User Manual

Original Instructions

Allen-Bradley

Compact 5000 I/O Serial Module

Catalog Number 5069-SERIAL

Allen-Bradley • Rockwell Software

Rockwell Automation

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).

Preface **7**
Additional Resources 7

Chapter 1

Compact 5000 I/O Serial Module

Compact 5000 I/O Serial Module Overview. 11
About the Module. 12

Chapter 2

Compact 5000 I/O Serial Module Operation in a Logix 5000 Control System

Controller and Software Compatibility 13
Local I/O or Remote I/O Modules. 14
Local I/O Module 14
Remote I/O Module. 15
Ownership 16
Construct a System That Uses a Compact 5000 I/O Serial Module 16
Power Compact 5000 I/O Serial Module. 17
Configure Compact 5000 I/O Modules. 18
Connections 18
5069-ARM and 5069-FPD Modules. 21
5069-ARM Address Reserve Module. 21
5069-FPD Field Potential Distributor. 22

Chapter 3

Compact 5000 I/O Serial Module Features

Purpose of the Module 23
General Module Features 23
- Software Configurable 24
- Requested Packet Interval 24
- Fault and Status Reporting 25
- Module Inhibiting 26
- Electronic Keying 27
- Status Indicators 28
- Module Firmware 28
- Common Module Functions 29
- Control Line Menu 30

Data Exchange 31
- Generic ASCII Data Exchange 31
- Data Sent with the Serial Port 31
- Generic ASCII Transmit Functions 32
- Generic ASCII Receive Functions 33
- Data Received from the Serial Port in Immediate Mode 35
- Data Received from the Serial Port in Master/Slave Handshake Mode 36

Modbus Master Data Exchange 37
- Modbus Master Write Command 37
- Modbus Master Read Command 38

Modbus Slave Data Exchange 39
- Modbus Slave Write Command 39
- Modbus Slave Read Command 40
- Modbus Master Functions 41
- Modbus Slave Functions 42

Chapter 4

Configure Compact 5000 I/O Serial Module

Before You Begin 43
Create a New Module 44
- Discover Local I/O Modules 44
- New Local I/O Modules 47
- Discover Remote I/O Modules 49
- New Remote I/O Module 51

Edit the Module Configuration 54
- General Category 54
- Connection Category 63
- Module Info Category 68

View the Module Tags 69

Chapter 5

Troubleshoot Your Module

Module Status Indicator 71
Compact 5000 I/O Serial Module Status Indicators 73

Appendix A

Module Tags

Name Conventions 75
Generic ASCII and Modbus Slave Name Conventions 75
Modbus Master Name Conventions 76
Access the Tags 76
Channel Configured Generic ASCII Tags 77
Channel Configured for Generic ASCII 78
Input Tags 78
Output Tags 81
Channel Configured for Modbus Master 82
Input Tags 82
Output Tags 85
Channel Configured for Modbus Slave 86
Input Tags 87
Output Tags 88

Appendix B

Master Command List

Master Command List Function Codes 91
Read Coil Status (Function Code 01) 91
Read Input Status (Function Code 02) 93
Read Holding Registers (Function Code 03) 94
Read Input Registers (Function Code 04) 95
Force Single Coil (Function Code 05) 96
Preset Single Register (Function Code 06) 98
Force Multiple Coils (Function Code 15) 99
Preset Multiple Registers (Function Code 16) 100

Appendix C

Programming Example

Generic ASCII Sample Code Configuration 101
Generic ASCII Transmit and Receive Channel Configurations 101
Generic ASCII Sample Code 102
Modbus Sample Code Configuration 104
Modbus Master Command List 104
Modbus Slave Address Mapping Table 104
Modbus Sample Code Configuration Example 104
Modbus Master Sample Code 105
Modbus Slave Sample Code 105

ASCII Conversion Tables

Appendix D

ASCII Conversions 107

Index 109

This manual describes how to use Compact 5000™ I/O Serial modules in Logix 5000™ systems.

Make sure that you are familiar with the following:

- Use of a Logix 5000™ controller
- Use of an EtherNet/IP network, if the serial module is installed in a remote location that is accessible via the EtherNet/IP network.
- Studio 5000 Logix Designer® environment.

IMPORTANT The Compact 5000 I/O Serial module is only compatible with the following controllers:

- CompactLogix™ 5380
- Compact GuardLogix® 5380
- ControlLogix® 5580
- GuardLogix® 5580

Additional Resources

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

Resource	Description
Compact 5000 I/O Serial Module Install Instructions, publication 5069-IN022	Describes how to install the Compact 5000 I/O Serial Module.
5069 Compact I/O™ EtherNet/IP Adapter Installation Instructions, publication 5069-IN003	Describes how to install and wire the Compact 5000 I/O EtherNet/IP adapters.
Compact 5000 I/O Modules Specifications Technical Data, publication 5069-TD001	Provides specifications, wiring diagrams, and module block diagrams Compact 5000 I/O modules
5000 Series Digital I/O Modules in Logix5000 Control Systems User Manual, publication 5000-UM004	Describes how to use 5000 Series digital I/O modules.
5000 Series Analog I/O Modules in Logix 5000 Control Systems User Manual, publication 5000-UM005	Describes how to use Compact 5000 I/O analog modules.
5000 Series High-speed Counter Modules in Logix 5000 Control Systems User Manual, publication 5000-UM006	Describes how to use the Compact 5000 I/O.
EtherNet/IP Communication Modules in 5000 Series Control Systems User Manual, publication ENET-UM004	Describes how to use Compact 5000 I/O EtherNet/IP adapters.
CompactLogix 5380 and Compact GuardLogix 5380 Controllers User Manual, publication 5069-UM001	Describes how to use CompactLogix 5380 and Compact GuardLogix 5380 controllers.
ControlLogix 5580 and GuardLogix 5580 Controllers User Manual, publication 1756-UM543	Describes how to use ControlLogix 5580 and GuardLogix 5580 controllers.
Electronic Keying in Logix 5000 Control Systems Application Technique, publication LOGIX-AT001	Describes how to use electronic keying in Logix 5000 control system applications.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Product Certifications website, http://www.rockwellautomation.com/global/certification/overview.page	Provides declarations of conformity, certificates, and other certification details.

You can view or download publications at http://www.rockwellautomation.com/global/literature-library/overview.page.

To order paper copies of technical documentation, contact your local Allen-Bradley distributor or Rockwell Automation sales representative.

Notes:

Compact 5000 I/O Serial Module

Compact 5000 I/O Serial Module Overview

The Compact 5000™ I/O 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 module has two channels that are independent of each other. 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)

The serial module can reside locally in a CompactLogix™ 5380 system or a Compact GuardLogix® 5380 system. The module can also reside in a remote Compact 5000 I/O system accessible via an EtherNet/IP network.

IMPORTANT Use Studio 5000 Logix Designer® Version 31 or greater. You must install an Add-On Profile to use the serial module. To find the Add-On Profile go to the Product Compatibility and Download Center (PCDC).

For more information on how a Compact 5000 I/O serial module can function in a control system, see Chapter 2, Compact 5000 I/O Serial Module Operation in a Logix 5000 Control System.

About the Module

The module provides the communication connections to the serial devices.

Figure 1 - Example Compact 5000 I/O Serial Module

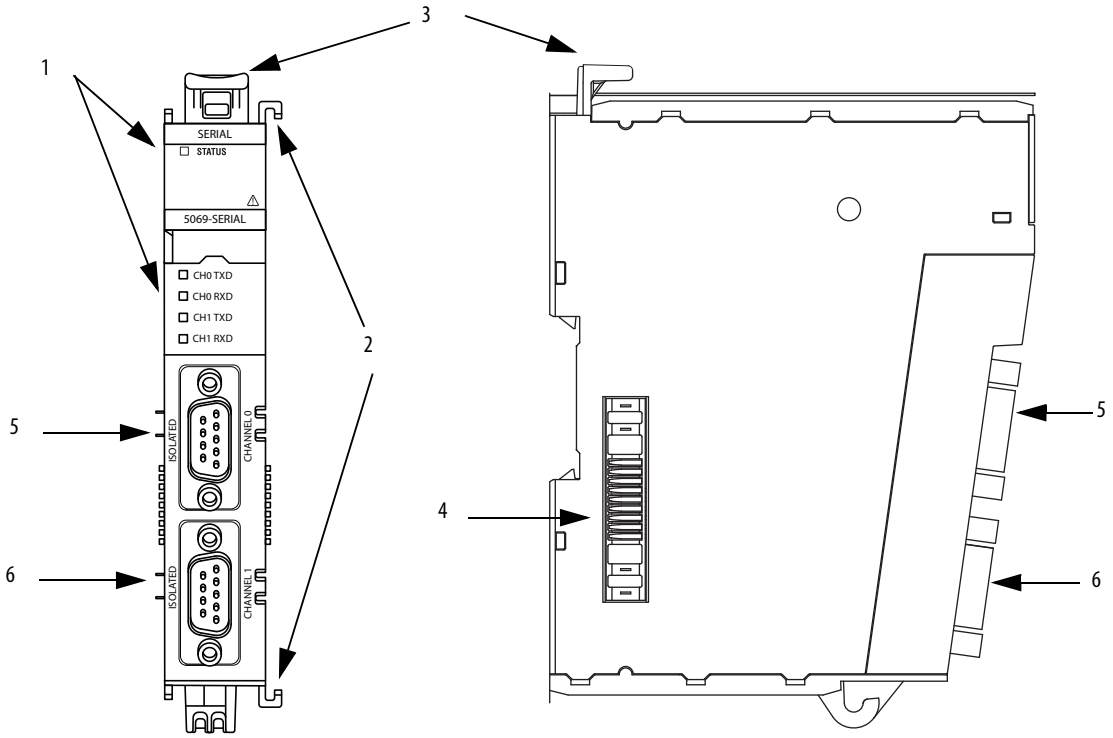


Table 1 - Compact 5000 I/O Serial Module Parts

Item	Item Name	Description
1	Status Indicators	Displays the status of communication, module health, and input/output devices. Indicators help with troubleshooting anomalies.
2	Interlocking side pieces	Securely installs Compact 5000 I/O serial modules in the system.
3	DIN rail latch	Secures the module on the DIN rail.
4	MOD Power bus and SA Power bus connectors	Pass system-side and field-side power across the internal circuitry of the I/O modules in a Compact 5000 I/O system. The connectors are isolated from each other.
5	Channel 0	Channel 0 isolated serial port.
6	Channel 1	Channel 1 isolated serial port.

