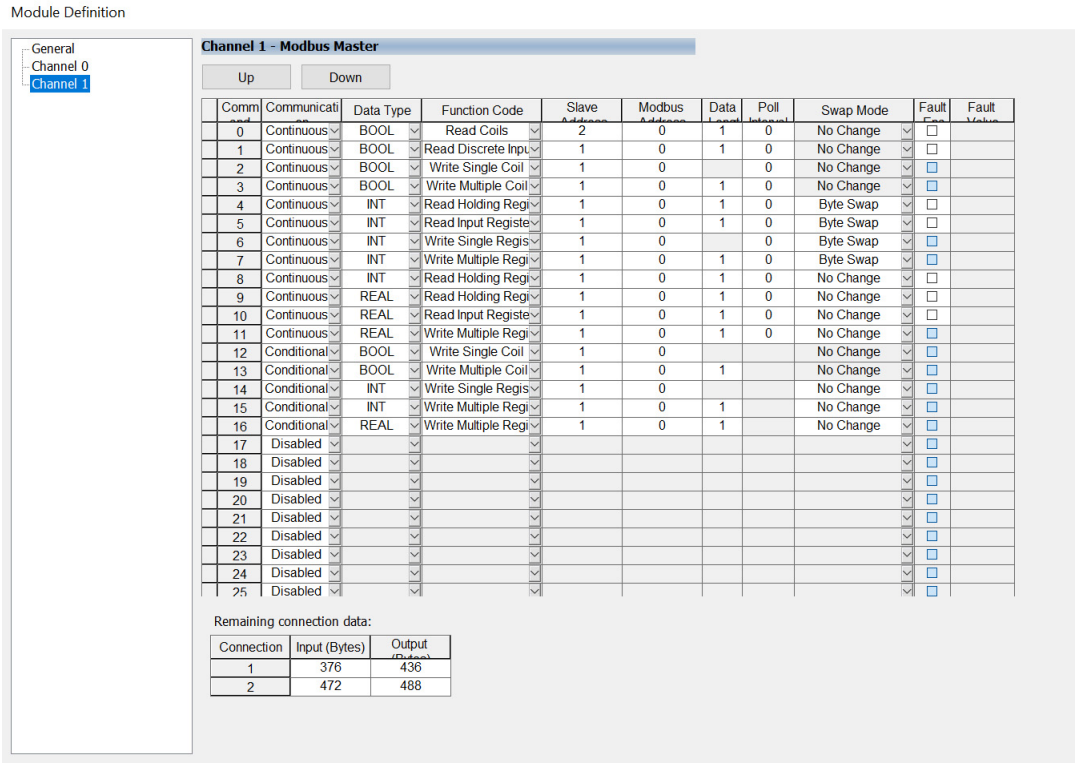
Table 16 - Communication Mode Definitions

Communication Mode	Definition
Disabled	The channel is unused and no physical connection is enabled between the controller and the serial module.
Generic ASCII	A general mode of serial communication where you can define any user data to be transmitted or received in the communication.
Modbus Master	The device sends Modbus queries or write commands to the slaves devices connected to it.
Modbus Slave	The device operates as a slave e to an external master and waits for commands from the Master.

Depending on your parameter choice for one channel, you can have additional configurable parameters. The parameters are available on the left-side corresponding pages. Depending on what you choose, you can have additional parameters.

- If you choose Modbus Master, see Figure 7 for an example of your options.
- If you choose Modbus Slave, see Figure 8 for an example of your options.

Figure 7 - Modbus Master Module Definition Parameters



IMPORTANT On the module definition screen, you will see two options at the top for moving the commands up and down. If either of these buttons are used, make sure that the user program is adjusted to reflect the new location of the command or the program will show and error.

Table 17 shows definitions of the Modbus Master configurable parameters.

Table 17 - Modbus Master Command List Parameters

Parameter	Definition
Communication Method	Communication Method: • Disabled • Continuous - sending the command based on the Poll interval value. • Conditional - only for Write command and triggered when the write value has changed. – After initialization, the base value for conditional is 0. So if the first consumed data is not 0, the conditional command is transmitted. If you want to use value 0, you must set another value and reset the value to 0 again.
Data Type	• BOOL • INT • REAL
Function Code	• Read Coil Status- This code reads Modbus addresses 000000…065535. These bit values indicate coil status. • Read Input Status- This code reads Modbus addresses 100000…165535. These read-only bit values indicate discrete input status. • Read Holding Registers - This code reads Modbus addresses 400000…465535. This is a 16-bit word value. • Read Input Registers - This code reads Modbus addresses 300000…365535. They are also 16-bit word values, but are read-only data. • The Modbus Master cannot write to these registers. • Force Single Coil - This code writes to Modbus addresses 000000…065535. This command writes to only one coil. • Preset Single Register - This code writes to Modbus addresses 400000…465535. This command writes to only one coil. • Force Multiple Coils - This code writes to multiple coil values to the slave addresses 000000…065535. • Preset Multiple Registers - This code writes to multiple register values to the slave device at addresses 400000…465535 For more information about Master Command List Function Codes, see Appendix B.
Slave Address	Node Address of the Modbus Slave device (1…247 and 0 for broadcast).
Modbus Address Offset (0-based)	Offset to the actual Modbus Address. Holding Register address of 400003 is equal to 00003 in the Modbus Address offset. (0…65535).
Data Length	Number of data points being read (1…125 for registers, 1…2000 for coils and discrete inputs) or written (1…123 for registers, 1…1968 for coils.

Table 17 - Modbus Master Command List Parameters (Continued)

Parameter	Definition
Poll Interval	• 0 – keeps repeating as fast as possible. • 1...32,767– the time, in seconds, for the command to be sent periodically in Continuous mode.
Swap Mode	• No Change – no swapping of data. • Word Swap – Words are swapped before sending out. For example, 11112222 = 22221111 • Byte Swap – each byte is swapped. For example, 11223344 = 22114433 • Word and Byte swap – both word and byte is swapped.
Fault Enable	Check to write the Fault Value into the received data if this **Read Command** fails. This **does not** apply to write commands.
Fault Value	User-defined value to replace received data.

Master Command List Limitations

- A maximum of 50 commands can be created. The commands are subject to available connection memory.
- Each Modbus Master supports up to two data connections.
- Each data connection supports a max of 446 bytes of read data and 482 bytes of write data. For a second connection, the connection supports a max of 450 bytes of read data.
- Each command can have a max of either 125 words read or 123 words write or 2000 coils/discrete input read or 1968 coils written.
- Each command uses:
  - Two bytes of input data per holding register or input register read.
  - One byte of input data per every 1...8 coils or discrete inputs read.
  - Two bytes of output data per holding register written.
  - One byte of output data per every 1...8 coils written.
- An error message appears when connection memory is exceeded.

Master Command Memory Usage

The following is an example of master command memory usage.

- Maximum single connection configuration for registers:

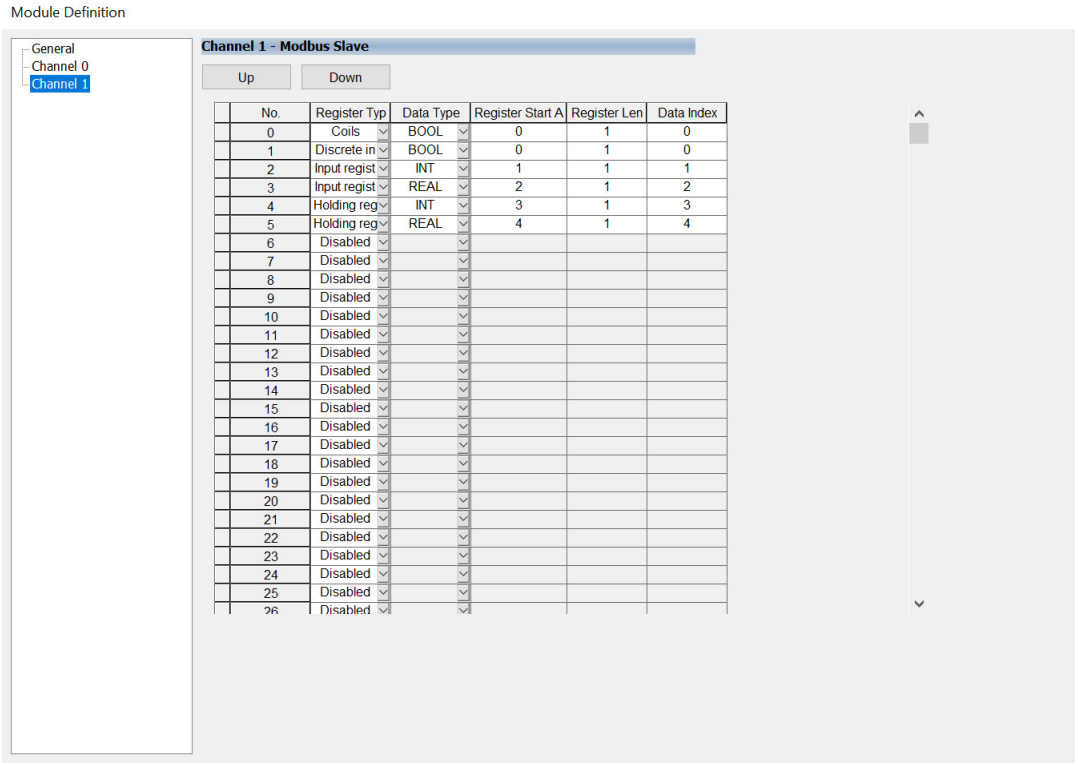
	Comm	Communicati	Data Type	Function Code	Slave	Modbus	Data	Poll	Swap Mode	Fault	Fault
	0	Continuous	INT	Read Holding Regi	1	0	125	0	No Change	☐	
	1	Continuous	INT	Read Holding Regi	1	125	98	0	No Change	☐	
	2	Conditional	INT	Write Multiple Regi	1	0	123		No Change	☐	
	3	Conditional	INT	Write Multiple Regi	1	123	118		No Change	☐	

  - 223 words * 2 bytes/word read = 446 input bytes
  - 241 words * 2 bytes/word written = 482 output bytes
- Maximum single connection configuration for coils:

	Comm	Communicati	Data Type	Function Code	Slave	Modbus	Data	Poll	Swap Mode	Fault	Fault
	0	Continuous	BOOL	Read Coils	1	0	2000	0	No Change	☐	
	1	Continuous	BOOL	Read Coils	1	2000	1576	0	No Change	☐	
	2	Conditional	BOOL	Write Multiple Coil	1	0	1968		No Change	☐	
	3	Conditional	BOOL	Write Multiple Coil	1	1968	1888		No Change	☐	

  - 3576 bits / 8 bits/byte read = 447 input bytes
  - 3856 bits / 8 bits/byte = 482 output bytes

Figure 8 - Modbus Slave Module Definition Parameters



IMPORTANT On the module definition screen, you will see two options at the top for moving the commands up and down. If either of these buttons are used, make sure that the user program is adjusted to reflect the new location of the command or the program will show and error.

Table 18 shows definitions of the Modbus Slave configurable parameters.

Table 18 - Modbus Slave Data Mapping Parameters

Parameter	Definition
Register Type	• Disabled (default) • Coils • Discrete Inputs • Input Registers • Holding Registers
Data Type	• BOOL • INT • REAL
Register Start Address	• 0...65535 (default = 0)
Register Length	• Dependent on Data Type – BOOL: 1...128 (default =1) – INT: 1...100 (default = 1) – REAL: 1...50 (default = 1)
Data Index	Location of data in the output or input tags depending on the register type defined. For more information, see the following section on Data Index.

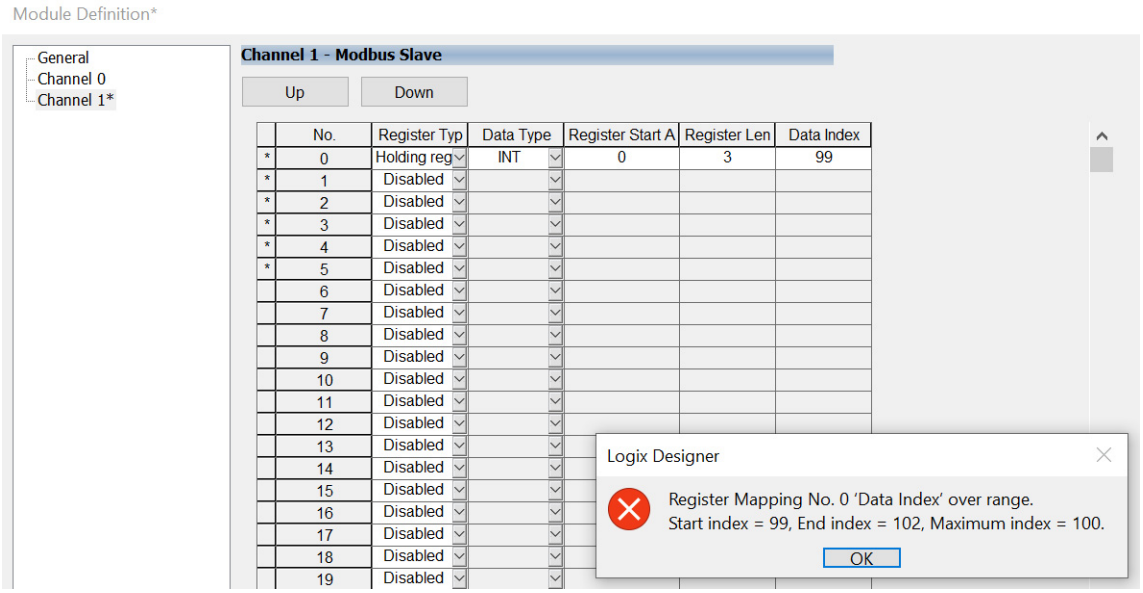
Data Index

This value indicates the offset of the register type address in the controller tags that allows the serial module to read or write the required data to or from the controller.

Each register type has a predefined array size that can be used for the Modbus Slave as indicated in the Modbus Slave Address Table Limits. For example, a Holding register has the limit of 100 INT. The array size is 0...99 and the Data index is 0...99 when the Holding register is used depending on the size and the location where you want to store the information.

If you exceed the data limit an error may occur. This error would appear like Figure 9.

Figure 9 - Data Index Error



Modbus Slave Address Table Limits

Up to 30 data point ranges can be created, subject to available memory:

- 200 byte maximum of Holding registers (up to 100 INTs or 50 REALs)
- 200 byte maximum of Input registers (up to 100 INTs or 50 REALs)
- Up to 128 Coils (Data Indexes 0...15 at 8-bit boundaries)
- Up to 128 Discrete inputs (Data Indexes 0...15 at 8-bit boundaries)

Modbus Slave Data Mapping Example

	No.	Register Typ	Data Type	Register Start A	Register Len	Data Index
	0	Holding reg	INT	3	3	0
	1	Holding reg	INT	32000	7	3
	2	Holding reg	INT	999	90	10
	3	Coils	BOOL	0	8	0
	4	Coils	BOOL	10	1	1
	5	Coils	BOOL	32000	17	2
	6	Coils	BOOL	999	88	5

- v:2:O1.Slave.HoldingRegister[0...2] = 400003...400005
- v:2:O1.Slave.HoldingRegister[3...9] = 432000...432006
- v:2:O1.Slave.HoldingRegister[10...99] = 400999...410088
- v:2:O1.Slave.Coil[0].0...0.7 = 000000...000007
- v:2:O1.Slave.Coil[1].0 = 000010
- v:2:O1.Slave.Coil[2].0...[4].0 = 0320000...032016
- v:2:O1.Slave.Coil[5].0...[15].7 = 000999...001086

▲	v:2:O1.Slave	{...}	{...}		AB:5000_Modbu...
	v:2:O1.Slave.Run	0		Decimal	BOOL
▶	v:2:O1.Slave.SequenceNumber	0		Decimal	INT
▶	v:2:O1.Slave.HoldingRegister	{...}	{...}	Decimal	INT[100]
▶	v:2:O1.Slave.Coil	{...}	{...}	Decimal	SINT[16]
▶	v:2:O1.Slave.InputRegister	{...}	{...}	Decimal	INT[100]
▶	v:2:O1.Slave.DiscreteInput	{...}	{...}	Decimal	SINT[16]

Connection Category

You use the Connection category to complete the following tasks:

- Set the RPI rate. For more information on the RPI, see page 20.
- Set the connection type to use on the EtherNet/IP network.

  For more information on unicast and multicast connections, see the EtherNet/IP Network Devices User Manual, publication ENET-UM006.
- Inhibit the module. For more information about inhibiting a module, see page 21.
- Configure whether a connection failure while the controller is in Run module causes a major or minor fault.
- View the fault code and description when a module fault occurs.

The Module Fault area of the Connection category is useful during module troubleshooting. For more information on the Module Fault area, see page 53.

Figure 10 - Generic ASCII Connection

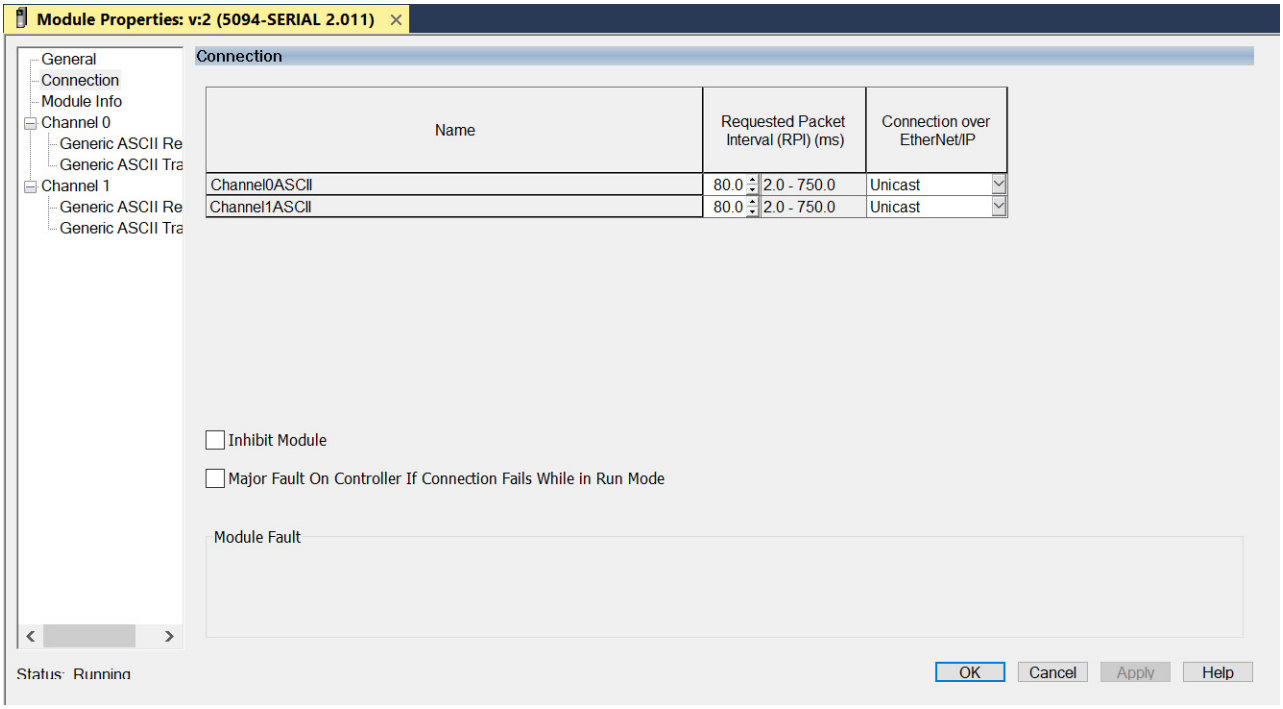
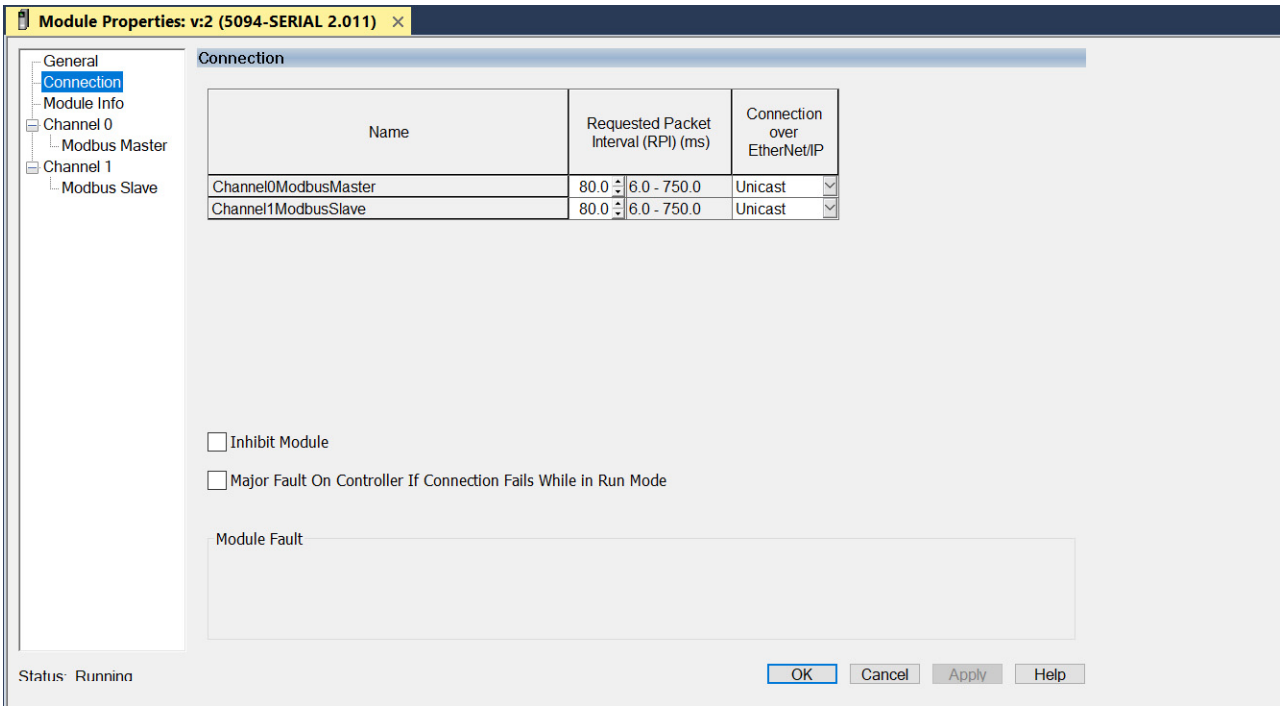


Figure 11 - Modbus Connection

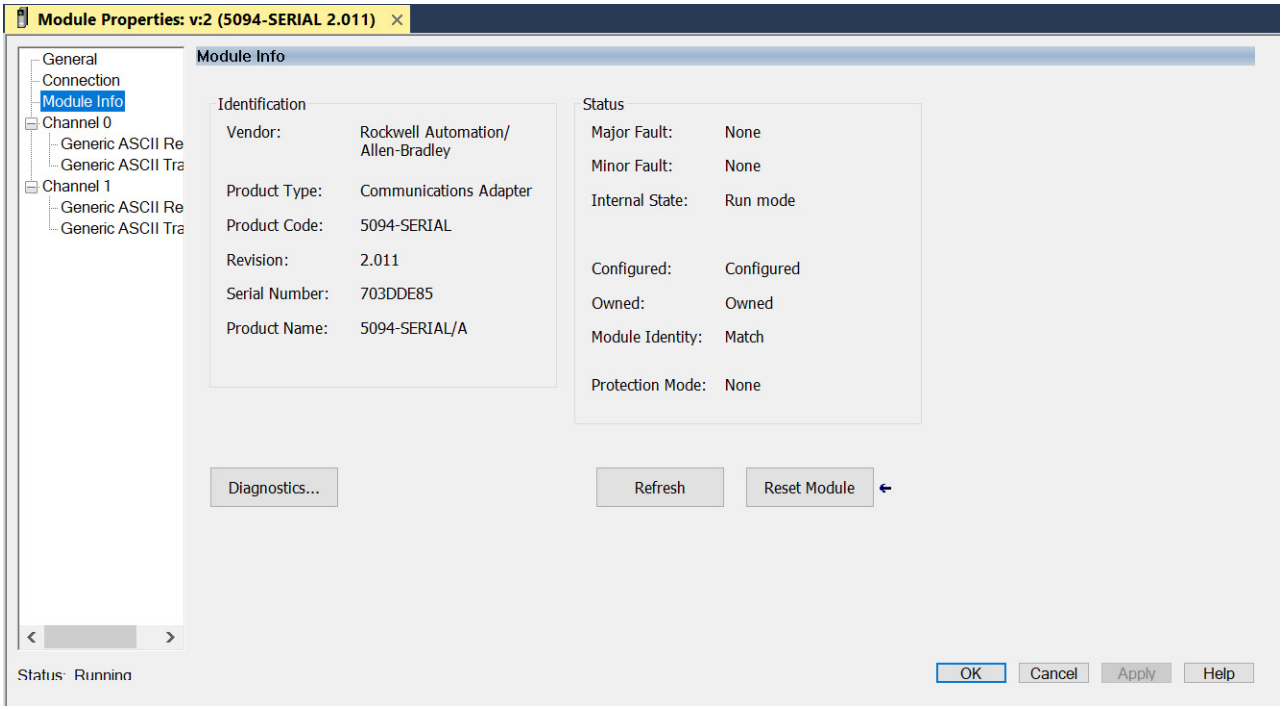


Module Info Category

The Module Info category displays module and status information about the module when the project is online. You use this category to complete the following tasks:

- Determine the identity of the module.
- Refresh the data on the screen.
- Reset the module.
- Access module diagnostic information.

Figure 12 - Module Info

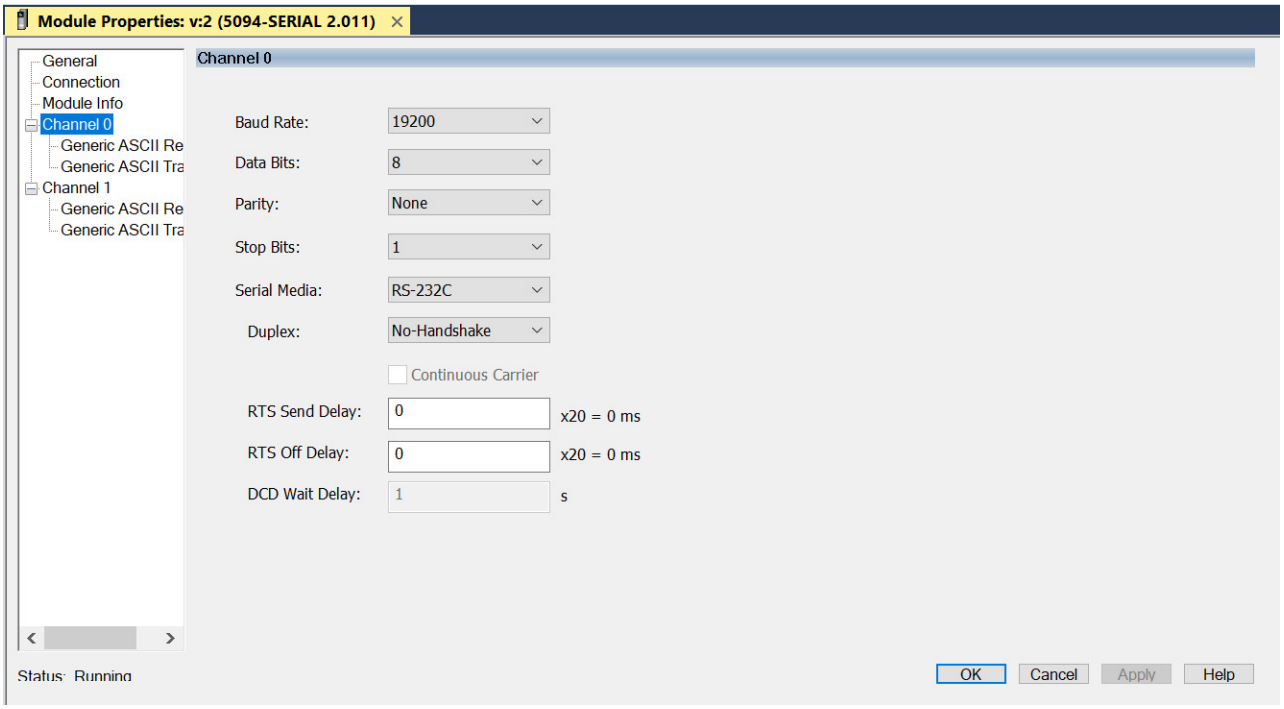


Channel Category

You use this category to complete the following tasks:

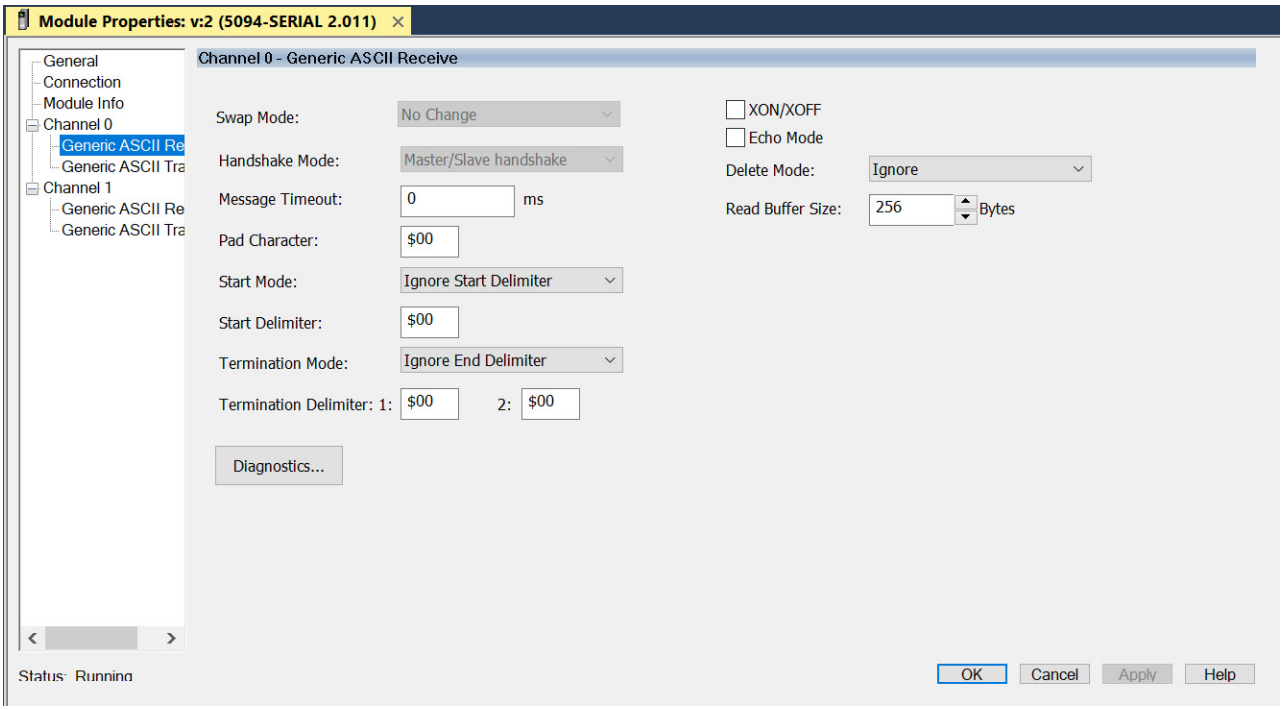
- Configure communication parameters for each channel.
- Configure Generic ASCII parameters.
- Configure Modbus parameters.

Figure 13 - Generic ASCII Channel Parameters



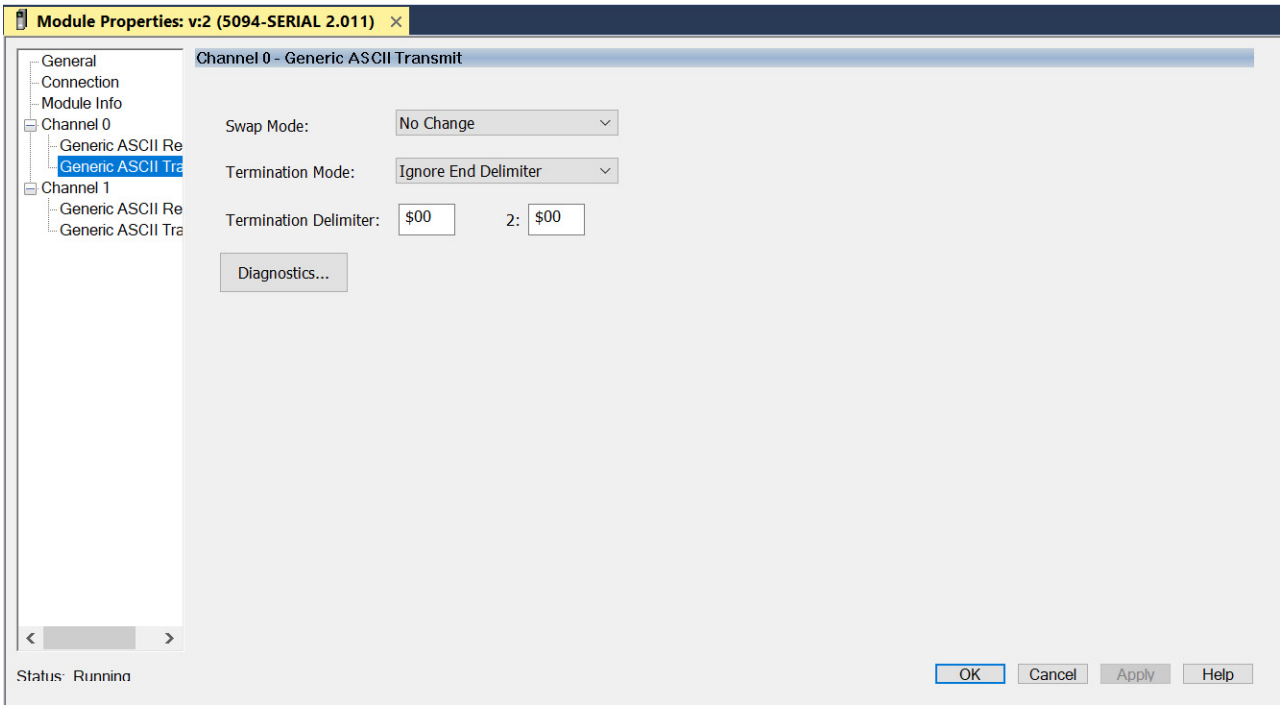
For definitions of the parameters on this page, see Common Module Functions on page 23.