Compact 5000 I/O Serial Module Operation in a Logix 5000 Control System

Topic	Page
Controller and Software Compatibility	13
Compact 5000 I/O Serial Module as Remote I/O Module	15
Power Compact 5000 I/O Serial Module	17
Configure Compact 5000 I/O Modules	18

Controller and Software Compatibility

These compatibility requirements apply when you use Compact 5000™ I/O serial module in Logix 5000™ controller control system:

- The Compact 5000 I/O Serial module is compatible with the following controllers:
  - CompactLogix™ 5380
  - Compact GuardLogix® 5380
  - ControlLogix® 5580
  - GuardLogix® 5580
- The manner in which you use the serial module affects controller compatibility.
- You can use Compact 5000 I/O serial modules as local or remote I/O modules.

IMPORTANT The serial module is not compatible with the 5069-AEN2TR. See the Product Compatibility and Download Center (PCDC) for more information.

Local I/O or Remote I/O Modules

You can use a Compact 5000 I/O Serial module as a local or remote I/O module.

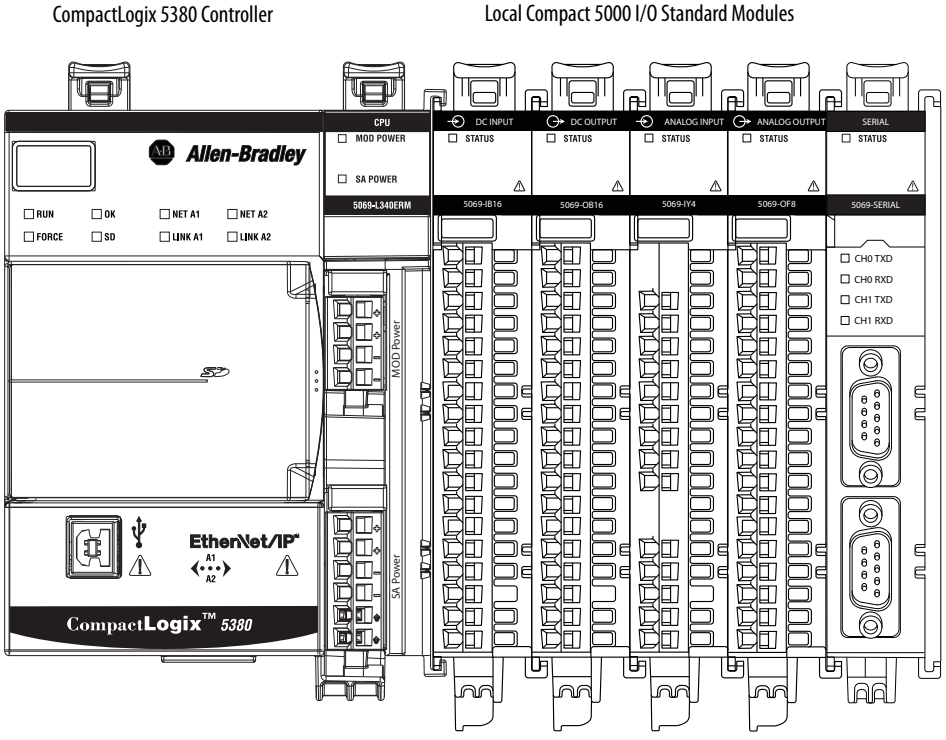
Local I/O Module

When Compact 5000 I/O Serial module resides in the same system as the controller, it is a local I/O module. Local I/O modules are installed to the right of the controller and exchange data with the controller over the system backplane.

IMPORTANT Compact 5000 I/O Serial modules can function as local I/O modules in the following:

- CompactLogix 5380 control systems
- Compact GuardLogix 5380 control systems.

Figure 2 - Compact 5000 I/O Serial Module as Local I/O Module



Remote I/O Module

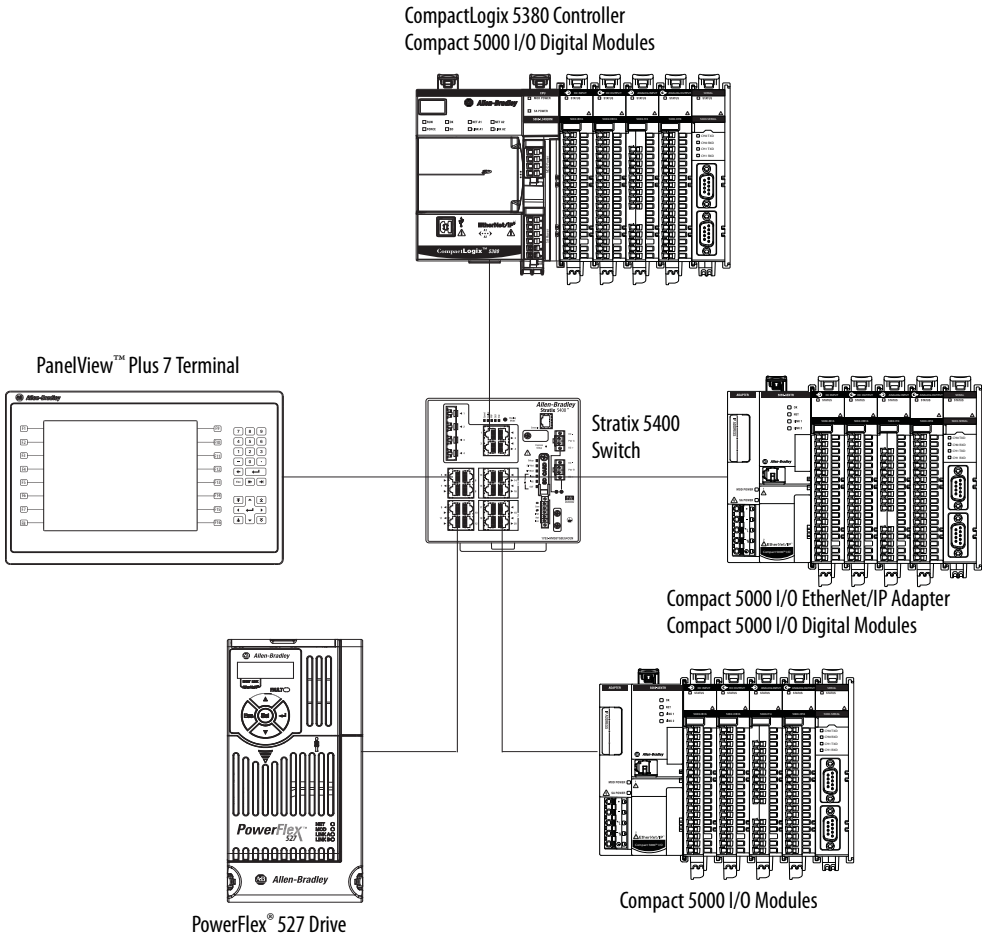
When a Compact 5000 I/O Serial module resides in a separate location from the controller, it is a remote I/O module. Remote Compact 5000 I/O modules are accessible over an EtherNet/IP network via a Compact 5000 I/O EtherNet/IP adapter.

Remote Compact 5000 I/O modules are installed to the right of the adapter and exchange data across the remote system backplane. The data is exchanged with the controller over the EtherNet/IP network.

IMPORTANT Compact 5000 I/O Serial modules can function as remote I/O modules in the following:

- CompactLogix 5380 control systems
- Compact GuardLogix 5380 control systems
- ControlLogix 5580 control systems
- GuardLogix 5580 control systems

Figure 3 - Compact 5000 I/O Serial Module as Remote I/O Module



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 Compact 5000 I/O modules are 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 Compact 5000 I/O system.
- Sends the I/O module configuration data to define module behavior and begin operation in the control system.

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

Construct a System That Uses a Compact 5000 I/O Serial Module

Before you use your serial module, you must complete tasks based on the way that you use the modules:

- Local I/O modules - Complete the following:
  a. Install a controller that can use the serial modules local I/O modules.
  b. Install the serial module to the right of the controller.

- Remote I/O modules - Complete the following:
  a. Install an adapter that is compatible with remote Compact 5000 I/O standard and/or safety modules via an EtherNet/IP network.
  b. Install the serial module to the right of the adapter.
  c. Install an EtherNet/IP network.
  d. Install the controller that accesses the serial modules via an EtherNet/IP network.

Power Compact 5000 I/O Serial Module

Compact 5000 I/O serial modules receive the following power types:

- System-side Power - Powers the system and lets modules transfer data and execute logic.

  System-side power is also known as MOD power.
- Field-side Power - Powers field-side devices that are connected to some Compact 5000 I/O modules.

  Field-side power is also known as SA power.

IMPORTANT The Compact 5000 I/O Serial Module does not use SA power. However, verify that the module is installed in a position where SA power uses DC voltage.

Power begins at the left-most device in the system and passes across the Compact 5000 I/O module internal circuitry via power buses, that is, a MOD power bus. The left-most device is either a controller or an EtherNet/IP adapter, depending on whether the Serial module is a local or remote I/O module.

For more information on how to power local Compact 5000 I/O modules, see the CompactLogix 5380 and Compact GuardLogix 5380 Controllers User Manual, publication 5069-UM001.

For more information on how to power remote Compact 5000 I/O modules, see the EtherNet/IP Communication Modules in 5000 Series Systems User Manual, publication ENET-UM004.

IMPORTANT Remember the following:

- A system uses only one MOD power bus that starts at a controller or adapter and passes across all modules installed in the system.
- A system can use more than one SA power bus. The first SA power bus typically starts at the controller or adapter, and 5069-FPD field potential distributors let you establish new SA power buses in the same system.

Configure Compact 5000 I/O Modules

You must create a Logix Designer application project for the Logix 5000 controller that owns the Compact 5000 I/O Serial modules. The project includes module configuration data for the Compact 5000 I/O Serial module.

The Logix Designer application transfers the project to the owner-controller during the program download. Data is then transferred to the Compact 5000 I/O Serial module either across the backplane or over an EtherNet/IP network.

The Compact 5000 I/O modules can operate immediately after receiving the configuration data.

Connections

During module configuration, you must define the module. In the Module Definition parameters for most Compact 5000 I/O modules, you must choose a Connection type. 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 number in the local CompactLogix 5380 or Compact GuardLogix 5380 controller system or remote Compact 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 prevent normal operation.

The owner-controller monitors its connection with a module. Any break in the connection, for example, the loss of power to the Compact 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 Compact 5000 I/O modules that make one connection to the owner-controller, the Compact 5000 I/O Serial module can have multiple connections based on how the module channels are configured.

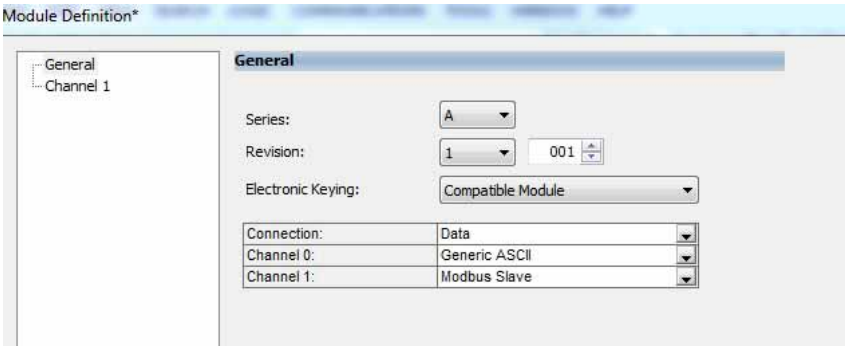
The Serial module is required to use the Data connection type in the Module Definition dialog box. However, the module has two channels that you must configure independently of each other. You can disable a channel or choose a mode, that is, the Generic ASCII, Modbus Master, or Modbus Slave mode.

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

Selected Protocol	Description
Generic ASCII	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

For example, 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

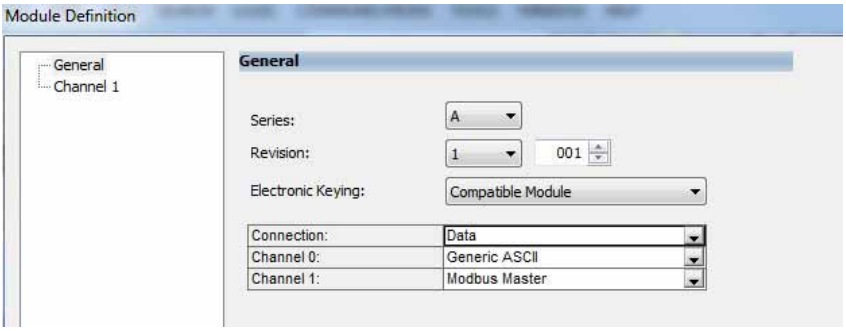


Table 2 shows the total number of connections for all protocol combinations.

Table 2 - I/O Connections For Each Protocol Combination

		CH0 Protocol Choice			
		Disabled	**Generic ASCII**	**Modbus Slave**	**Modbus Master**
CH1 Protocol	**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.

5069-ARM and 5069-FPD Modules

The following Compact 5000 I/O modules are available for unique purposes.

- 5069-ARM Address Reserve Module
- 5069-FPD Field Potential Distributor

5069-ARM Address Reserve Module

The 5069-ARM address reserve module reserves a node address in a Compact 5000 I/O system. The module remains installed until you insert another Compact 5000 I/O module into the same location.

For example, your application can require the use of a 5069-SERIAL module in a specific node location. The module is typically installed when you install the Compact 5000 I/O system. In this case, however, the required 5069-SERIAL module is not available for insertion.

To install Compact 5000 I/O modules, you attach them to left-most device in the system. The node addresses increment as each module is installed. To make sure that the 5069-SERIAL module is installed in the correct location later, you install a 5069-ARM during initial system installation.

When the required Compact 5000 I/O module is available, you remove the 5069-ARM address reserve module and replace it with the 5069-SERIAL module. Thus, you insert the module in the correct node address location.

Figure 4 shows a Compact 5000 I/O system that uses a 5069-ARM address reserve module to reserve a node address.

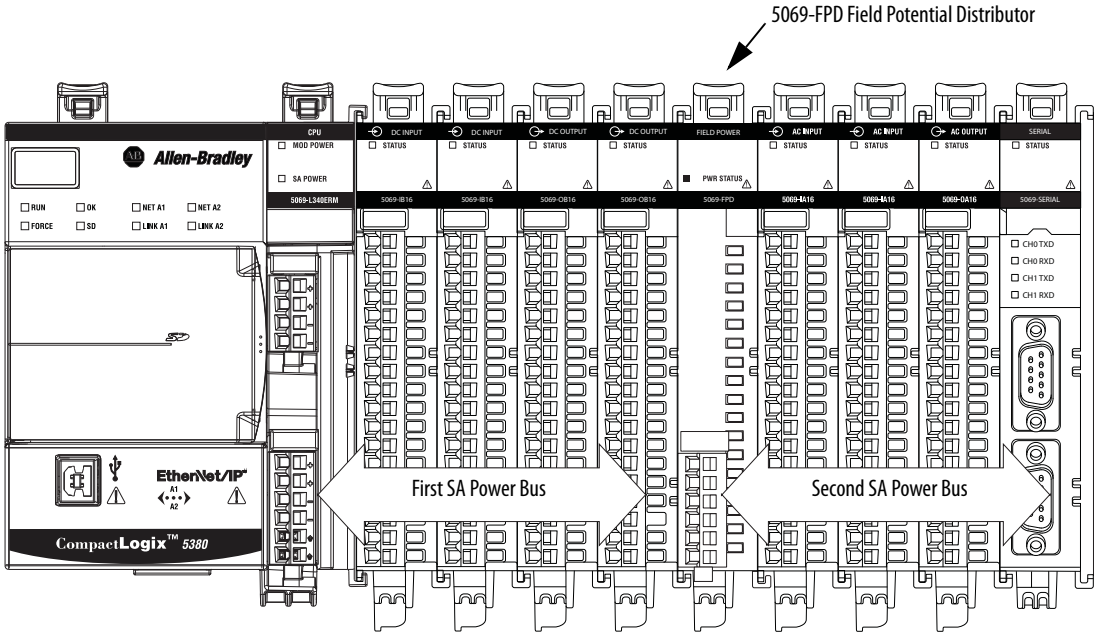
Figure 4 - Compact 5000 I/O System with 5069-ARM Address Reserve Module

5069-FPD Field Potential Distributor

The controller or adapter, based on whether the module is a local or remote I/O module, is the primary source of field-side power, that is, SA power, in the system. However, you can use a 5069-FPD field potential distributor to break field-side power distribution in a system and establish a new SA power bus.

Field-side power passes across the internal circuitry of the Compact 5000 I/O modules beginning with the controller or the adapter. The field potential distributor blocks the passage of field-side power from the left and functions as a new field-side power source for the modules to the right.

Figure 5 - Compact 5000 I/O System with 5069-FPD Field Potential Distributor



For more information on how to power Compact 5000 I/O modules, see the following:

- EtherNet/IP Communication Modules in Logix5000 Control Systems User Manual, publication ENET-UM004.
- CompactLogix 5380 and Compact GuardLogix 5380 Controllers User Manual, publication 5069-UM001.

Compact 5000 I/O Serial Module Features

Topic	Page
Purpose of the Module	23
General Module Features	23
Generic ASCII Data Exchange	31
Modbus Master Data Exchange	37
Modbus Slave Data Exchange	39

Purpose of the Module

The Compact 5000™ I/O Serial Module allows you to select 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 Compact 5000 I/O serial module supports the following module-wide features:

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

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.

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

RPI Range

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

IMPORTANT
- If the RPI is not a multiple of 0.1 ms, round the number down to the closest multiple value of 0.1 ms. For example, if the RPI is 2.37 ms, round to 2.3 ms.
- Depending on the software version, you can set each channel value separately.

Fault and Status Reporting

The Compact 5000 I/O 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 the full chapter, Troubleshoot Your Module on page 71.

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 I/O 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.

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, digital I/O 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.
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 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.

More Information

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

Status Indicators

Each Compact 5000 I/O 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 5, Troubleshoot Your Module on page 71.

Module Firmware

The Compact 5000 I/O 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 through the Product Compatibility and Download Center (PCDC). At the PCDC, you can use the module catalog number to check for firmware updates. If the catalog number is not available, no updates exist.

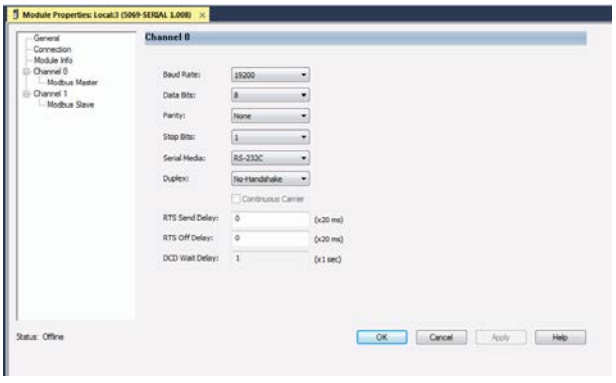
Common Module Functions

The module supports the following terms, definitions, and parameters.

Table 3 - Common Module Function

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
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 = 1)

Selecting Functions in Logix Designer Application



Control Line Menu

When you are required to connect to a dial up modem, see Table 4 explaining the duplex setting in the serial port.

Table 4 - 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 the system are half-duplex.	—	Half-duplex	Clear the continuous carrier checkbox (default).

Data Exchange

Generic ASCII Data Exchange

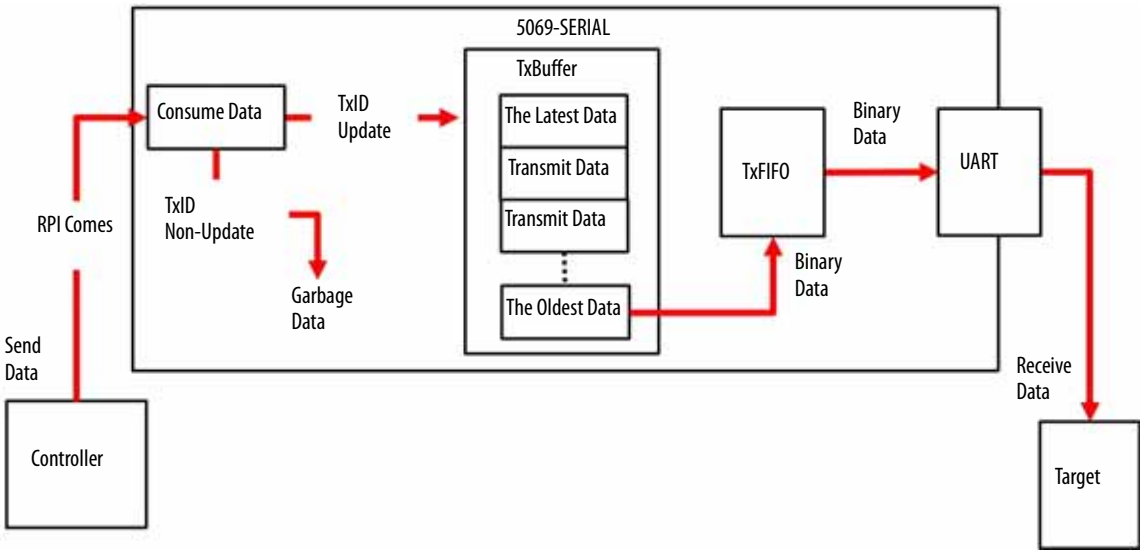
The following illustrations show the different data exchanges using the serial module.