Figure 14 - Generic ASCII Receive



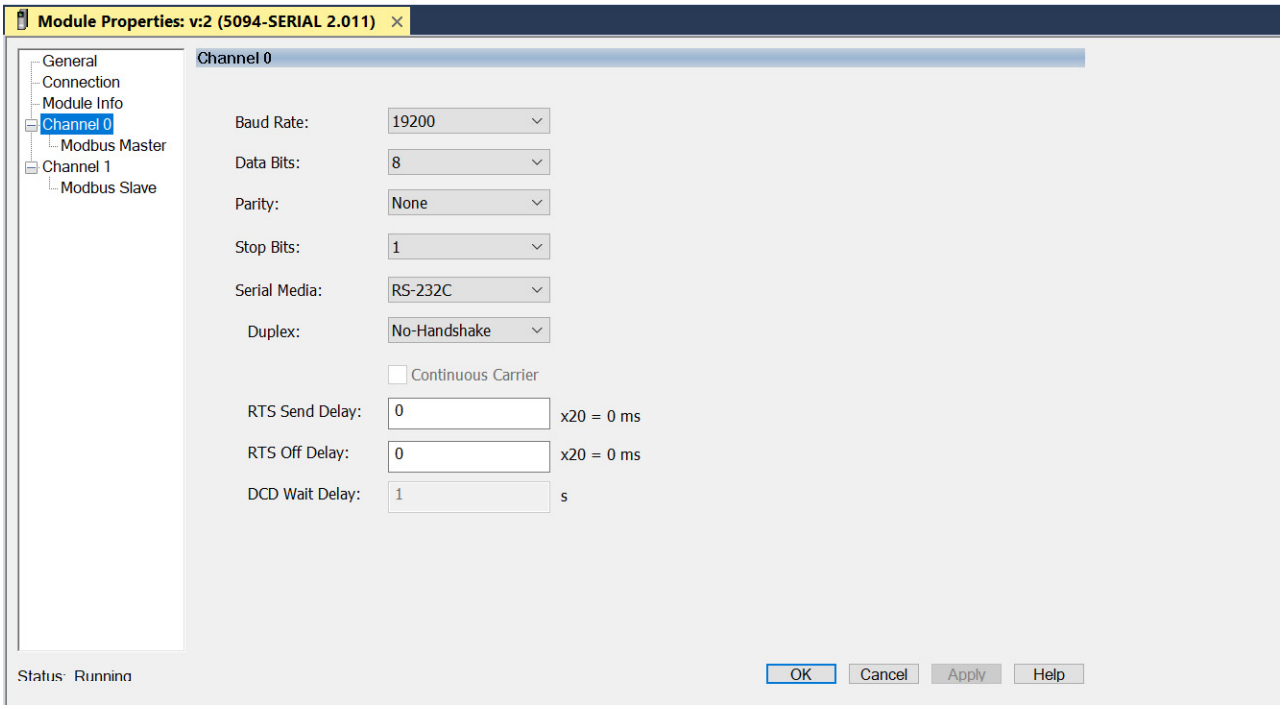
For definitions of the parameters on this page, see Generic ASCII Receive Functions on page 26.

Figure 15 - Generic ASCII Transmit



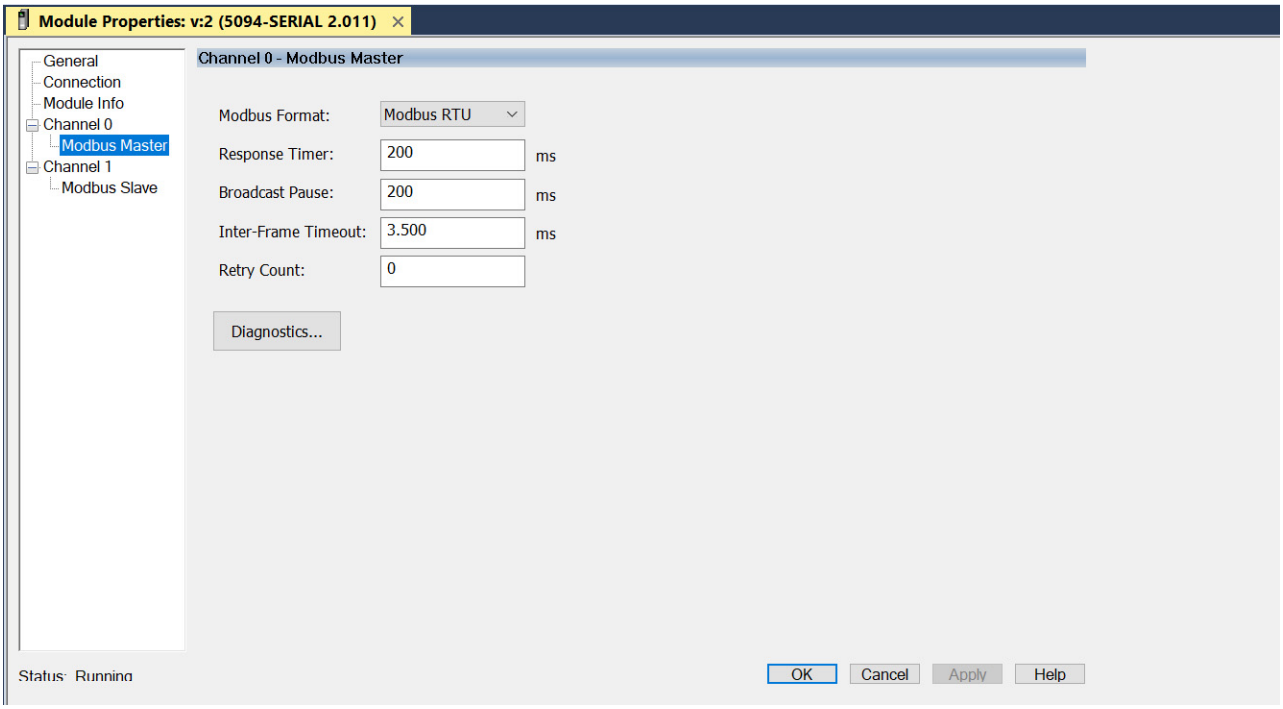
For definitions of the parameters on this page, see Generic ASCII Transmit Functions on page 25.

Figure 16 - Modbus Channel Parameters



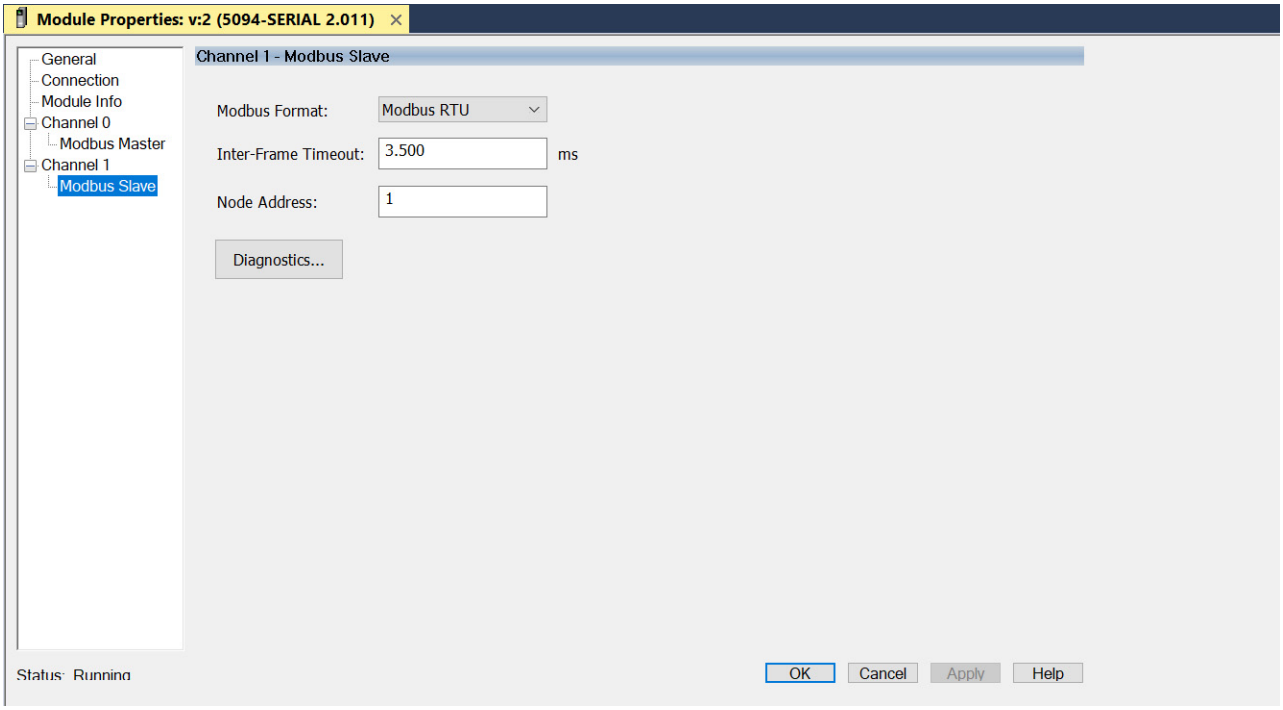
For definitions of the parameters on this page, see Common Module Functions on page 23.

Figure 17 - Modbus Master Channel Parameters



For definitions of the parameters on this page, see Modbus Master Functions on page 32.

Figure 18 - Modbus Slave Channel Parameters



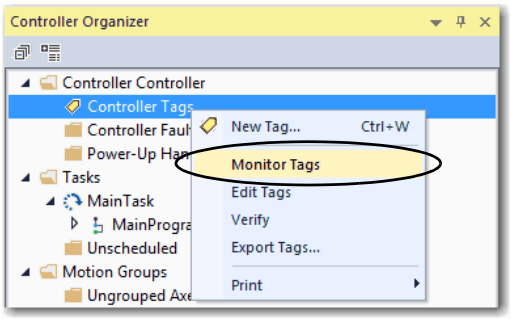
For definitions of the parameters on this page, see Modbus Slave Functions on page 33.

View the Module Tags

When you create a module, the Logix Designer application creates a set of tags that you can view in the Tag Editor. Each configured feature on your module has a distinct tag that is available for use in the controller program logic.

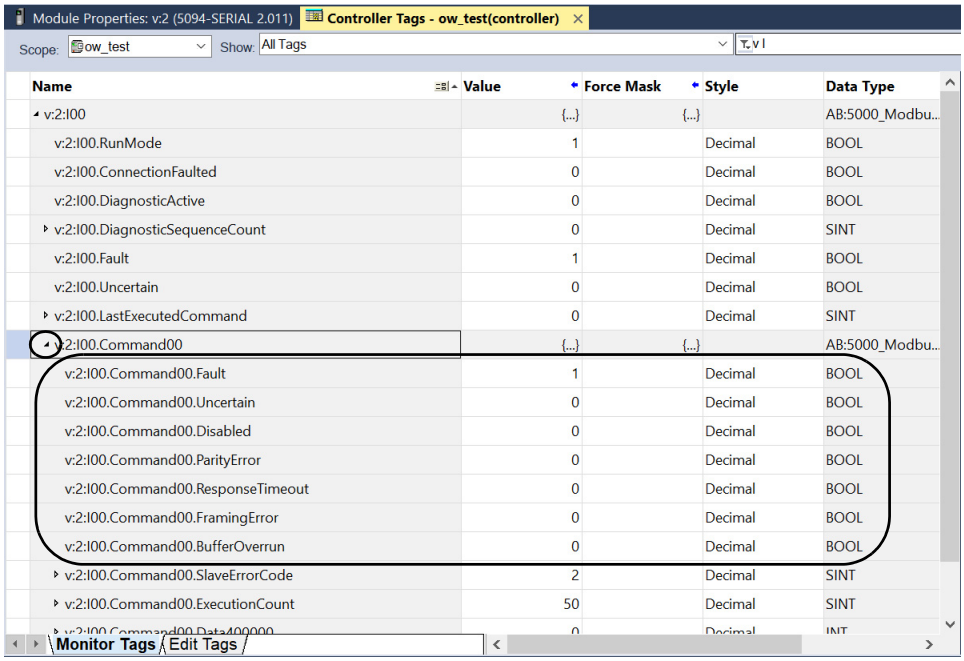
Complete the following steps to access the module tags.

1. In the Controller Organizer, right-click Controller Tags and choose Monitor Tags.



   The Controller Tags dialog box appears with data.

2. To view the tags, click the ▷ symbols as shown.



For more information on module tags, see Appendix A, Module Tags on page 63.

Troubleshoot Your Module

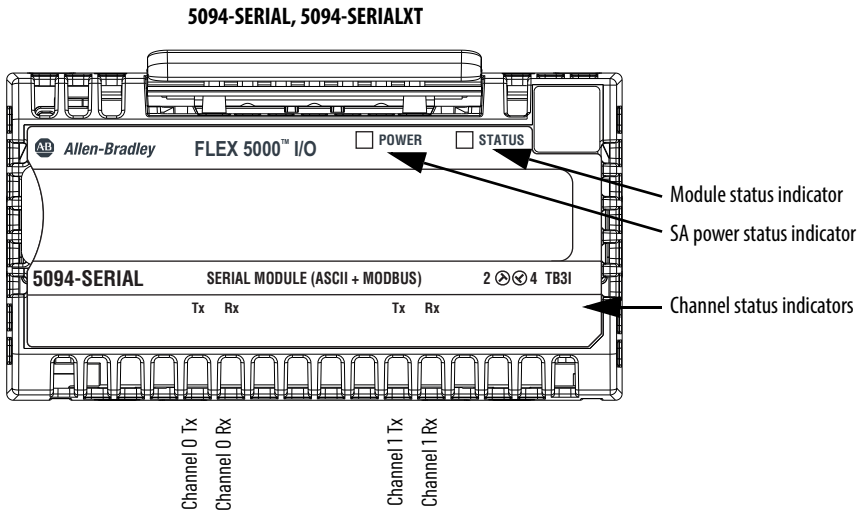
This section describes the status indicators on your FLEX 5000 serial module and methods to troubleshoot your application. Your serial module uses the following status indicators:

- SA Power indicator
- Module Status indicator
- Channel Status indicator

Module Status Indicators

Figure 19 shows the status indicators on FLEX 5000 serial modules.

Figure 19 - FLEX 5000 Serial Module Status Indicators



SA Power Indicator

Table 19 describes the SA Power indicator on FLEX 5000 serial modules.

Table 19 - SA Power Indicator - FLEX 5000 Serial Module

Indicator State	Description	Recommended Action
Off	The module is not powered.	Complete the following actions: 1. Confirm that the system is powered. 2. Confirm that the module is installed properly.
Steady green	There is SA power to the module.	None
Steady red	There is no SA power to the module.	Complete the following actions: 1. Confirm that the SA Power wiring on the terminal base is installed properly. 2. Check the following: – Confirm that there is sufficient voltage that is supplied to the module. – If an external power supply is used, confirm that the power supply is turned on. – If power is daisy chained from the previous terminal base, confirm that the wiring on the previous terminal base is installed properly.

Module Status Indicator

Table 20 describes the Module Status indicator on FLEX 5000 serial modules.

Table 20 - Module Status Indicator - FLEX 5000 Serial Module

Indicator State	Description	Recommended Action
Off	The module is not powered.	Complete the following actions: 1. Confirm that the system is powered. 2. Confirm that the module is installed properly.
Steady green	The module has a connection to the owner-controller and is operating normally.	None
Flashing green	One of the following conditions exist: • The module has powered up successfully. • The module is OK, but it does not have a connection. No connection can result from missing, incomplete, or incorrect module configuration.	Complete the following actions: • Troubleshoot your Logix Designer application to determine what is preventing a connection from the module to the controller and correct the issue. • Confirm that the system conditions require the controller to be in Remote Run mode or Run mode, transition the controller to one of those modes.
Steady red	The module experienced a nonrecoverable fault.	Complete the following actions: 1. Cycle power to the module. 2. If the status indicator remains in the steady red state, replace the module.
Flashing red	One of the following conditions exist: • A module firmware update is in progress. • A module firmware update attempt failed. • The device has experienced a recoverable fault. • A connection to the module has timed out.	Complete one of the following: • Let the firmware update progress complete. • Reattempt a firmware update after one fails. • Use the Logix Designer application to determine the cause of the module fault. The Connection and Module Info categories of the modules configuration indicate the fault type. To clear a recoverable fault, complete one of the following: – Cycle module power. – Click Reset Module in the Logix Designer application project via the Module Info category of the Module Properties dialog box. If the fault does not clear after cycling power and clicking Reset Module, contact Rockwell Automation Technical Support. • Use the Logix Designer application to determine if a connection has timed out. The Connection category in the Module Properties for the module indicates the module state, including if a connection has timed out. If a connection has timed out, determine the cause and correct it. For example, a cable failure can cause a connection timeout.

Channel Status Indicators

Table 21 describes the Channel Status indicators on FLEX 5000 serial modules.

Table 21 - Channel Status Indicators - FLEX 5000 Serial Module

Indicator State	Description	Recommended Action
Off	Channel is disabled or there is no communication	None
Steady yellow	Communicating	None
Flashing yellow	Communicating	None
Flashing red	Serial Port Communication Error	• Cycle module power. • Check SA power, see Table 19 for more information. • Check serial port configuration and setup.