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 6 - 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 5 - 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.

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

Generic ASCII Receive Functions

Table 6 - Generic ASCII Receive Functions

Swap Mode	Swapping is done 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 in order 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) - Receives packet based on number of bytes specified only. • Exclude Delimiter - Determines end of data when it finds the Termination Delimiter characters, but these bytes are not included in the ASCII.RxData input Tag. • Include Delimiter - Termination Delimiters bytes are included in the ASCII.RxData input tag.
Termination Delimiter 1	Configure the channel's termination delimiter.	• 7 bit $00…$7F – Recommendation: When Delete Mode is enabled, do not configure Termination 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)

Generic 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.

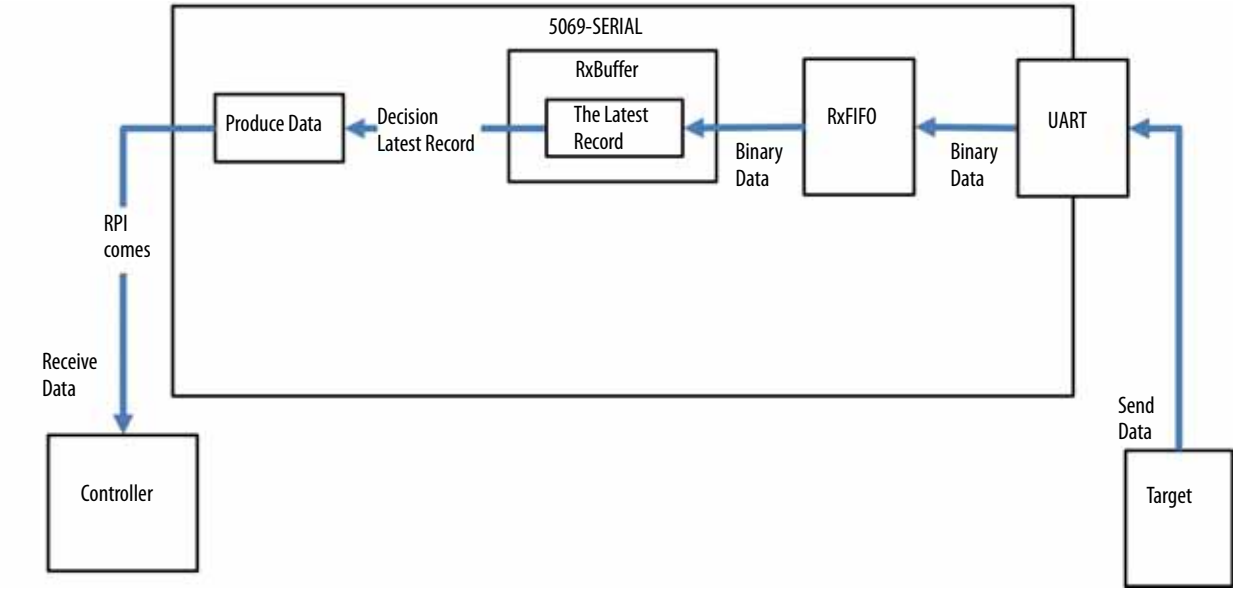
For Generic ASCII Sample Code, see page 101.

Data Received from the 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 7 - Data Received from the Serial Port in Immediate Mode

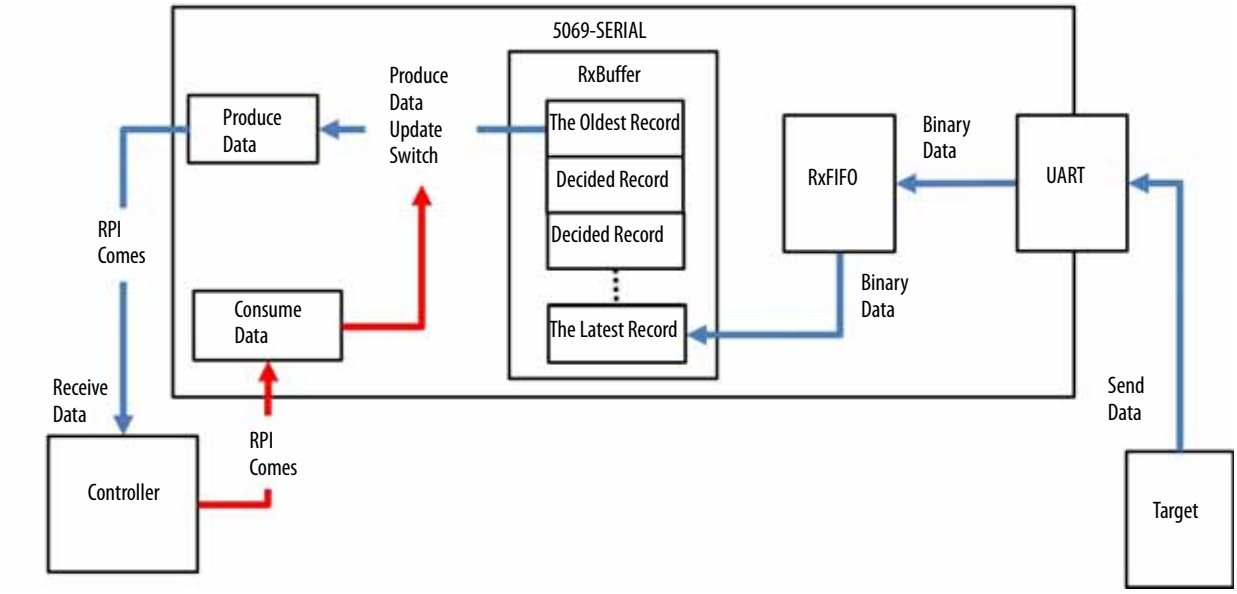


Data Received from the Serial Port in Master/Slave Handshake Mode

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 8 - 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.

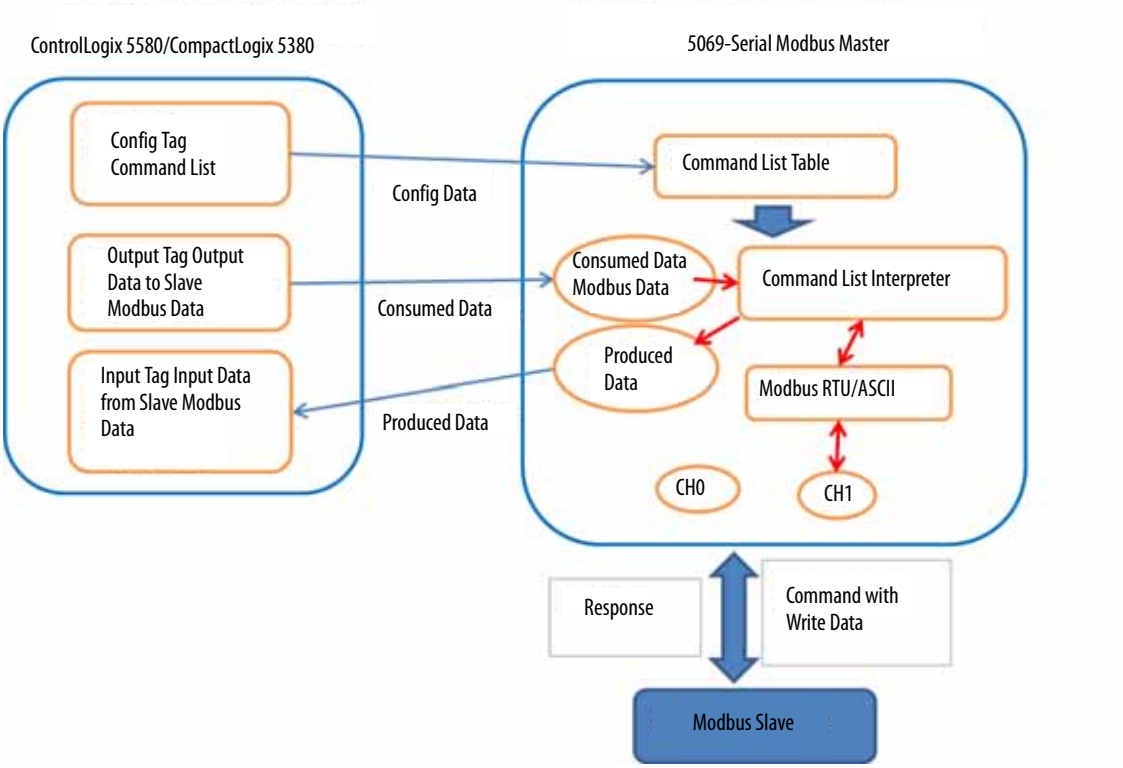
Modbus Master Data Exchange

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

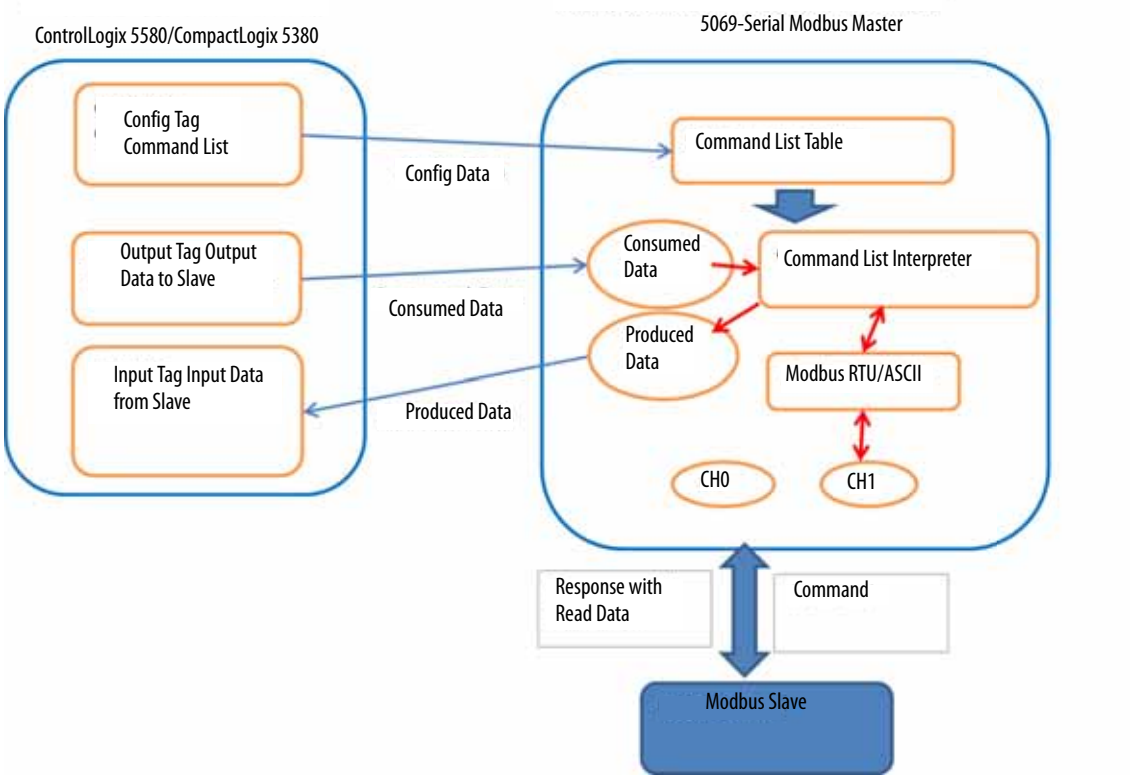
- Write - the Modbus Master writes data to the Modbus Slave.
- Read - the Modbus Master reads data from the Modbus Slave.