Use the Logix Designer Application for Troubleshooting

In addition to the status indicator display on the module, the Logix Designer application indicates the presence of fault conditions.

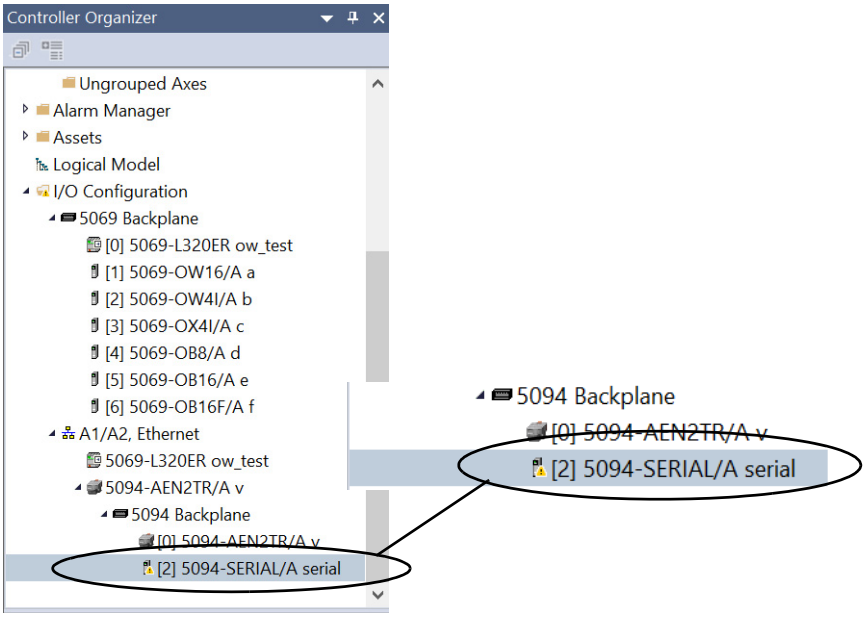
Fault conditions are reported in the following ways:

- Warning Signal in the I/O Configuration Tree
- Status and Fault Information in Module Properties Categories
- Logix Designer Application Tag Editor

Warning Signal in the I/O Configuration Tree

As shown in Figure 20, a warning icon appears in the I/O Configuration tree when a fault occurs.

Figure 20 - Warning Icon in Controller Organizer



Status and Fault Information in Module Properties Categories

The Module Properties section in the Logix Designer application includes a series of categories. The numbers and types of categories varies by module type.

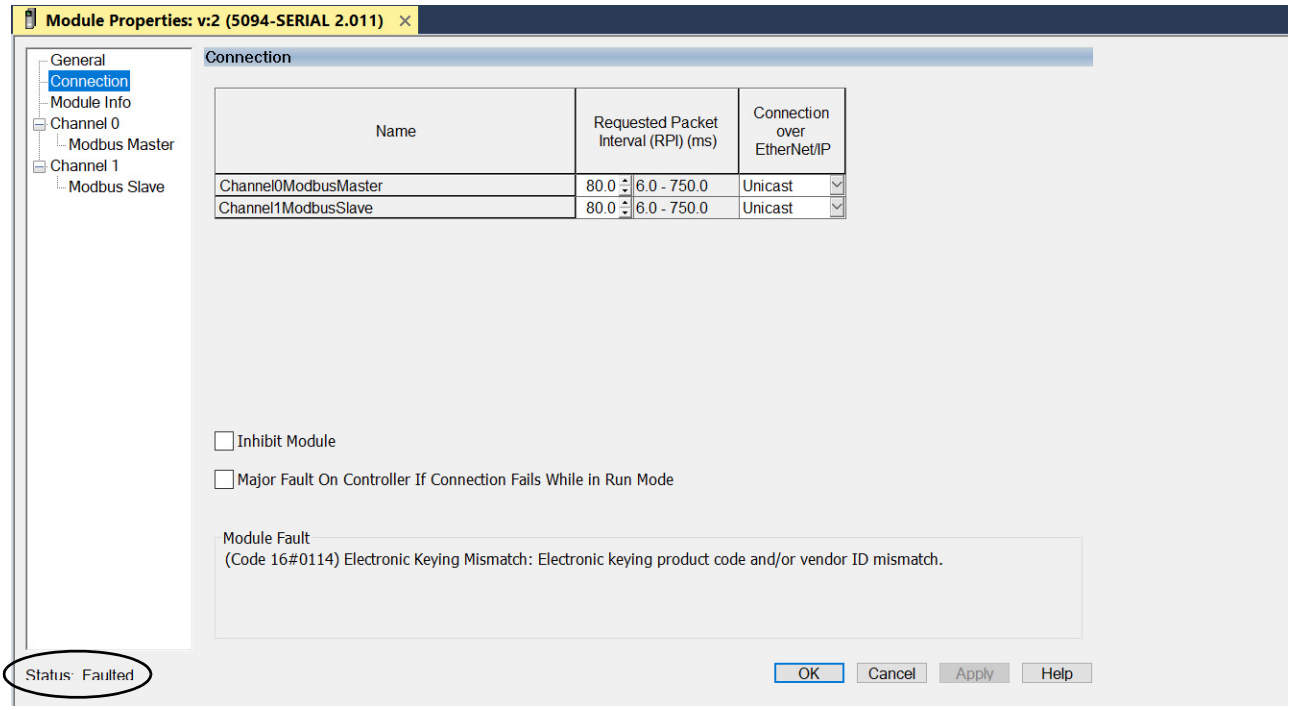
Each category includes options to configure the module or monitor the status of the module. The following are ways to monitor the state of a module for faults:

- Module Status on Module Properties
- Module Fault Descriptions on Connection Category
- Module Diagnostics Dialog Box
- Channel Diagnostics

Module Status on Module Properties

As shown in Figure 21, the status of a module is indicated on the lower left corner of the Modules Properties.

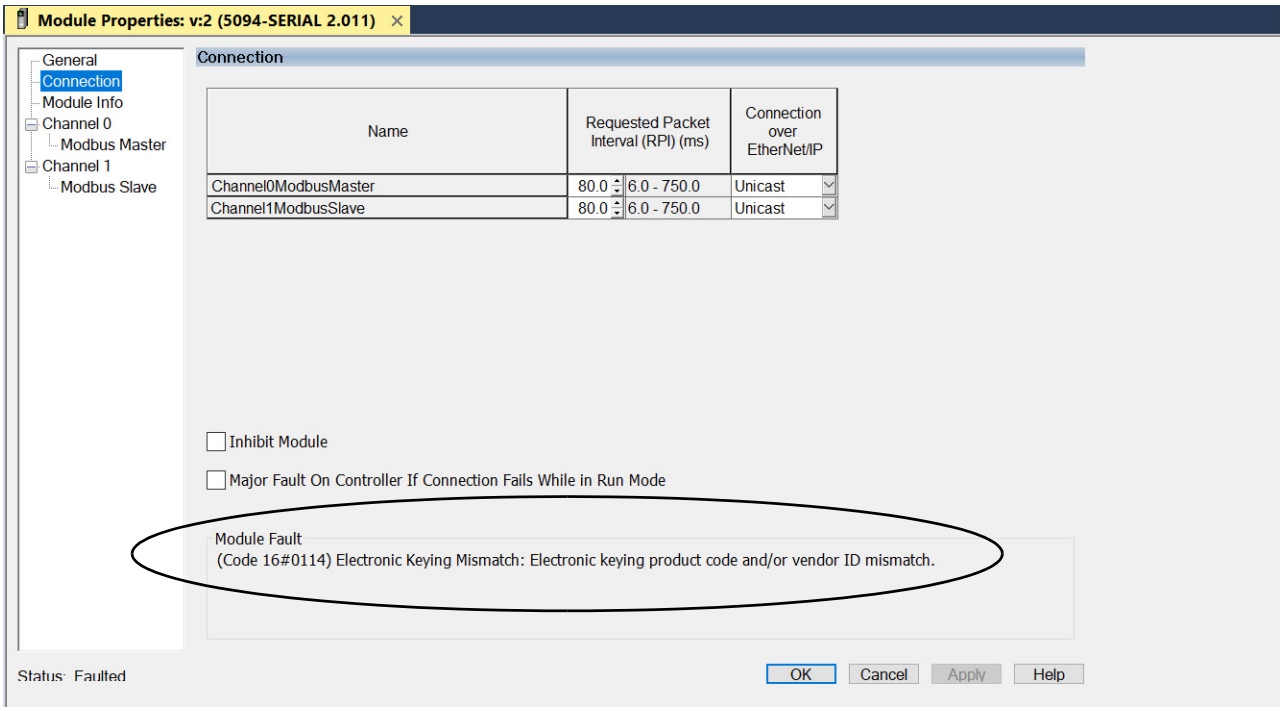
Figure 21 - Fault Message in Status Line



Module Fault Descriptions on Connection Category

As shown in Figure 22, a module fault description that includes an error code that is associated with the specific fault type is listed on the Connection category.

Figure 22 - Fault Description with Error Code



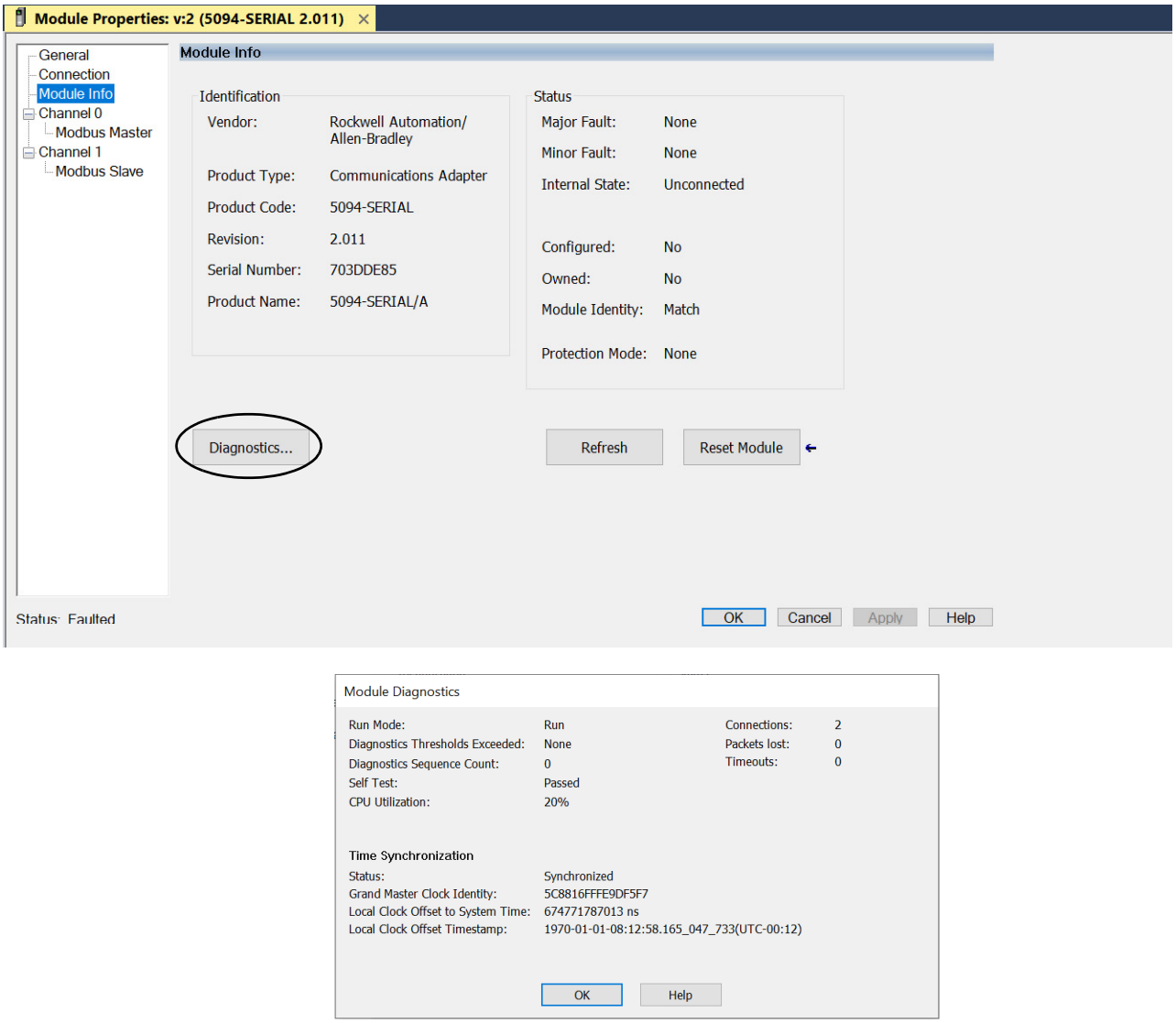
Module Diagnostics Dialog Box

Module Diagnostics are accessible from the Module Info category on the Module Properties dialog box, as shown in Figure 23.

You can use diagnostics in a Logix Designer application project to monitor module operating conditions and to troubleshoot issues that affect a module. You can use diagnostics only when the **project is online**.

Module diagnostics provide information on a module-wide basis. For example, the Module Diagnostics dialog box indicates the mode within which a module is operating, that is, Run, Remote Run, Remote Program, or Program.

Figure 23 - Module Diagnostics



Channel Diagnostics

You can use diagnostics in a Logix Designer application project to monitor module and/or channel operating conditions and to troubleshoot issues that affect a module and/or channel. You can use diagnostics only when the **project is online**.

Channel diagnostics provide information on an individual channel basis. For example, you can check individual channels on a FLEX 5000 serial module for the presence of a Short Circuit condition.

Figure 24 shows how to access the channel diagnostics and Figure 25 shows the diagnostics dialog box.

Figure 24 - Channel Diagnostics

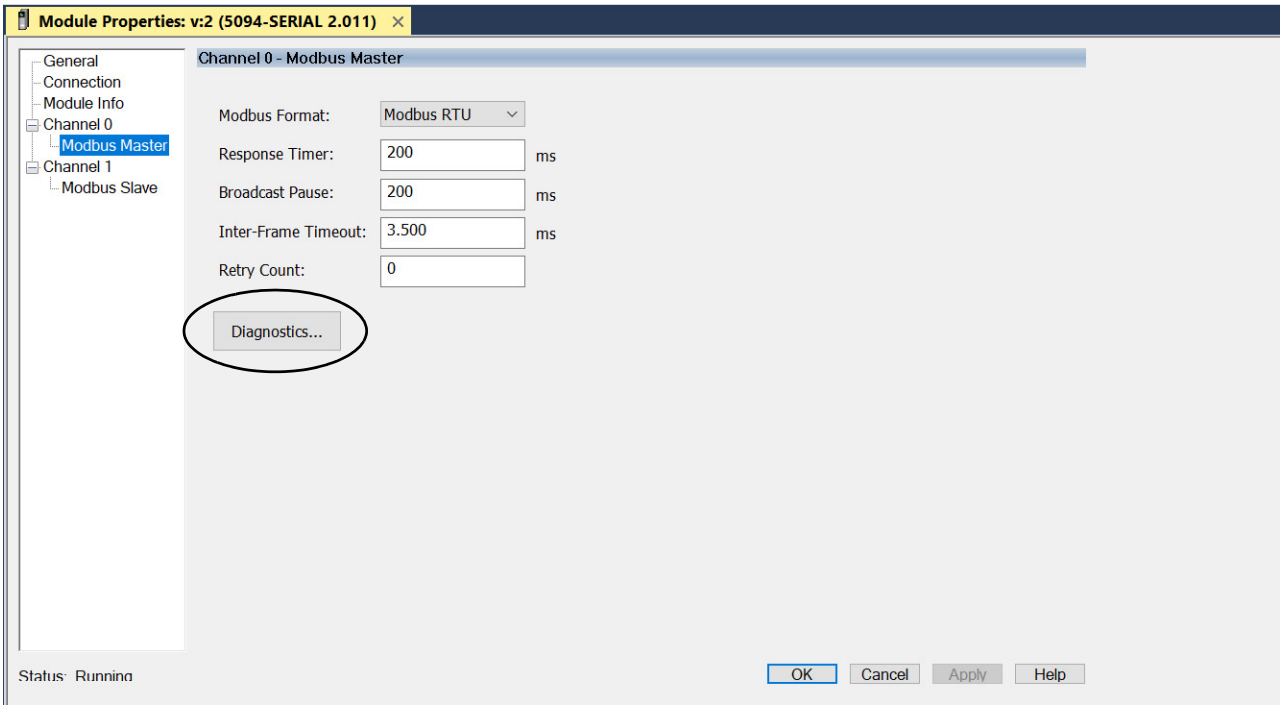
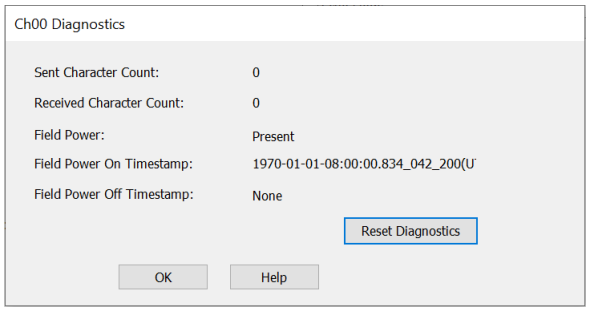
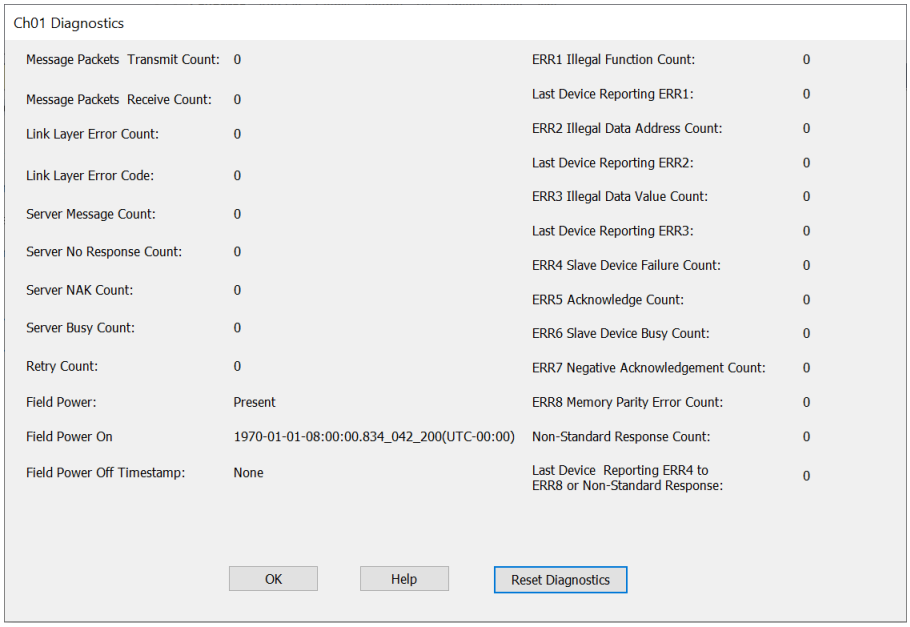