The Compact 5000 I/O Serial Module can get Modbus Data from Produced/Consumed Data command, every RPI.

Modbus Master Write Command



Modbus Master Read Command

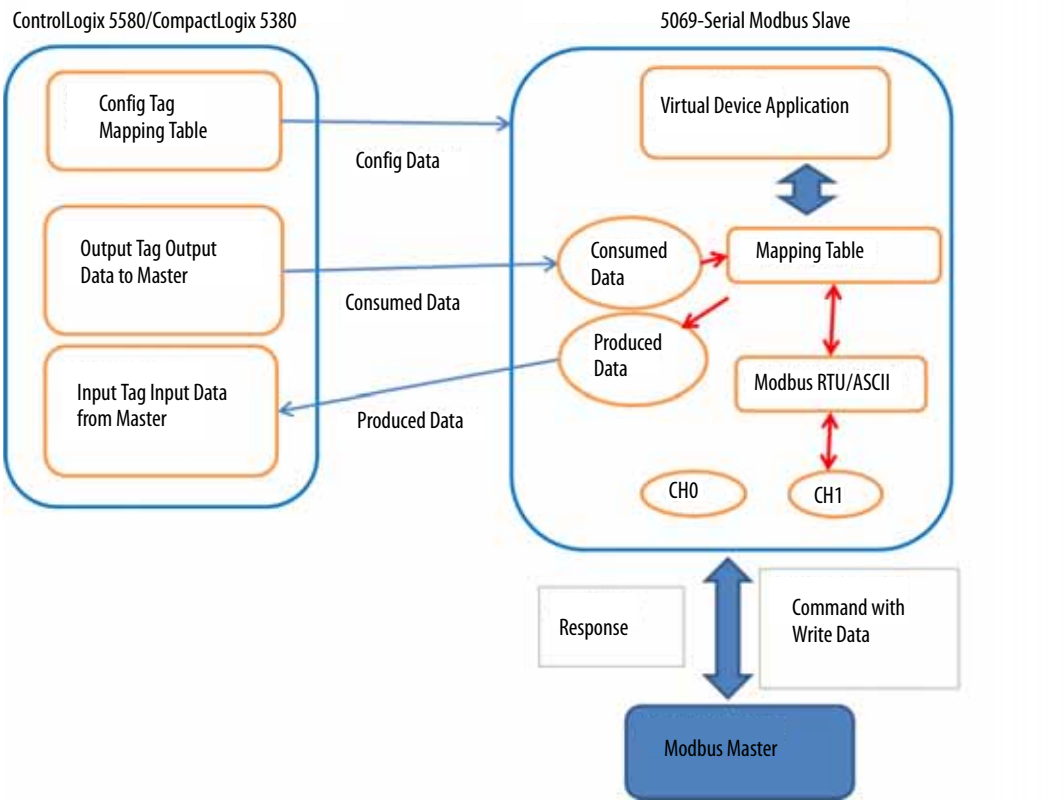


Modbus Slave Data Exchange

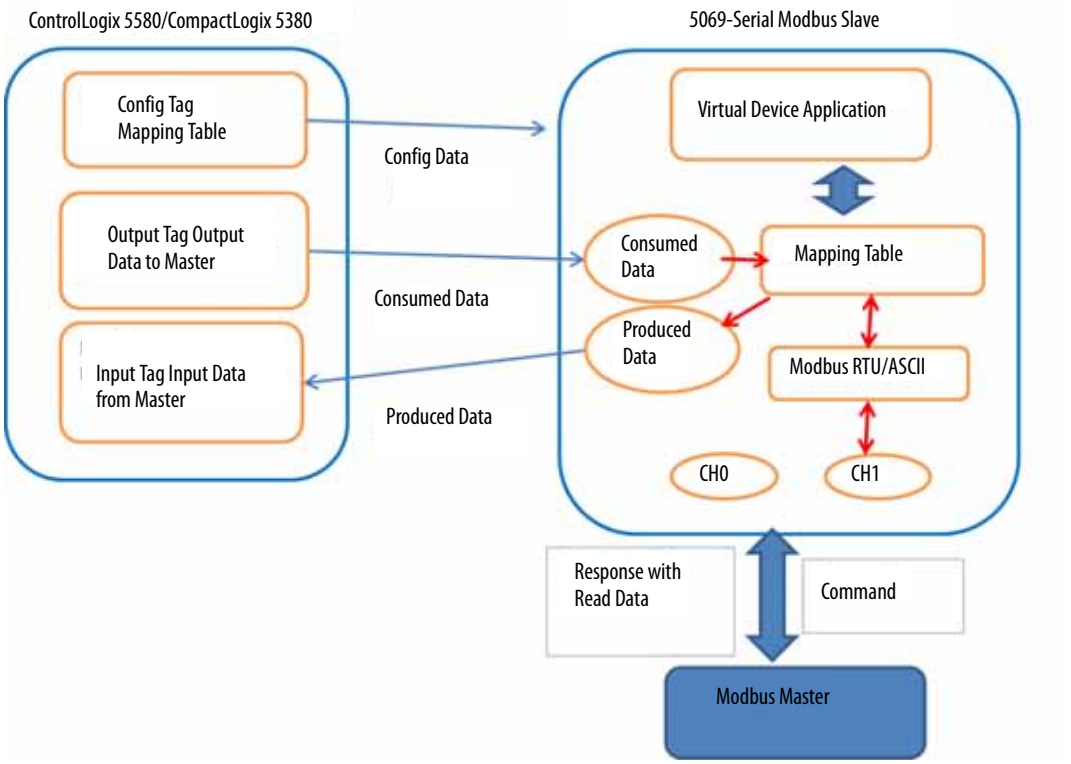
For the Modbus Slave data exchange, the following definitions apply:

- 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 7 - Modbus Master Configuration Parameters

Parameter	Definition	Available Options
Modbus Format	Selecting 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 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 8 and Table 9 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 8 - Inter-frame Timeout Minimum Values (10 Bit)

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

Table 9 - Inter-frame Timeout Minimum Values (11 Bit)

Baud Rate	Default Value	1Byte (11Bit)	3.5t (11Bit)	Legal Range (11Bit)
1200	64000	9166.667	32083.333	32000…65535000 (us)
2400	32000	4583.333	16041.667	16000…65535000 (us)
4800	16000	2291.667	8020.833	8000…65535000 (us)
9600	8000	1145.833	4010.417	4000…65535000 (us)
19200	3500	572.917	2005.208	1750…65535000 (us)
38400	3500	286.458	1002.604	1750…65535000 (us)
57600	3500	190.972	668.403	1750…65535000 (us)
115200	3500	95.486	334.201	1750…65535000 (us)

Modbus Slave Functions

Table 10 - Modbus Slave Configuration Parameters

Parameter	Definition	Available Options
Modbus Format	Selecting 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 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.	• 1…247 (default = 1)
Inter-frame Timeout	Maximum delay time to receive Data of each channel	• 0 is not a valid value. Use Table 8 and Table 9 to determine the minimum value.

For Modbus Master and Modbus Slave Sample code, see page 104.

Configure Compact 5000 I/O Serial Module

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

TIP When a controller establishes a connection to a remote 5069-SERIAL module, it uses a class 3 connection. We recommend that you reserve one class 3 connection on the Compact 5000 I/O EtherNet/IP adapter to establish a connection to the module. Otherwise, you can encounter a "Connection Request Error: Module connection limit exceeded" error.

Topic	Page
Before You Begin	43
Create a New Module	44
Edit the Module Configuration	54
View the Module Tags	69

Before You Begin

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

1. Create a Logix Designer application project.
2. If you use the Compact 5000 I/O serial module as remote I/O module, add a Compact 5000 I/O EtherNet/IP adapter to the project.

   For more information on how to add a Compact 5000 I/O EtherNet/IP adapter to a Logix Designer application project, see the EtherNet/IP Communication Modules in 5000 Series Systems User Manual, publication ENET-UM004.

IMPORTANT Use Studio 5000 Logix Designer® Version 31 or greater. You must install an Add-On Profile to use the serial module. To find the Add-On Profile go to the Product Compatibility and Download Center (PCDC).

Create a New Module

After you create a Logix Designer application project and, if necessary, add a Compact 5000 I/O EtherNet/IP adapter to the project, complete the following steps to create a module in the project.

There are two methods to add modules to your Logix Designer application project.