Figure 25 - Channel Diagnostics Dialog Box

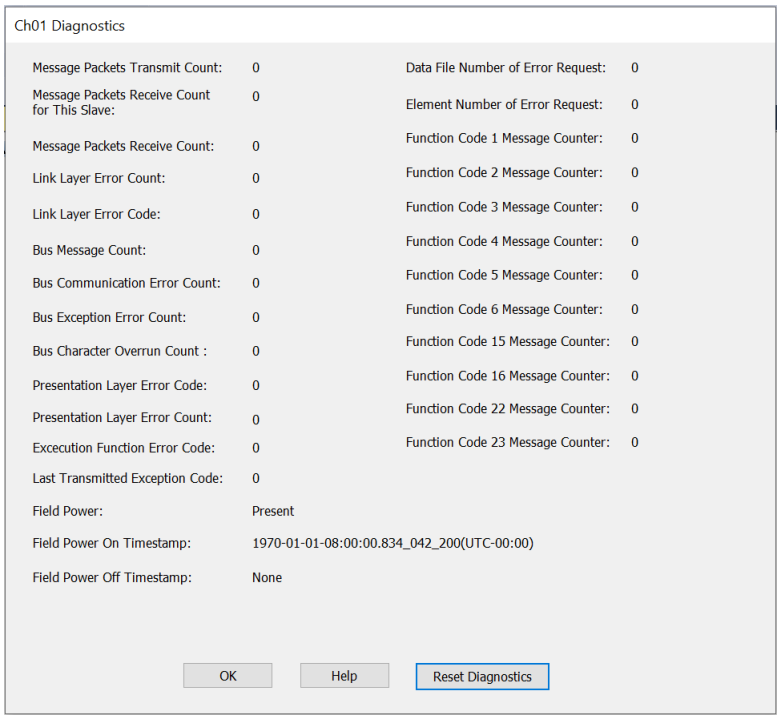
Generic ASCII Transmit/Receive



Modbus Master



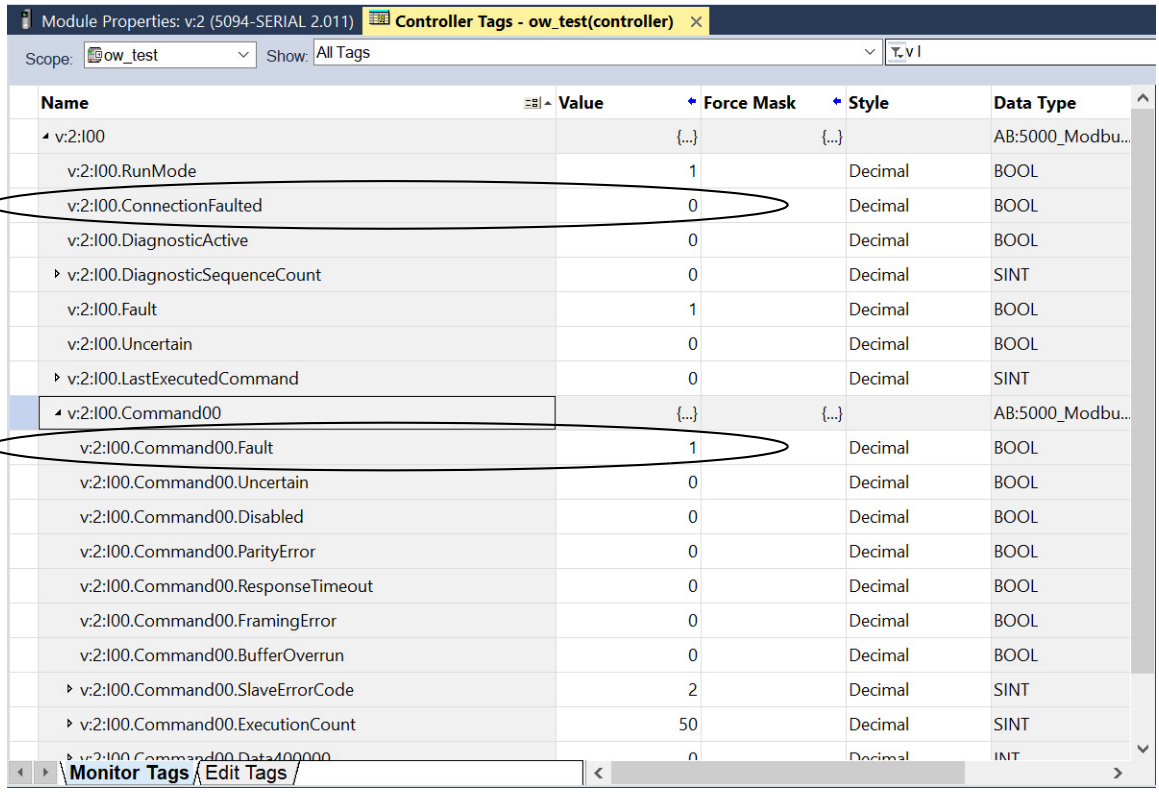
Modbus Slave



Logix Designer Application Tag Editor

Figure 26 show how fault conditions are indicated in the controller tags for the module.

Figure 26 - Fault Indication in Controller Tags



Notes:

Module Tags

Module tags are created when you add a module to the Logix Designer application project. The 5094-SERIALXT module uses the same module tags as the 5094-SERIAL module.

The tables contained in this appendix list all tags available with a module. Not all tags in the list are used when that module type is added to a project. Tag use varies by module configuration.

Tag Name Conventions

The module tag names use defined naming conventions. The conventions are as follows:

- Adapter name
- Module name
- Slot number
- Tag type and channel number – If Generic ASCII or Modbus Slave is used in the Module Definition for the channel.
- Tag type, channel number, and number of connections – If Modbus Master is used in the Module Definition for the channel.
- Parameter

Generic ASCII and Modbus Slave Name Conventions

The following is an example for a Generic ASCII or Modbus Slave tag name. The conventions for the example, “SERIAL_PROJECT:1:I0.RunMode”, would be the following:

- Adapter = name of the FLEX 5000 EtherNet/IP adapter in the FLEX 5000 I/O system
- SERIAL_PROJECT = name of the module
- 1 = slot number
- I0 = tag type (input) and channel (0)
  - The possible FLEX 5000 serial tag types are I (input) and O (output)
  - The possible channels are channel 0 and channel 1

- RunMode = Parameter

Modbus Master Name Conventions

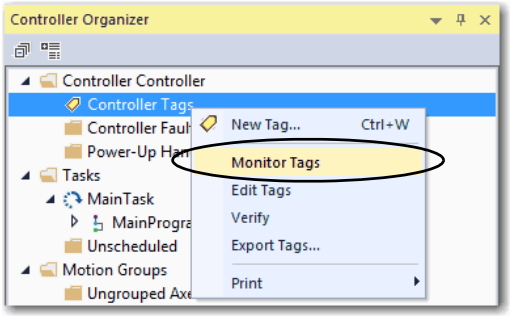
The following is an example for a Modbus Master tag name. The conventions for the example, “SERIAL_PROJECT:1:I00.RunMode”, would be the following:

- Adapter = name of the FLEX 5000 EtherNet/IP adapter in the FLEX 5000 I/O system
- SERIAL_PROJECT = name of the module
- 1 = slot number
- I00 = tag type (input), channel (0), and number of connections (0)
  - The possible FLEX 5000 serial tag types are I (input) and O (output)
  - The possible channels are channel 0 and channel 1
  - The possible number of connections are 0 or 1
- RunMode = Parameter

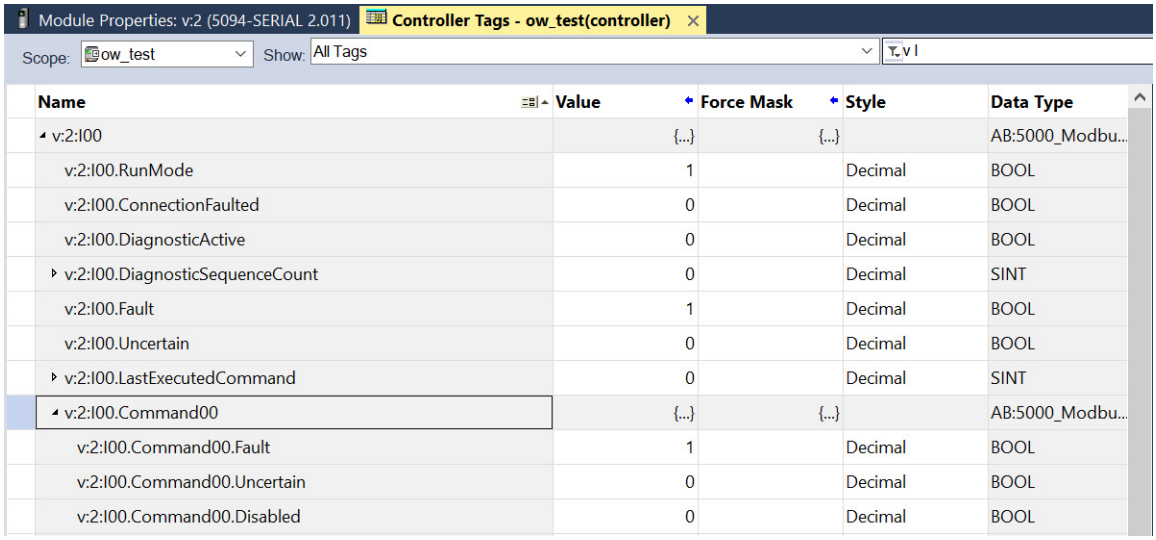
Access the Tags

You view tags from the Tag Editor.

1. Open your Logix Designer application project.
2. Right-click Controller Tags and choose Monitor Tags.



3. Open the tags as necessary to view specific tags.



Channel Configured for Generic ASCII

This section describes the tags that are created when you choose the Generic ASCII option for a channel in the module definition dialog box.

Input Tags

Table 22 describes the Generic ASCII input tags for the FLEX 5000 serial module.

Table 22 - FLEX 5000 Serial Module Generic ASCII Input Tags

Name	Data Type	Definition	Valid Values
Ix.RunMode	BOOL	The channel's operating state.	• 0 = Idle • 1 = Run
Ix.ConnectionFaulted	BOOL	Indicates if a connection is running. The module sets this tag to 0 when connected. If the module is not connected, it changes the tag to 1.	• 0 = Connection running • 1 = Connection not running
Ix.DiagnosticActive	BOOL	Indicates if any diagnostics are active or if the prognostics threshold is reached.	• 0 = No diagnostics active • 1 = One or more diagnostics are active or the prognostics threshold is reached
Ix.DiagnosticSequenceCount	SINT	Increments for each time a distinct diagnostic condition is detected, and when a distinct diagnostic condition transitions from detected to not detected. Set to zero by product reset or power cycle. Wraps from 255 (-1) to 1 skipping zero.	-128...+127 The value of 0 is skipped except during module power-up.
Ix.ASCII.Fault	BOOL	Indicates that channel data is inaccurate and cannot be trusted for use in the application.	• 0 = Good • 1 = Bad, causing fault If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ix.ASCII.Uncertain	BOOL	Indicates that the channel data can be inaccurate but the degree of inaccuracy is not known.	• 0 = Good data • 1 = Uncertain data If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ix.ASCII.TxDataLost	BOOL	The transmitted data was lost. Until Clear Buffer, this bit continues to set.	• 0 = Non-occurrence • 1 = Occurrence
Ix.ASCII.RxDataLost	BOOL	The received data was lost. Until Clear Buffer, this bit continues to set.	• 0 = Non-occurrence • 1 = Occurrence
Ix.ASCII.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
Ix.ASCII.TxFIFOEmpty	BOOL	Data in transmit FIFO. The FIFO is not empty. The output FIFO has not sent all of its data to the ASCII device.	• 0 = Not Empty • 1 = Empty
Ix.ASCII.RxFIFOEmpty	BOOL	Data in the receive FIFO. The FIFO is not empty. The input FIFO has not sent all of its data to the interface.	• 0 = Not Empty • 1 = Empty
Ix.ASCII.NonDelimitedRecord	BOOL	An input record is received and sent to the interface that was not triggered by receiving a delimiter character. This event occurs when either the buffer is filled to its maximum receive size or a Message Timeout has occurred.	• 0 = Not Produced • 1 = Produced
Ix.ASCII.HandshakeError	BOOL	Handshake error. Used for Handshake mode only.	• 0 = Non-occurrence • 1 = Occurrence
Ix.ASCII.NewData	BOOL	New data. Used for Handshake mode only.	• 0 = None • 1 = New Data
Ix.ASCII.TxDataSent	BOOL	Indicates that the module has sent the data indicated by the Tx Transaction ID and can accept more transmit data.	• 0 = Not Complete • 1 = Complete
Ix.ASCII.TxDataLengthInvalid	BOOL	Indicates whether the Ox.ASCII.TxDataLength is valid.	• 0 = Correct • 1 = Length of TxDataLength is illegal
Ix.ASCII.RxDataLengthInvalid	BOOL	Indicates whether the Ox.ASCII.RxDataLength is valid.	• 0 = Correct • 1 = Length of RxDataLength is illegal[1]
Ix.ASCII.FramingError	BOOL	Indicates whether an error in framing has occurred.	• 0 = Non-occurrence • 1 = Occurrence

Table 22 - FLEX 5000 Serial Module Generic ASCII Input Tags (Continued)

Name	Data Type	Definition	Valid Values
Ix.ASCII.BufferOverRun	BOOL	Indicates whether a buffer overrun has occurred.	• 0 = Non-occurrence • 1 = Occurrence
Ix.ASCII.CTS	BOOL	Indicates whether the CTS is active.	• 0 = Inactive • 1 = Active
Ix.ASCII.RTS	BOOL	Indicates whether the RTS is active.	• 0 = Inactive • 1 = Active
Ix.ASCII.DSR	BOOL	Indicates whether the DSR is active.	• 0 = Inactive • 1 = Active
Ix.ASCII.DCD	BOOL	Indicates whether the DCD is active.	• 0 = Inactive • 1 = Active
Ix.ASCII.DTR	BOOL	Indicates whether the DTR is active.	• 0 = Inactive • 1 = Active
Ix.ASCII.XOFF	BOOL	Flag for detection XOff (Flow control)	• 0 = On • 1 = XOff is detected
Ix.ASCII.BREAK	BOOL	Status shows whether a Break Interrupt occurred or not. If a Break Interrupt is detected, the module keeps receiving binary data from the Serial Port. The Receive data includes null Data (0x00).	• 0 = Not Detected • 1 = Detected
Ix.ASCII.TxAck	SINT	Feedback from the latest Transmit Transaction ID.	-128...+127
Ix.ASCII.RxID	SINT	Notification of receiving by updating number.	-128...+127
Ix.ASCII.RxDataLength	INT	Length of Produce Data of each channel.	1...256
Ix.ASCII.RxData[x](2)	SINT	Received data from serial port.	-128...+127

(1) Length greater than the Max buffer length defined in the module definition page.
(2) X represents any possible value from 0...255.

Output Tags

Table 23 describes the Generic ASCII output tags for the FLEX 5000 serial module.

Table 23 - FLEX 5000 Serial Module Generic ASCII Output Tags

Name	Data Type	Definition	Valid Values
Ox.ASCII.TxID	SINT	This ID is incremented when you want to transmit data from the serial port.	-128...+127
Ox.ASCII.RxID	SINT	This ID is incremented when you receive data from the serial port. It is only used in Master/Slave Handshake Mode.	-128...+127
Ox.ASCII.TxDataLength	INT	Length of Transmitted Data of each channel.	1...256
Ox.ASCII.RxDataLength	INT	Length of Received Data of each channel.	1...256
Ox.ASCII.ClearBuffer	BOOL	If Clear Buffer the bit changes from 0 to 1, the Receive and Transaction buffer is cleared. In Half Duplex, the RTS signal level is cleared (set to Inactive) When it is at 0: Clearbuffer is triggered.	• 0 = No Change • 1 = Buffer Cleared
Ox.ASCII.DTR	BOOL	Signal level of DTR line sent out when rising edge of EXEC bit is detected.	• 0 = Inactive • 1 = Active
Ox.ASCII.RTS	BOOL	Signal level of the RTS line sent out when rising edge of EXEC bit is detected.	• 0 = Inactive • 1 = Active
Ox.ASCII.EXEC	BOOL	If EXEC bit changes 0 to 1, FW will output a signal level, which is set in the DTR/RTS tag.	• 0 = Inactive • 1 = Active
Ox.ASCII.TxData[x](1)	SINT	Output data from module.	-128...+127

(1) X represents any possible value from 0...255.

Channel Configured for Modbus Master

This section describes the tags that are created when you choose the Modbus Master option for a channel in the module definition dialog box.

The *xx* in the tag names represents the channel number because the module has two channels, and both channels support the use of Modbus Master. The *yy* represents the Modbus Master command number and the *zzzzzz* represents the Modbus data address.

Input Tags

Table 24 describes the Modbus Master input tags for the FLEX 5000 serial module.

Table 24 - FLEX 5000 Serial Module Modbus Master Input Tags

Name	Data Type	Definition	Valid Values
Ixx.RunMode	BOOL	The channel's operating state.	• 0 = Idle • 1 = Run
Ixx.ConnectionFaulted	BOOL	Indicates if a connection is running. The module sets this tag to 0 when connected. If the module is not connected, it changes the tag to 1.	• 0 = Connection running • 1 = Connection not running
Ixx.DiagnosticActive	BOOL	Indicates if any diagnostics are active or if the prognostics threshold is reached.	• 0 = No diagnostics active • 1 = One or more diagnostics are active or the prognostics threshold is reached
Ixx.DiagnosticSequenceCount	SINT	Increments for each time a distinct diagnostic condition is detected, and when a distinct diagnostic condition transitions from detected to not detected. Set to zero by product reset or power cycle. Wraps from 255 (-1) to 1 skipping zero.	-128...+127 The value of 0 is skipped except during module power-up.
Ixx.Fault	BOOL	Indicates that channel data is inaccurate and cannot be trusted for use in the application.	• 0 = Good • 1 = Bad, causing fault If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.Uncertain	BOOL	Indicates that the channel data can be inaccurate but the degree of inaccuracy is not known.	• 0 = Good data • 1 = Uncertain data If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.LastExecutedCommand	SINT	Indicates the number of the latest executed command, and the Update Counter indicates the status for command level. If this value is updated, either a new command is executed, not changed, command is not executed, or have not finished yet.	• -1 (value that is shown before any command) • 0...49
Ixx.Command.yy.Fault	BOOL	Indicates that channel data is inaccurate and cannot be trusted for use in the application.	• 0 = Good • 1 = Bad, causing fault If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.Command.yy.Uncertain	BOOL	Indicates that the channel data can be inaccurate but the degree of inaccuracy is not known.	• 0 = Good data • 1 = Uncertain data If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.Command.yy.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Command.yy.ResponseTimeout	BOOL	Shows if there was a response timeout.	• 0 = Non-occurrence • 1 = Occurrence

Table 24 - FLEX 5000 Serial Module Modbus Master Input Tags (Continued)

Name	Data Type	Definition	Valid Values
Ixx.Command.yy.FramingError	BOOL	Shows if there was a framing error.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Command.yy.BufferOverRun	BOOL	Shows if a buffer overrun has occurred.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Command.yy.SlaveErrorCode	SINT	Indicates if a slave error code has occurred. **IMPORTANT**: A slave error code is not equal to an exception code.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Command.yy.ExecutionCount	SINT	The number of times the command is executed.	-128...+127
Ixx.Command.yy.Datazzzzzz	SINT, INT, or REAL	Command read response data. The data for this tag is dynamic according to how the command list is created.	-128...+127 -32768...+32767

Output Tags

Table 25 describes the Modbus Master output tags for the FLEX 5000 serial module.

Table 25 - FLEX 5000 Serial Module Modbus Master Output Tags

Name	Data Type	Definition	Valid Values
Oxx.Run	BOOL	The channel's operating state.	• 0 = Idle • 1 = Run(1)
Oxx.Command.yy.Datazzzzzz	SINT, INT, or REAL	Command read response data. The data for this tag is dynamic according to how the command list is created.	• -128...+127 • -32768...+32767

(1) User logic must set the Run bit in order for the Modbus Master commands to execute.

Channel Configured for Modbus Slave

This section describes the tags that are created when you choose the Modbus Slave option for a channel in the module definition dialog box.

The *xx* in the tag names represents the channel number because the module has two channels, and both channels support the use of Modbus Slave.

Input Tags

Table 26 describes the Modbus Slave input tags for the FLEX 5000 serial module.

Table 26 - FLEX 5000 Serial Module Modbus Slave Input Tags

Name	Data Type	Definition	Valid Values
Ixx.RunMode	BOOL	The channel's operating state.	• 0 = Idle • 1 = Run
Ixx.ConnectionFaulted	BOOL	Indicates if a connection is running. The module sets this tag to 0 when connected. If the module is not connected, it changes the tag to 1.	• 0 = Connection running • 1 = Connection not running
Ixx.DiagnosticActive	BOOL	Indicates if any diagnostics are active or if the prognostics threshold is reached.	• 0 = No diagnostics active • 1 = One or more diagnostics are active or the prognostics threshold is reached
Ixx.DiagnosticSequenceCount	SINT	Increments for each time a distinct diagnostic condition is detected, and when a distinct diagnostic condition transitions from detected to not detected. Set to zero by product reset or power cycle. Wraps from 255 (-1) to 1 skipping zero.	-128...+127 The value of 0 is skipped except during module power-up.

Table 26 - FLEX 5000 Serial Module Modbus Slave Input Tags (Continued)

Name	Data Type	Definition	Valid Values
Ixx.Slave.Fault	BOOL	Indicates that channel data is inaccurate and cannot be trusted for use in the application.	• 0 = Good • 1 = Bad, causing fault If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.Slave.Uncertain	BOOL	Indicates that the channel data can be inaccurate but the degree of inaccuracy is not known.	• 0 = Good data • 1 = Uncertain data If the tag is set to 1, you must troubleshoot the module to correct the cause of the inaccuracy. **IMPORTANT**: Once the condition that causes the tag to change to 1 is removed, the tag automatically resets to 0.
Ixx.Slave.CRC_LRCError	BOOL	Status that shows whether a CRC (LRC) Error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Slave.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Slave.IllegalDataAddress	BOOL	Status that shows whether user requests are out of the Modbus Register Address.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Slave.BufferOverRun	BOOL	Shows if a buffer overrun has occurred.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Slave.FramingError	BOOL	Shows if there was a framing error.	• 0 = Non-occurrence • 1 = Occurrence
Ixx.Slave.SequenceNumberAck	INT	Acknowledges the sequence number.	-32768...+32767
Ixx.Slave.MasterUpdateCount	INT	Any Modbus change will update this counter.	-32768...+32767
Ixx.Slave.HoldingRegister[x](1)	INT	Produce Data that are written by Modbus Master as Data in Produce Tag.	-32768...+32767
Ixx.Slave.Coil[x](2)	SINT	Produce Data that are written by Modbus Master as Data in Produce Tag.	-128...+127

(1) X represents any possible value from 0...99.
(2) X represents any possible value from 0...15.

Output Tags

Table 27 describes the Modbus Slave output tags for the FLEX 5000 serial module.

Table 27 - FLEX 5000 Serial Module Modbus Slave Output Tags

Name	Data Type	Definition	Valid Values
Oxx.Run	BOOL	The channel's operating state.	• 0 = Idle • 1 = Run(1)
Oxx.Slave.SequenceNumber	INT	Sequence number for updating slave data from controller.	-32768...+32767
Oxx.Slave.HoldingRegister[x](2)	INT	Location of holding register values defined by user for the serial module.	-32768...+32767
Oxx.Slave.Coils[x](3)	SINT	Location of slave coil values defined by user for the serial module.	-128...+127
Oxx.Slave.InputRegister[x](1)	INT	Location of input register values defined by user for the serial module.	-32768...+32767
Oxx.Slave.DiscreteInput[x](2)	SINT	Location of discrete input values defined by user for the serial module.	-128...+127

(1) The Run bit is to start the update of the output (O) tags values into the Serial module. The serial module will always respond to the external Modbus master, but they will be using the old values if the RUN bit is not enabled but new data is on the output O tag.
(2) X represents any possible value from 0...99.
(3) X represents any possible value from 0...15.

Notes:

Master Command List

Master Command List Function Codes

Read Coil Status (Function Code 01)

Query

This function allows you to obtain the ON/OFF status of logic coils (Modbus 0x range) used to control discrete outputs from the addressed slave only. Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial coil address to be read (Starting Address) and the number of locations that are interrogated to obtain status data.

The addressing allows up to 2000 coils to be obtained at each request. However, the specific slave device can have restrictions that lower the maximum quantity. The coils are numbered from zero; (coil number 1 = zero, coil number 2 = one, coil number 3 = two, and so on). The following table is a sample read output status request to read coils 0020 to 0056 (37 coils) from slave device number 11.

This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

Node Address	Function Code	Data Start Point High	Data Start Point Low	Number of Points High	Number of Points Low	Error Check Field (2 bytes)
0B	01	00	13	00	25	CRC

Response

An example response to Read Coil Status is as shown in the following table. The data is packed one bit for each coil. The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data is packed with one bit for each coil (1 = ON, 0 = OFF). The low-order bit of the first character contains the addressed coil, and the remainder follows. For coil quantities that are not even multiples of eight, the last characters are completed with zeros at high-order end. The quantity of data characters is always specified as quantity of RTU characters, that is, the number is the same whether RTU or ASCII is used.

Because the slave interface device is serviced at the end of a controller's scan, data reflects coil status at the end of the scan. Some slaves limit the quantity of coils provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status from sequential scans.

Node Address	Function Code	Byte Count	Data Coil Status 20...27	Data Coil Status 28...35	Data Coil Status 36...43	Data Coil Status 44...51	Data Coil Status 52...56	Error Check Field (2 bytes)
0B	01	05	CD	6B	B2	0E	1B	CRC

The status of coils 20...27 is shown as CD (HEX) = 1100 1101 (Binary). Reading from left to right, this status shows that coils 27, 26, 23, 22, and 20 are all on. The other Data Coil Status bytes are decoded similarly. Due to the quantity of coil statuses that are requested, the last data field, which is shown 1B (HEX) = 0001 1011 (Binary), contains the status of only five coils (52...56) instead of eight coils. The 3 leftmost bits are provided as zeros to fill the 8-bit format.

Read Input Status (Function Code 02)

Query

This function allows you to obtain the ON/OFF status of discrete inputs (Modbus 1x range) in the addressed slave. PC Broadcast mode is not supported with this function code. In addition to the slave address and function fields, the message requires that the information field contain the initial input address to be read (Starting Address) and the number of locations that are interrogated to obtain status data.

The addressing allows up to 2000 inputs to be obtained at each request; however, the specific slave device can have restrictions that lower the maximum quantity. The inputs are numbered form zero; (input 10001 = zero, input 10002 = one, input 10003 = two, and so on, for a 584).

The following table is a sample read input status request to read inputs 10197 to 10218 (22 coils) from slave number 11.

This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

Node Address	Function Code	Data Start Point High	Data Start Point Low	Number of Points High	Number of Points Low	Error Check Field (2 bytes)
0B	02	00	C4	00	16	CRC

Response

An example response to Read Input Status is as shown in the following table. The data is packed one bit for each input. The response includes the slave address, function code, quantity of data characters, the data characters, and error checking. Data is packed with one bit for each input (1=ON, 0=OFF). The lower-order bit of the first character contains the addressed input, and the remainder follows. For input quantities that are not even multiples of eight, the last characters are completed with zeros at high-order end. The quantity of data characters is always specified as a quantity of RTU characters, that is, the number is the same whether RTU or ASCII is used.

Because the slave interface device is serviced at the end of a controller's scan, the data reflect input status at the end of the scan. Some slaves limit the quantity of inputs provided each scan; thus, for large coil quantities, multiple PC transactions must be made using coil status for sequential scans.

Node Address	Function Code	Byte Count	Data Discrete Input 10197...10204	Data Discrete Input 10205...10212	Data Discrete Input 10213...10218	Error Check Field (2 bytes)
0B	02	03	AC	DB	35	CRC

The status of inputs 10197...10204 is shown as AC (HEX) = 10101 1100 (binary). Reading from left to right, this show that inputs 10204, 10202, and 10199 are all on. The other input data bytes are decoded similar.

Due to the quantity of input statuses that are requested, the last data field that is shown as 35 HEX = 0011 0101 (binary) contains the status of only 6 inputs (10213...102180) instead of 8 inputs. The two leftmost bits are provided as zeros to fill the 8-bit format.

Read Holding Registers (Function Code 03)

Query

This function allows you to retrieve the contents of holding registers 4xxxx (Modbus 4x range) in the addressed slave. The registers can store the numerical values of associated timers and counters that can be driven to external devices. The addressing allows retrieving up to 125 registers at each request; however, the specific slave device can have restrictions that lower this maximum quantity. The registers are numbered form zero (40001 = zero, 40002 = one, and so on). The broadcast mode is not allowed.

The following example reads registers 40108...40110 (three registers) from slave number 11.

This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

Node Address	Function Code	Data Start Registers High	Data Start Registers Low	Number of Registers High	Number of Registers Low	Error Check Field (2 bytes)
0B	03	00	6B	00	03	CRC

Response

The addressed slave responds with its address and the function code, followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are two bytes each, with the binary content right justified within each pair of characters. The first byte includes the high-order bits and the second, the low-order bits.

Because the slave interface device is normally serviced at the end of the controller's scan, the data reflect the register content at the end of the scan. Some slaves limit the quantity of register content provided each scan; thus for large register quantities, multiple transmissions are made using register content from sequential scans.

In the example below, the registers 40108...40110 have the decimal contents 555, 0, and 100 respectively.

Node Address	Function Code	Byte Count	High Data	Low Data	High Data	Low Data	High Data	Low Data	Error Check Field (2 bytes)
0B	03	06	02	2B	00	00	00	64	CRC

Read Input Registers (Function Code 04)

Query

This function retrieves the contents of the controller's input registers from the Modbus 3x range. These locations receive their values from devices that are connected to the I/O structure and can only be referenced, not altered from within the controller. The addressing allows retrieving up to 125 registers at each request; however, the specific slave device can have restrictions that lower this maximum quantity. The registers are numbered for zero (30001 = zero, 30002 = one, and so on). Broadcast mode is not allowed.

The following example requests the contents of register 30009 in slave number 11.

This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

Node Address	Function Code	Data Start Point High	Data Start Point Low	Data Number of Points High	Data Number of Points Low	Error Check Field (2 bytes)
0B	04	00	08	00	01	CRC

Response

The addressed slave responds with its address and the function code followed by the information field. The information field contains 1 byte describing the quantity of data bytes to be returned. The contents of the registers requested (DATA) are 2 bytes each, with the binary content right justified within each pair of characters. The first byte includes the high-order bits and the second, the low-order bits.

Because the slave interface is normally serviced at the end of the controller's scan, the data reflect the register content at the end of the scan. Each PC limits the quantity of register contents provided each scan; thus for large register quantities, multiple PC scans are required, and the data that is provided is from sequential scans.

In the following example, the register 30009 contains the decimal value 0.

Table 28 -

Node Address	Function Code	Byte Count	Data Input Register High	Data Input Register Low	Error Check Field (2 bytes)
0B	04	02	00	00	CRC

Force Single Coil (Function Code 05)

Query

This Function Code forces one coil (Modbus 0x range) either ON or OFF. Any coil that exists within the controller can be forced to either state (ON or OFF). However, because the controller is actively scanning, unless the coil is disabled, the controller can also alter the state of the coil. Coils are numbered from zero (coil 0001 = zero, coil 0002 = one, and so on). The data value 65,280 (FF00 HEX) sets the coil ON and the value zero turns it OFF; all other values are illegal and do not affect that coil.

The use of slave address 00 (Broadcast Mode) forces all attached slaves to modify the desired coil.

- Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.
- This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

The following example is a request to slave number 11 to turn ON coil 0173.

Node Address	Function Code	Data Start Bit High	Data Start Bit Low	Number of Bits High	Number of Bits Low	Error Check Field (2 bytes)
0B	05	00	AC	FF	00	CRC

Response

The normal response to the Command Request is to retransmit the message as received after the coil state has been altered.

Node Address	Function Code	Data Coil Bit High	Data Coil Bit Low	Data On/Off	Data	Error Check Field (2 bytes)
0B	05	00	AC	FF	00	CRC

The forcing of a coil via Modbus function 5 happens regardless of whether the addressed coil is disabled or not (In ProSoft products, the coil is only affected if you implement the necessary Ladder Logic).

IMPORTANT The Modbus protocol excludes standard functions for testing or changing the DISABLE state of discrete inputs or outputs. Where applicable, this can be accomplished via device-specific Program commands (In ProSoft products, this is only accomplished through Ladder Logic programming).

Coils that are reprogrammed in the controller logic program are not automatically cleared upon power-up. Thus, if such a coil is set ON by function Code 5 and (even months later), an output is connected to that coil, the output is "hot".