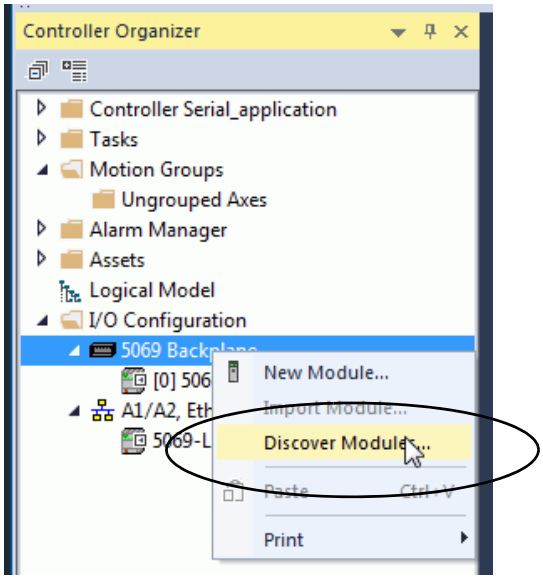
- Discover Local I/O Modules
- Discover Remote I/O Modules

Discover Local I/O Modules

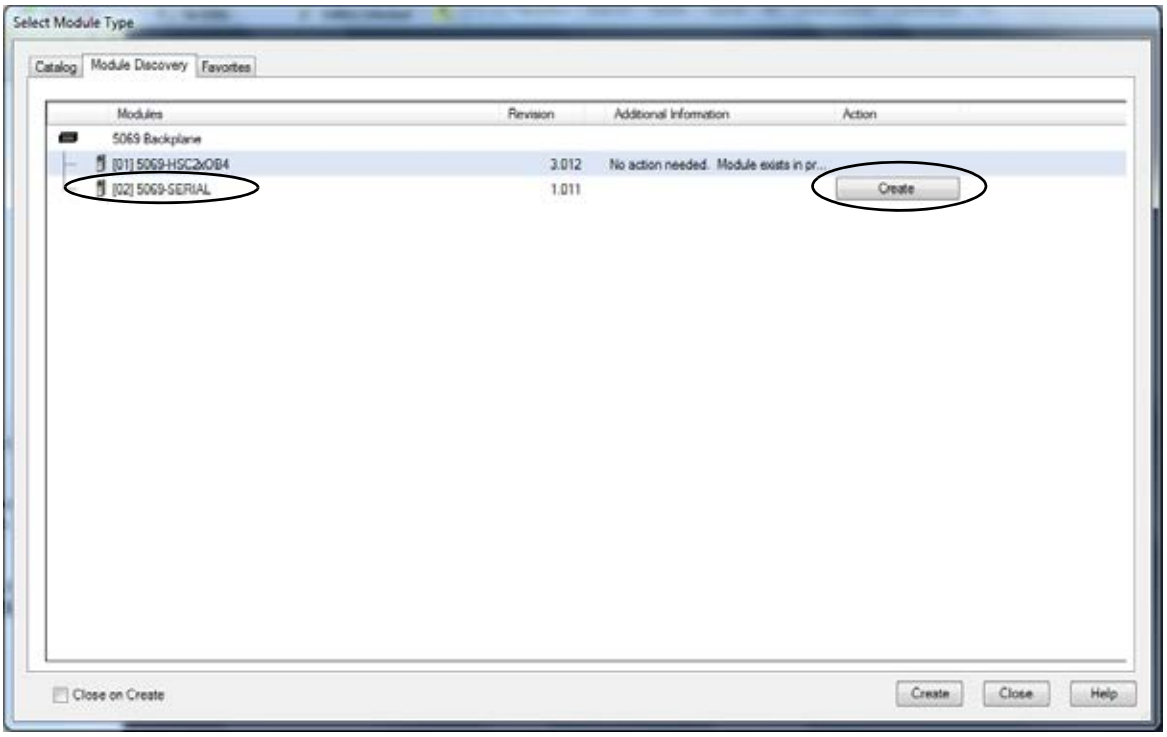
To use the Discover Modules method with local I/O modules, complete these steps.

1. Go online with your Logix Designer application.
2. Right-click the 5069 Backplane and choose Discover Modules.

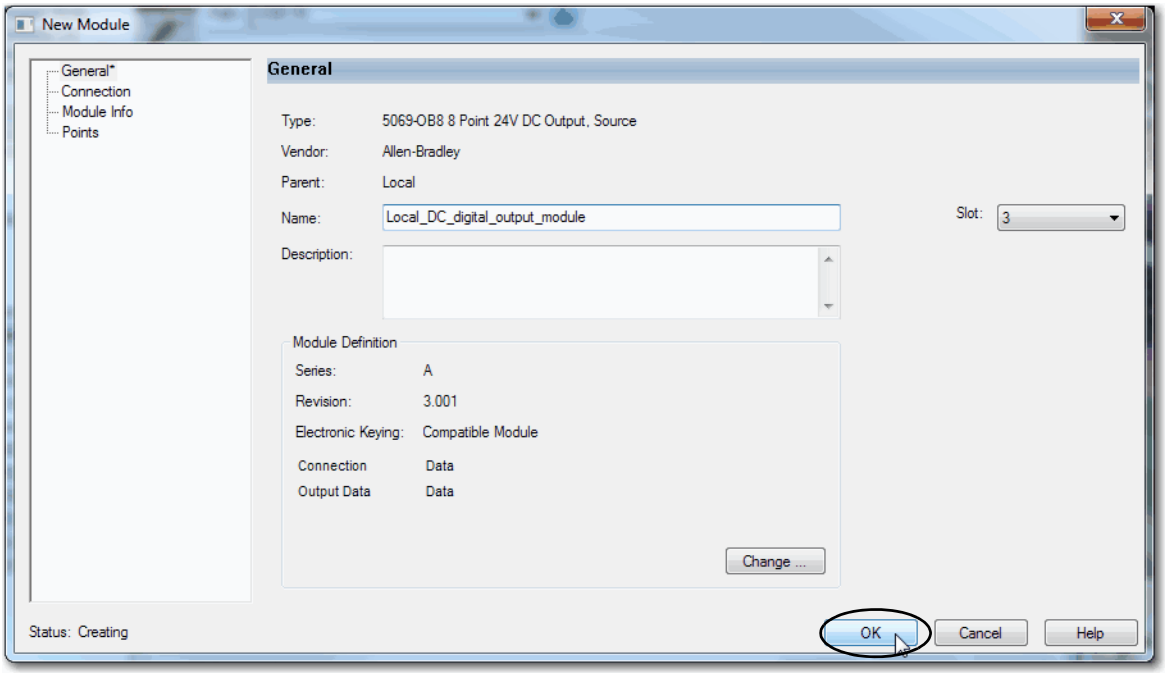
   The Logix Designer application automatically detects available modules that are connected to the backplane.



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

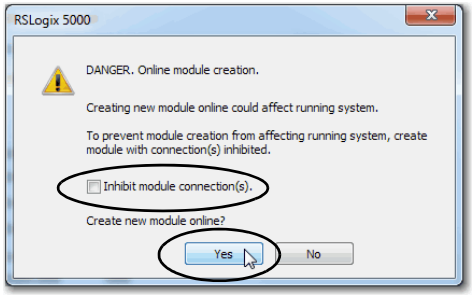


4. At the New Module window, configure the module properties and click OK.



5. At the warning dialog box, click Yes.

   TIP If you inhibit the module connection, you must remember to uninhibit the connection later.



6. Close the Select Module Type dialog box.

To add additional local 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.

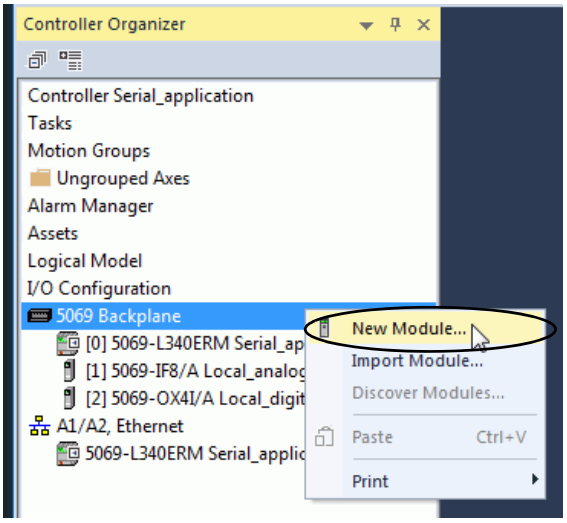
New Local I/O Modules

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

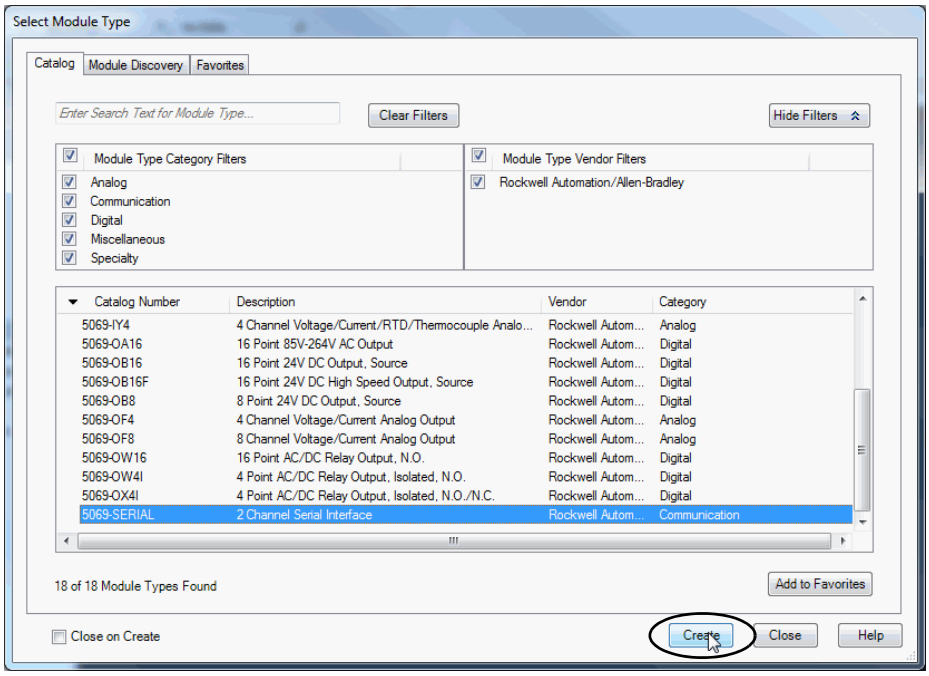
TIP This example shows how to add a local I/O module when the Logix Designer application project is offline.

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 Local I/O Modules on page 44. One exception is that, in step 1, you choose New Module instead of Discover Modules.

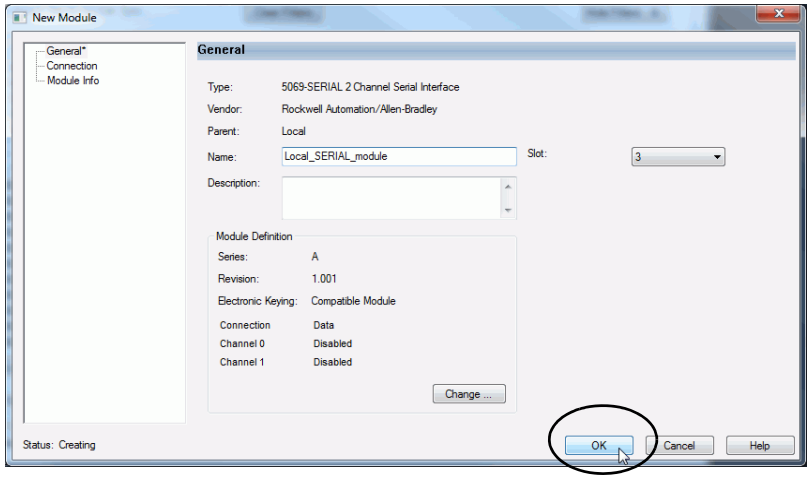
1. Right-click the 5069 Backplane and choose New Module.



2. Select the module and click create.



3. At the New Module window, configure the module properties and click OK.



To add additional local 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.

Discover Remote I/O Modules

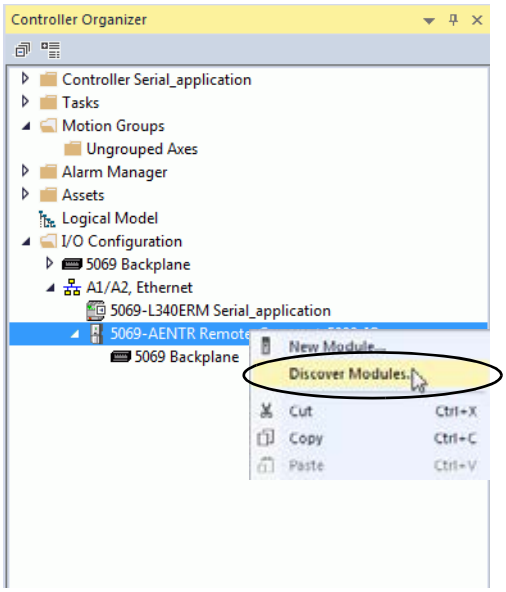
To use the Discover Modules method with remote I/O modules, complete these steps.

1. Go online with your Logix Designer application.

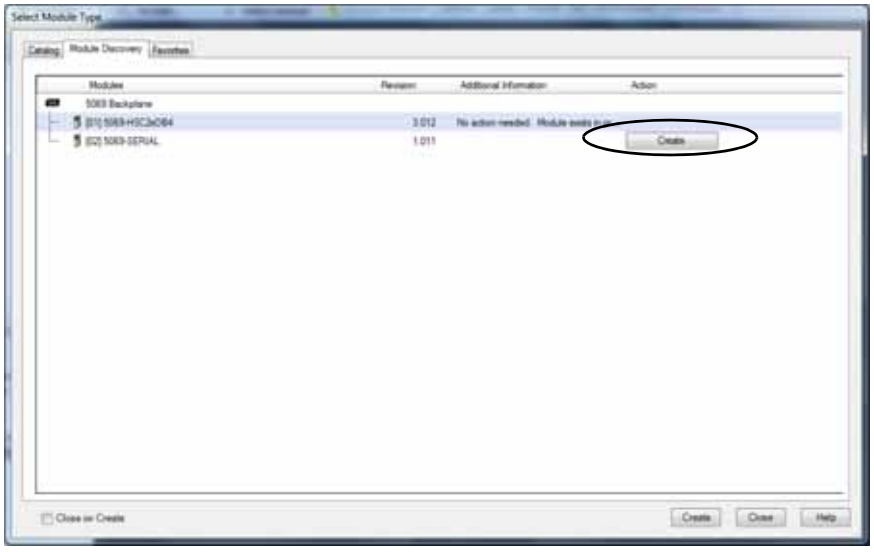
   The project must include at Compact 5000 I/O EtherNet/IP adapter.

2. Right-click the Compact 5000 I/O EtherNet/IP adapter and choose Discover Modules.

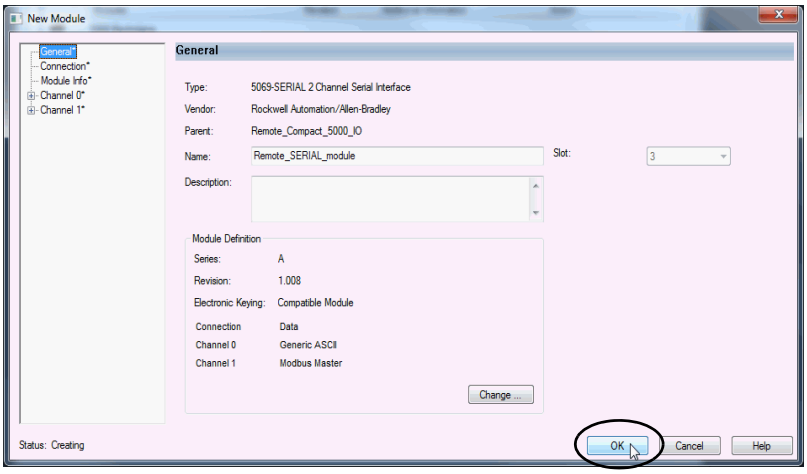
   The Logix Designer application automatically detects available modules that are connected to the backplane.



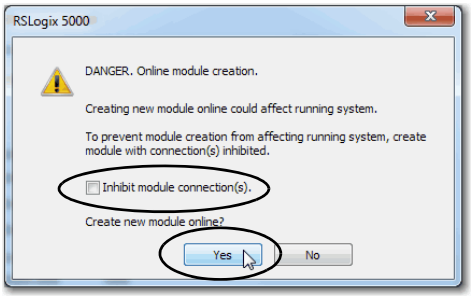
3. At the Select Module Type window, click Create to add the discovered module to your project.



4. At the New Module window, configure the module properties and click OK.



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

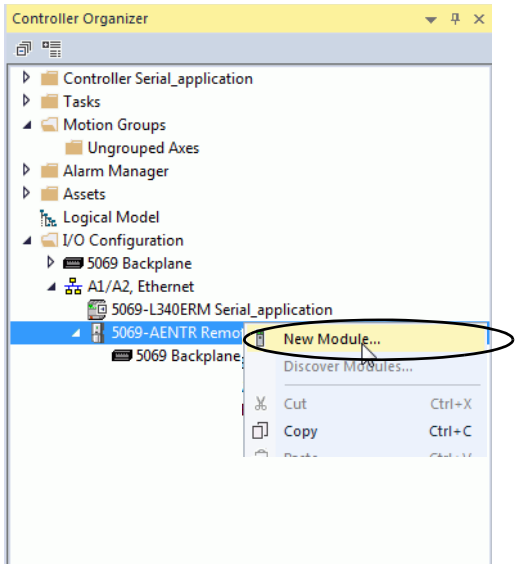
New Remote I/O Module

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

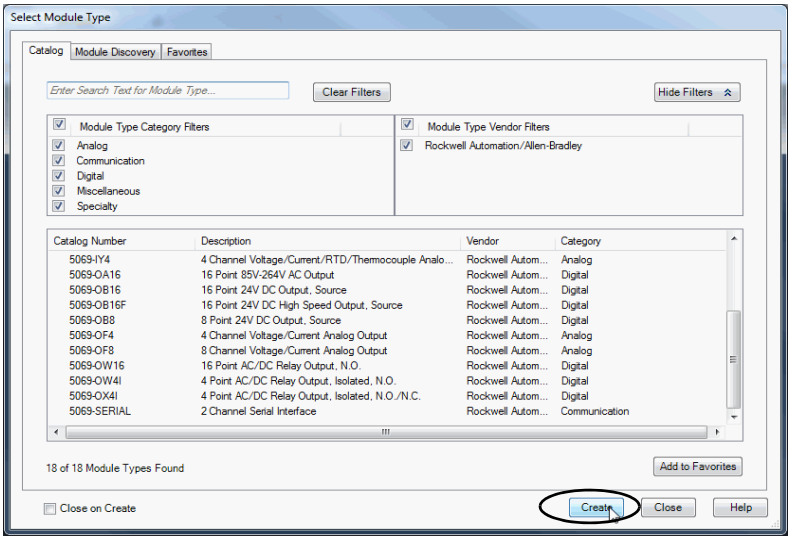
TIP This example shows how to add a local I/O module when the Logix Designer application project is offline.

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 Local I/O Modules on page 44. One exception is that, in step 1, you choose New Module instead of Discover Modules.

1. Right-click the 5069 Compact I/O EtherNet/IP adapter and choose New Module.

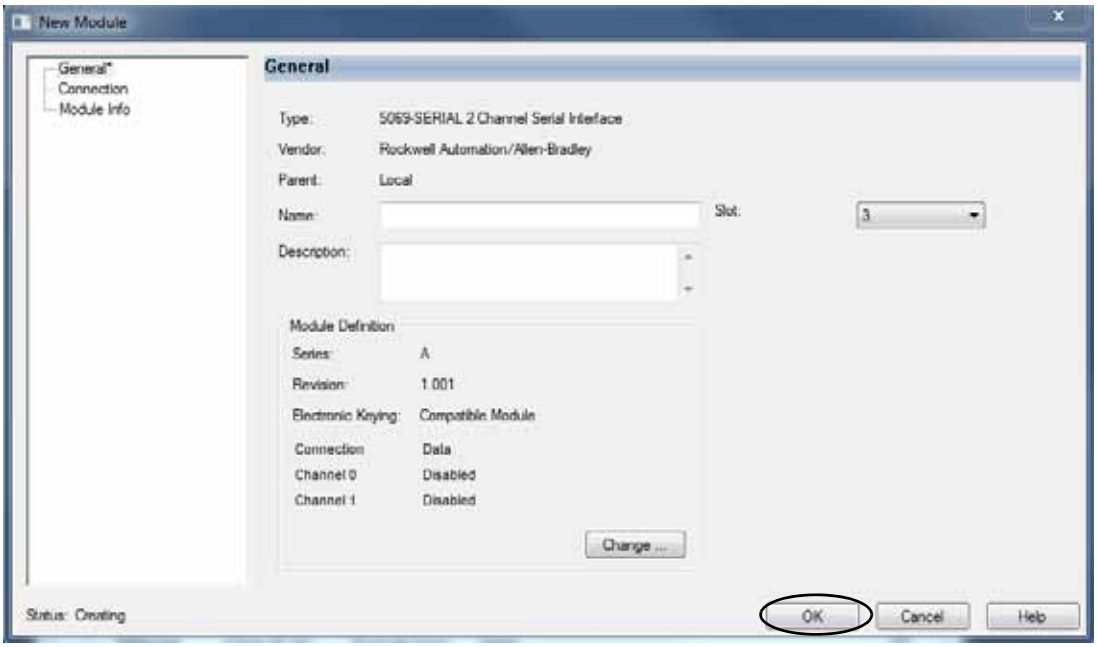


2. Select the module and click Create.



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

You click the category names in the New Module dialog box to view and change the configuration parameters.