Preset Single Register (Function Code 06)

Query

This Function Code allows you to modify the contents of a Modbus 4x range in the slave. This code writes to one register only. Any holding register that exists within the controller can have its contents changed by this message. However, because the controller is actively scanning, it can also alter the content of any holding register at any time. The values are provided in binary up to the maximum capacity of the controller. Unused high-order bits must be set to zero.

When used with slave address zero (Broadcast mode), all slave controllers load the specified register with the contents specified.

- Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.
- This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

The following example is a request to write the value '3' to register 40002 in slave 11.

Node Address	Function Code	Data Start Bit High	Data Start Bit Low	Preset Data Register High	Preset Data Register Low	Error Check Field (2 bytes)
0B	06	00	01	00	03	CRC

Response

The response to a preset single register request is to retransmit the query message after the register has been altered.

Node Address	Function Code	Data Register High	Data Register Low	Preset Data Register High	Preset Data Register Low	Error Check Field (2 bytes)
0B	06	00	01	00	03	CRC

Force Multiple Coils (Function Code 15)

Query

This function forces each coil (Modbus 0x range) in a consecutive block of coils to a desired ON or OFF state. Any coil that exists within the controller can be forced to either state (ON or OFF). However, because the controller is actively scanning, unless the coils are disabled, the controller can also alter the state of the coil.

Coils are numbered from zero (coil 00001 = zero, coil 00002 = one, and so on). The desired status of each coil is packed in the data field, one bit for each coil (1= ON, 0= OFF). The use of slave address 0 (Broadcast Mode) forces all attached slaves to modify the desired coils.

- Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.
- This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

The following example forces 10 coils starting at address 20 (13 HEX). The two data fields, CD =1100 and 00 = 0000 000, indicate that coils 27, 26, 23, 22, and 20 are to be forced on.

Node Address	Function Code	Coil Address High	Coil Address Low	Number of Coils High	Number of Coils Low	Byte Count	Force Data High 20...27	Force Data Low 28...29	Error Check Field (2 bytes)
0B	0F	00	13	00	0A	02	CD	01	CRC

Response

The normal response to a function 16 query is to echo the address, function code, starting address and number of registers to be loaded.

Node Address	Function Code	Coil Address High	Coil Address Low	Number of Coils High	Number of Coils Low	Byte Count	Force Data High 20...27	Force Data Low 28...29	Error Check Field (2 bytes)
0B	0F	00	13	00	0A	02	CD	01	CRC

Preset Multiple Registers (Function Code 16)

Query

The Function Code allows you to modify the contents of a Modbus 4x range in the slave. This writes up to 125 registers at a time. Since the controller is actively scanning, it can alter the content of any holding register at any time.

- Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.
- This is the structure of the message being sent out to the Modbus network. The following byte values are in hexadecimal display.

The following example is a request to write two registers starting at register 40002 in slave 11.

Node Address	Function Code	Data Start Address High	Data Start Address Low	Number of Points High	Number of Points Low	Byte Count	High Data	Low Data	High Data	Low Data	Error Check Field (2 bytes)
0B	10	00	01	00	02	04	00	0A	01	02	CRC

Response

The normal response to a function 16 query is to echo the address, function code, starting address and number of registers to be loaded.

Node Address	Function Code	Data Start Address High	Data Start Address Low	Number of Points High	Number of Points Low	Error Check Field (2 bytes)
0B	10	00	01	00	02	CRC

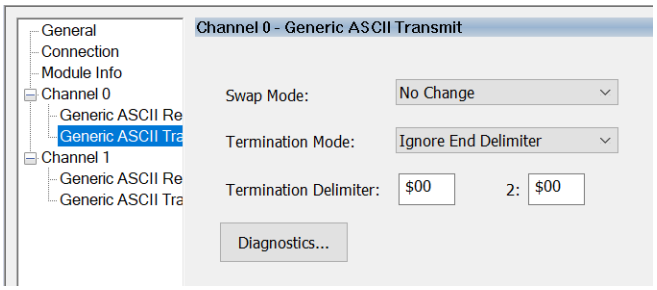
Programming Example

Generic ASCII Sample Code Configuration

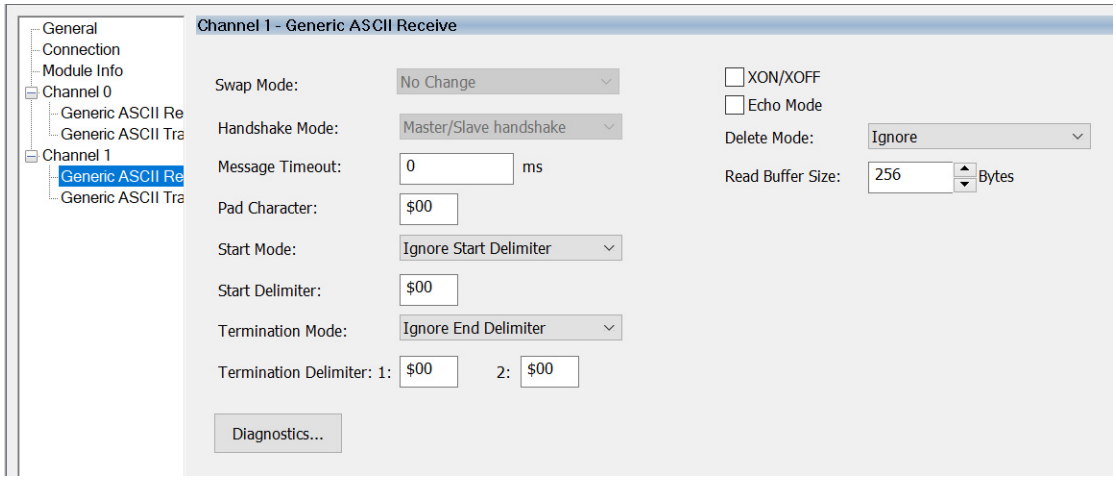
The following images show sample code for Generic ASCII Transmit data and Generic ASCII Receive data.

Generic ASCII Transmit and Receive Channel Configurations

The following image shows the Generic ASCII Transmit Configuration.



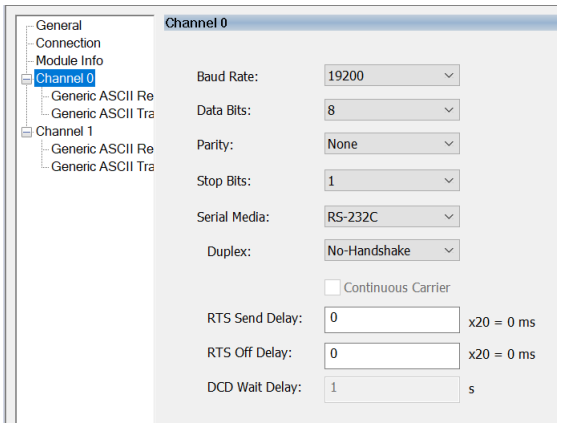
The following image shows the Generic ASCII Receive Configuration.



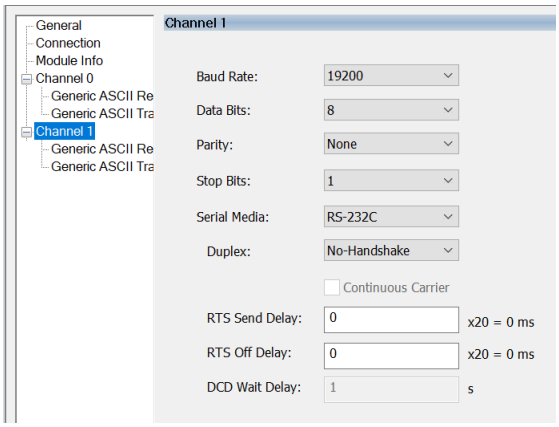
To test the ASCII capabilities, configure Channel 0 to transmit and Channel 1 to receive. On the Module Properties Channel screens, verify that both channels are configured identically for RS-232C. See Figure 27.

Figure 27 - Generic ASCII Channel 0 to Channel 1 Communication

General ASCII Transmit

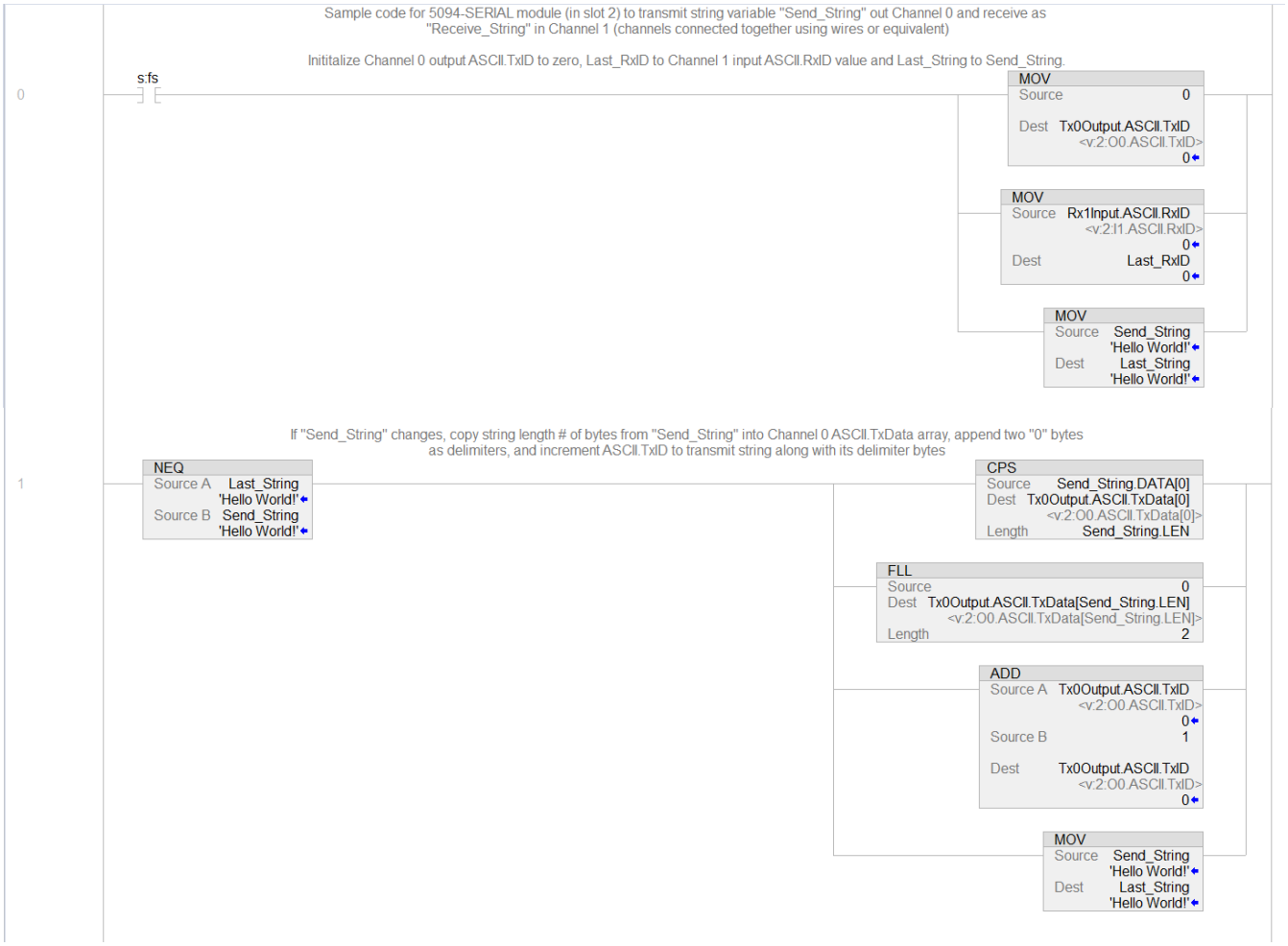


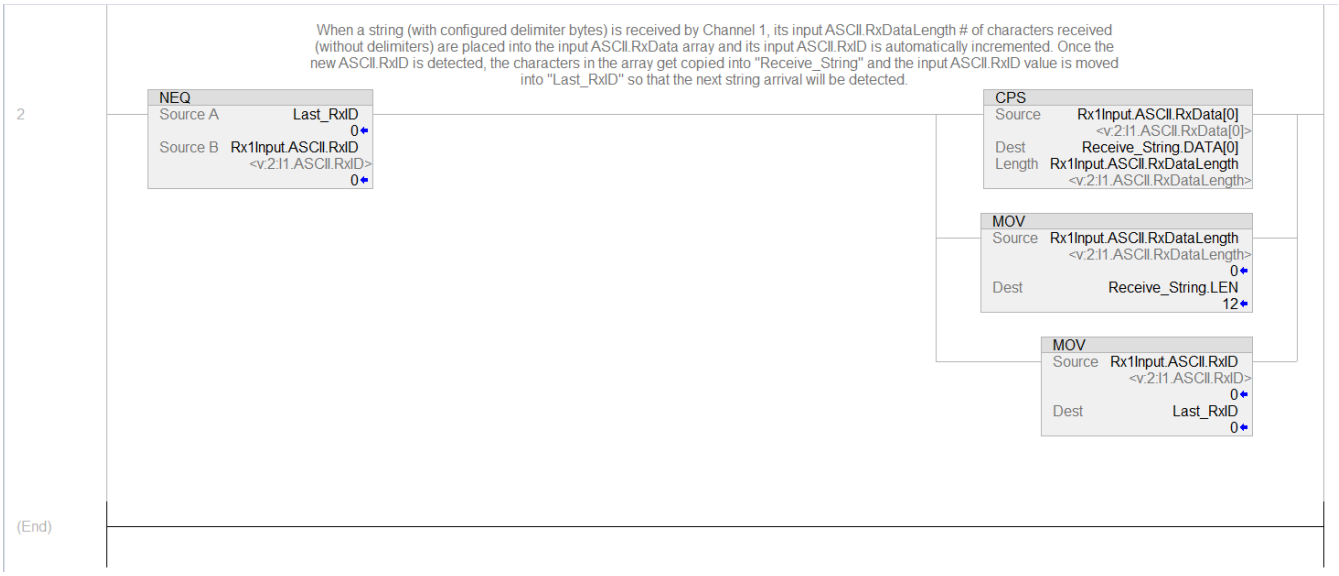
General ASCII Receive



Generic ASCII Sample Code

IMPORTANT The ASCII protocol does not support a retry mechanism in the event of data loss (for example, noise). To manage data loss in your application, you must implement a retry mechanism in your application logic.

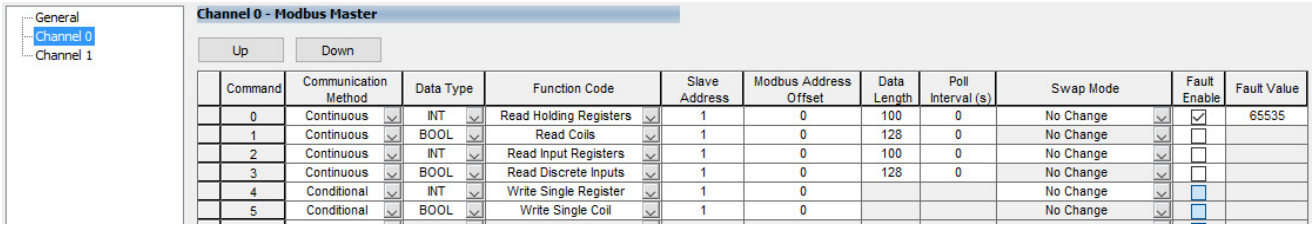




Modbus Sample Code Configuration

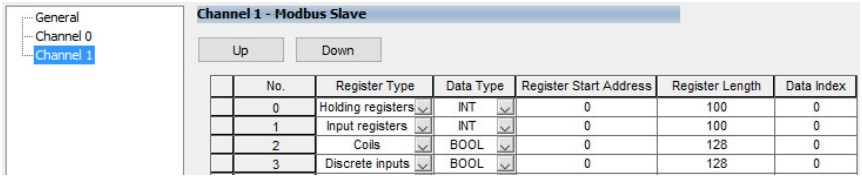
The following images show sample code configuration for Modbus Master and Modbus Slave communication.

Modbus Master Command List



	Command	Communication Method	Data Type	Function Code	Slave Address	Modbus Address Offset	Data Length	Poll Interval (s)	Swap Mode	Fault Enable	Fault Value
	0	Continuous	INT	Read Holding Registers	1	0	100	0	No Change	☑	65535
	1	Continuous	BOOL	Read Coils	1	0	128	0	No Change	☐	
	2	Continuous	INT	Read Input Registers	1	0	100	0	No Change	☐	
	3	Continuous	BOOL	Read Discrete Inputs	1	0	128	0	No Change	☐	
	4	Conditional	INT	Write Single Register	1	0			No Change	☐	
	5	Conditional	BOOL	Write Single Coil	1	0			No Change	☐	

Modbus Slave Address Mapping Table



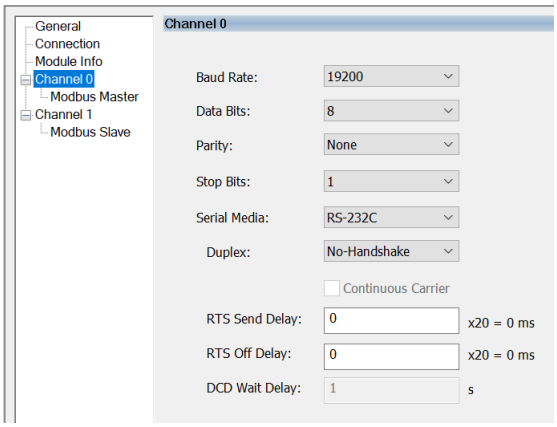
	No.	Register Type	Data Type	Register Start Address	Register Length	Data Index
	0	Holding registers	INT	0	100	0
	1	Input registers	INT	0	100	0
	2	Coils	BOOL	0	128	0
	3	Discrete inputs	BOOL	0	128	0

Modbus Sample Code Configuration Example

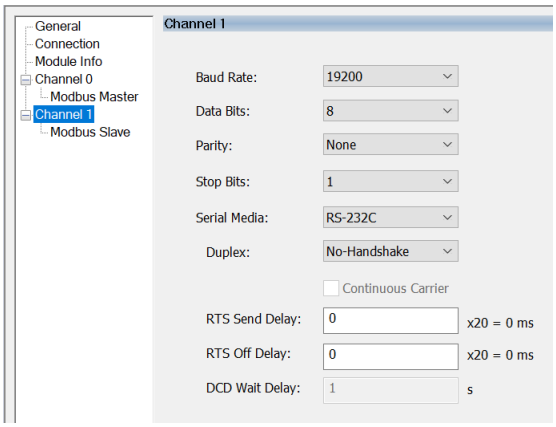
To test the Modbus capabilities, configure Channel 0 for Modbus Master and Channel 1 for Modbus Slave. On the Module Properties Channel screens, verify that both channels are configured identically for RS-232C. See Figure 28.

Figure 28 - Modbus Channel 0 to Channel 1 Communication

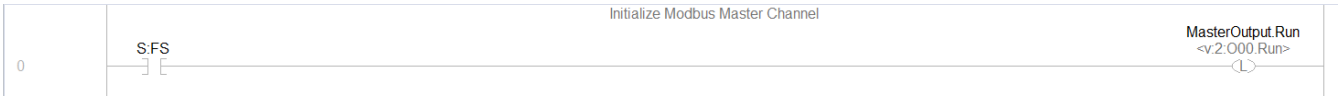
Modbus Master



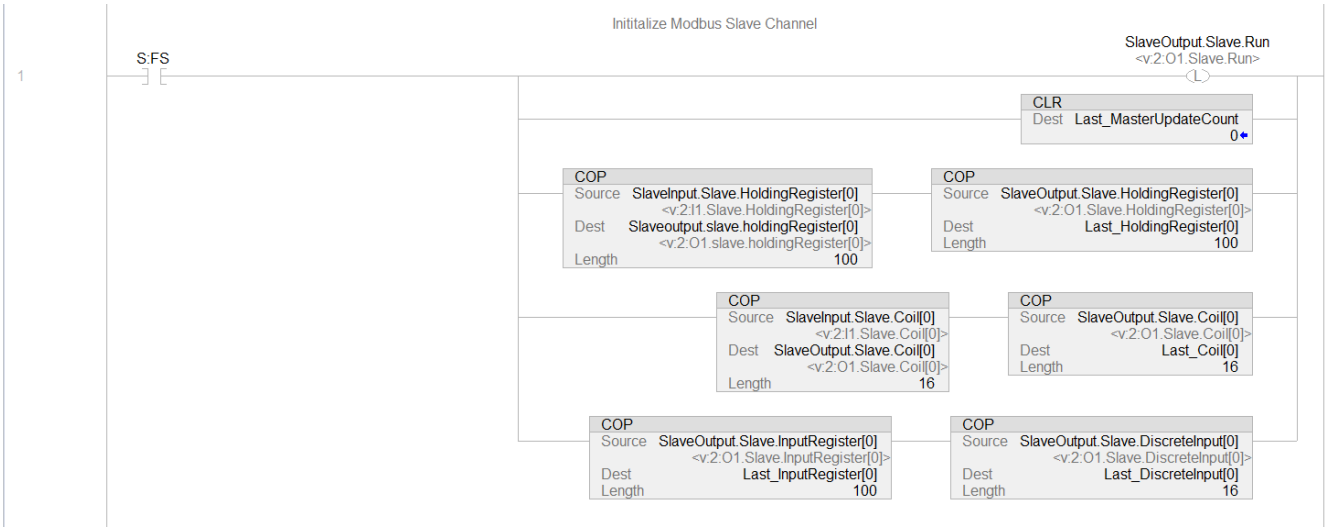
Modbus Slave

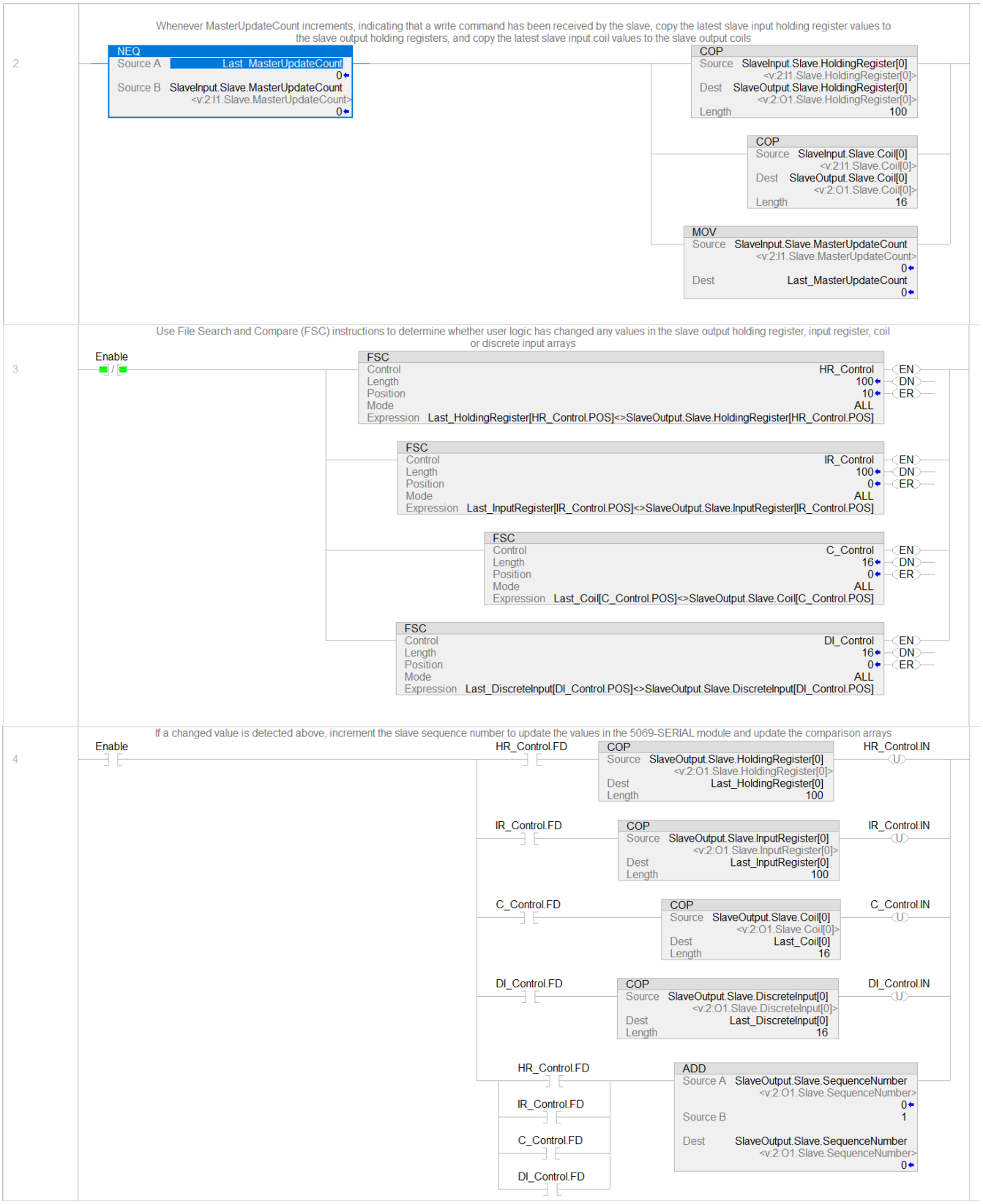


Modbus Master Sample Code



Modbus Slave Sample Code




2
Whenever MasterUpdateCount increments, indicating that a write command has been received by the slave, copy the latest slave input holding register values to the slave output holding registers, and copy the latest slave input coil values to the slave output coils
NEQ
Source A Last_MasterUpdateCount
0
Source B SlaveInput.Slave.MasterUpdateCount
<v:2:I1.Slave.MasterUpdateCount>
0
COP
Source SlaveInput.Slave.HoldingRegister[0]
<v:2:I1.Slave.HoldingRegister[0]>
Dest SlaveOutput.Slave.HoldingRegister[0]
<v:2:O1.Slave.HoldingRegister[0]>
Length 100
COP
Source SlaveInput.Slave.Coil[0]
<v:2:I1.Slave.Coil[0]>
Dest SlaveOutput.Slave.Coil[0]
<v:2:O1.Slave.Coil[0]>
Length 16
MOV
Source SlaveInput.Slave.MasterUpdateCount
<v:2:I1.Slave.MasterUpdateCount>
0
Dest Last_MasterUpdateCount
0
3
Use File Search and Compare (FSC) instructions to determine whether user logic has changed any values in the slave output holding register, input register, coil or discrete input arrays
Enable
FSC
Control HR_Control
Length 100
Position 10
Mode ALL
Expression Last_HoldingRegister[HR_Control.POS]<>SlaveOutput.Slave.HoldingRegister[HR_Control.POS]
EN
DN
ER
FSC
Control IR_Control
Length 100
Position 0
Mode ALL
Expression Last_InputRegister[IR_Control.POS]<>SlaveOutput.Slave.InputRegister[IR_Control.POS]
FSC
Control C_Control
Length 16
Position 0
Mode ALL
Expression Last_Coil[C_Control.POS]<>SlaveOutput.Slave.Coil[C_Control.POS]
FSC
Control DI_Control
Length 16
Position 0
Mode ALL
Expression Last_DiscreteInput[DI_Control.POS]<>SlaveOutput.Slave.DiscreteInput[DI_Control.POS]
4
If a changed value is detected above, increment the slave sequence number to update the values in the 5069-SERIAL module and update the comparison arrays
Enable
HR_Control.FD
COP
Source SlaveOutput.Slave.HoldingRegister[0]
<v:2:O1.Slave.HoldingRegister[0]>
Dest Last_HoldingRegister[0]
Length 100
HR_Control.IN
IR_Control.FD
COP
Source SlaveOutput.Slave.InputRegister[0]
<v:2:O1.Slave.InputRegister[0]>
Dest Last_InputRegister[0]
Length 100
IR_Control.IN
C_Control.FD
COP
Source SlaveOutput.Slave.Coil[0]
<v:2:O1.Slave.Coil[0]>
Dest Last_Coil[0]
Length 16
C_Control.IN
DI_Control.FD
COP
Source SlaveOutput.Slave.DiscreteInput[0]
<v:2:O1.Slave.DiscreteInput[0]>
Dest Last_DiscreteInput[0]
Length 16
DI_Control.IN
HR_Control.FD
IR_Control.FD
C_Control.FD
DI_Control.FD
ADD
Source A SlaveOutput.Slave.SequenceNumber
<v:2:O1.Slave.SequenceNumber>
0
Source B 1
Dest SlaveOutput.Slave.SequenceNumber
<v:2:O1.Slave.SequenceNumber>
0

Notes:

ASCII Conversion Tables

ASCII Conversions

The following table defines the conversions between decimal, octal, hex, and binary values and the ASCII character or control associated with that value.

Table 29 - ASCII Conversions

Decimal	Octal	Hex	Binary	ASCII Character or Control		Decimal	Octal	Hex	Binary	ASCII Character or Control
0	0	00	0000000	Control Shift P, NUL		28	34	1C	0011100	Control Shift L, FS
1	1	01	0000001	Control A, SOH		29	35	1D	0011101	Control Shift M, GS
2	2	02	0000010	Control B, STX		30	36	1E	0011110	Control Shift N, RS
3	3	03	0000011	Control C, ETX		31	37	1F	0011111	Control Shift O, US
4	4	04	0000100	Control D, EOT		32	40	20	0100000	Space, SP
5	5	05	0000101	Control E, ENQ		33	41	21	0100001	!
6	6	06	0000110	Control F, ACK		34	42	22	0100010	"
7	7	07	0000111	Control G, Rings bell		35	43	23	0100011	#
8	10	08	0001000	Control H, Backspace on some terminals		36	44	24	0100100	$
9	11	09	0001001	Control I, Horizontal tab on some terminals		37	45	25	0100101	%
10	12	0A	0001010	Control J, Line feed		38	46	26	0100110	&
11	13	0B	0001011	Control K, VT		39	47	27	0100111	'
12	14	0C	0001000	Control L, Form feed on some terminals		40	50	28	0101000	(
13	15	0D	0001101	Control M, Carriage return		41	51	29	0101001	)
14	16	0E	0001110	Control N, SO		42	52	2A	0101010	*
15	17	0F	0001111	Control O, SI		43	53	2B	0101011	+
16	20	10	0010000	Control P, DLE		44	54	2C	0101100	,
17	21	11	0010001	Control Q, DC1		45	55	2D	0101101	-
18	22	12	0010010	Control R, DC2		46	56	2E	0101110	.
19	23	13	0010011	Control S, DC3		47	57	2F	0101111	/
20	24	14	0010100	Control T, DC4		48	60	30	0110000	0
21	25	15	0010101	Control U, NAK		49	61	31	0110001	1
22	26	16	0010110	Control V, SYN		50	62	32	0110010	2
23	27	17	0010111	Control W, EB		51	63	33	0110011	3
24	30	18	0011000	Control X, CAN		52	64	34	0110100	4
25	31	19	0011001	Control Y, EM		53	65	35	0110101	5
26	32	1A	0011010	Control Z, SUB		54	66	36	0110110	6
27	33	1B	0011011	Control Shift K, ESC		55	67	37	0110111	7
56	70	38	0111000	8		92	134	5C	1011100	\
57	71	39	0111001	9		93	135	5D	1011101	]
58	72	3A	0111010	:		94	136	5E	1011110	^
59	73	3B	0111011	;		95	137	5F	1011111	_
60	74	3C	0111100	<		96	140	60	1100000	`
61	75	3D	0111101	=		97	141	61	1100001	a
62	76	3E	0111110	>		98	142	62	1100010	b
63	77	3F	0111111	?		99	143	63	1100011	c

Table 29 - ASCII Conversions (Continued)

Decimal	Octal	Hex	Binary	ASCII Character or Control		Decimal	Octal	Hex	Binary	ASCII Character or Control
64	100	40	1000000	@		100	144	64	1100100	d
65	101	41	1000001	A		101	145	65	1100101	e
66	102	42	1000010	B		102	146	66	1100110	f
67	103	43	1000011	C		103	147	67	1100111	g
68	104	44	1000100	D		104	150	68	1101000	h
69	105	45	1000101	E		105	151	69	1101001	i
70	106	46	1000110	F		106	152	6A	1101010	j
71	107	47	1000111	G		107	153	6B	1101011	k
72	110	48	1001000	H		108	154	6C	1101100	l
73	111	49	1001001	I		109	155	6D	1101101	m
74	112	4A	1001010	J		110	156	6E	1101110	n
75	113	4B	1001011	K		111	157	6F	1101111	o
76	114	4C	1001100	L		112	160	70	1110000	p
77	115	4D	1001101	M		113	161	71	1110001	w
78	116	4E	1001110	N		114	162	72	1110010	r
79	117	4F	1001111	O		115	163	73	1110011	s
80	120	50	1010000	P		116	164	74	1110100	t
81	121	51	1010001	Q		117	165	75	1110101	u
82	122	52	1010010	R		118	166	76	1110110	v
83	123	53	1010011	S		119	167	77	1110111	w
84	124	54	1010100	T		120	170	78	1111000	X
85	125	55	1010101	U		121	171	79	1111001	y
86	126	56	1010110	V		122	172	7A	1111010	z
87	127	57	1010111	W		123	173	7B	1111011	{
88	130	58	1011000	X		124	174	7C	1111100	|
89	131	59	1011001	Y		125	175	7D	1111101	}
90	132	5A	1011010	Z		126	176	7E	1111110	~
91	133	5B	1011011	[		127	177	7F	1111111	DEL

Notes:

Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Knowledgebase	Access Knowledgebase articles.	rok.auto/knowledgebase
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Download firmware, associated files (such as AOP, EDS, and DTM), and access product release notes.	rok.auto/pcdc

Documentation Feedback

Your comments help us serve your documentation needs better. If you have any suggestions on how to improve our content, complete the form at rok.auto/docfeedback.

Waste Electrical and Electronic Equipment (WEEE)

At the end of life, this equipment should be collected separately from any unsorted municipal waste.

Rockwell Automation maintains current product environmental compliance information on its website at rok.auto/pec.

Rockwell Otomasyon Ticaret A.Ş. Kar Plaza İş Merkezi E Blok Kat:6 34752, İçerenköy, İstanbul, Tel: +90 (216) 5698400 EEE Yönetmeliğine Uygundur

Connect with us.

rockwellautomation.com — expanding **human possibility**™

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
ASIA PACIFIC: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 5094-UM006A-EN-P - October 2020