IMPORTANT This chapter shows how to edit configuration when you add the module to the Logix Designer application project.

If you access the module configuration after it is added to the project, the dialog box is named Module Properties. The same categories are displayed as the categories displayed on the New Module dialog box.

Some new module configuration categories apply to all Compact 5000 I/O digital modules. Some categories are specific to the module type.

The following categories apply to all Compact 5000 I/O modules and are described in this section.

- General Category
- Connection Category
- Module Info Category
- View the Module Tags

General Category

The General category appears first when you create a module. The parameters in this category are the same for all Compact 5000 I/O modules.

You use this category to complete the following optional tasks:

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

Module Definition

Module Definition parameters are available on the General tab of the Module Properties dialog box in the Logix Designer application project. The module definition can only be edited during offline mode.

IMPORTANT To access the module definition parameters, click change on the general screen.

Table 11 describes the parameters on the Module Definition dialog box.

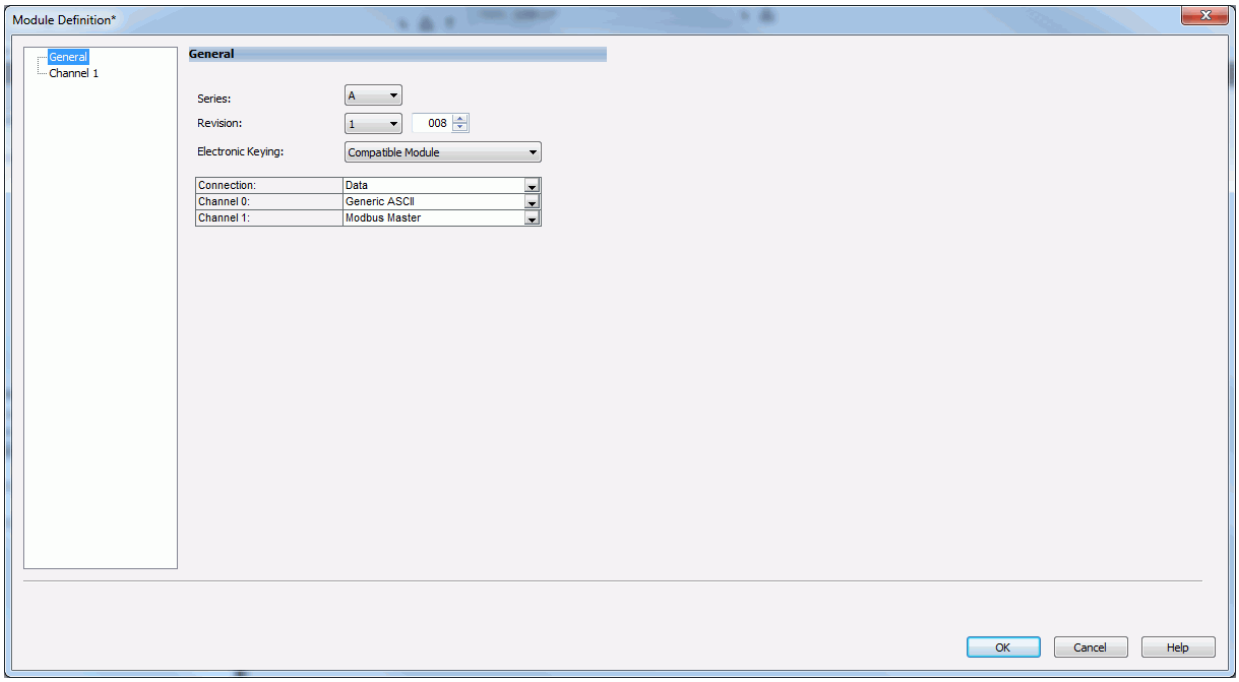


Table 11 describes the parameters that are available on the Module Definition dialog box.

Table 11 - 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 69 • 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 following for the module type you configure: • Determines the protocol that is used on Channel 0 or 1	Disabled Generic ASCII Modbus Master Modbus Slave For more information, see Table 12.

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

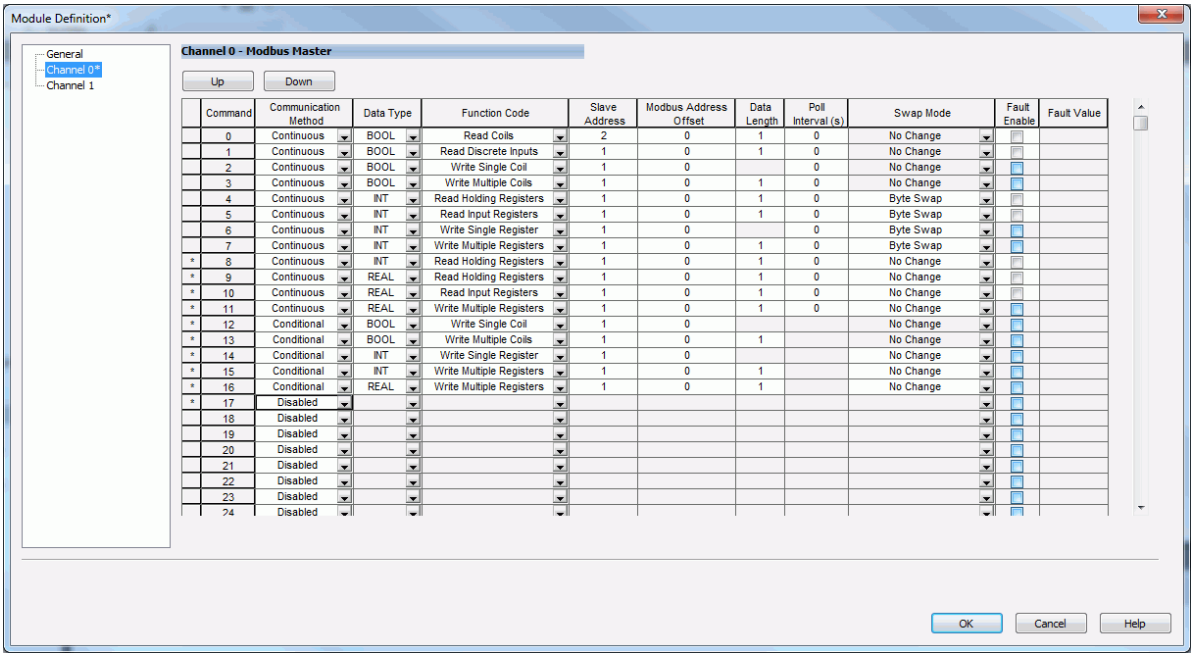
Table 12 - Communication Mode Definitions

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 devic 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 9 for an example of your options.
- If you choose Modbus Slave, see Figure 10 for an example of your options.

Figure 9 - 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 the user program is adjusted to reflect the new location of the command or the program will show and error.

Table 13 shows definitions of the Modbus Master configurable parameters.

Table 13 - Master Command List Modbus Master 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.
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, e.g., 11112222 = 22221111 • Byte Swap – each byte is swapped. e.g., 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:

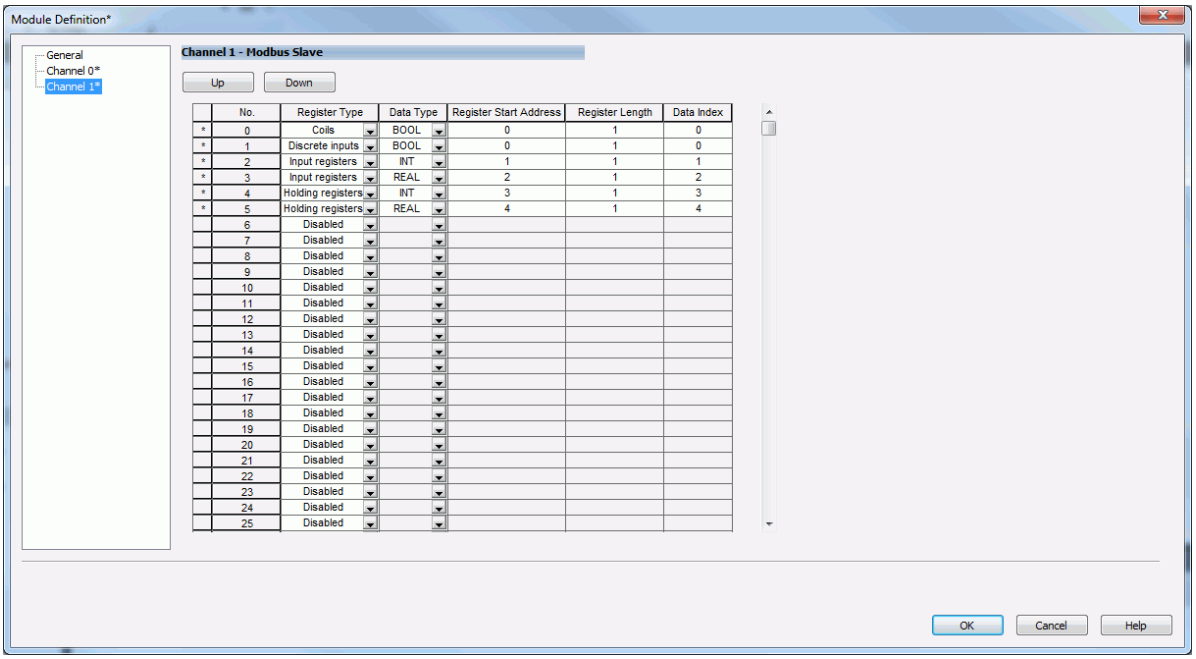
Command	Communication Method	Data Type	Function Code	Slave Address	Modbus Address Offset	Data Length	Poll Interval (s)	Swap Mode	Fault Enable	Fault Value
0	Continuous	INT	Read Holding Registers	1	0	125	0	No Change	☐	
1	Continuous	INT	Read Holding Registers	1	125	98	0	No Change	☐	
2	Conditional	INT	Write Multiple Registers	1	0	123		No Change	☐	
3	Conditional	INT	Write Multiple Registers	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:

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	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 Coils	1	0	1968		No Change	☐	
3	Conditional	BOOL	Write Multiple Coils	1	1968	1888		No Change	☐	

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

Figure 10 - 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 the user program is adjusted to reflect the new location of the command or the program will show and error.

Table 14 shows definitions of the Modbus Slave configurable parameters.

Table 14 - Modbus Slave Module Definition 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.

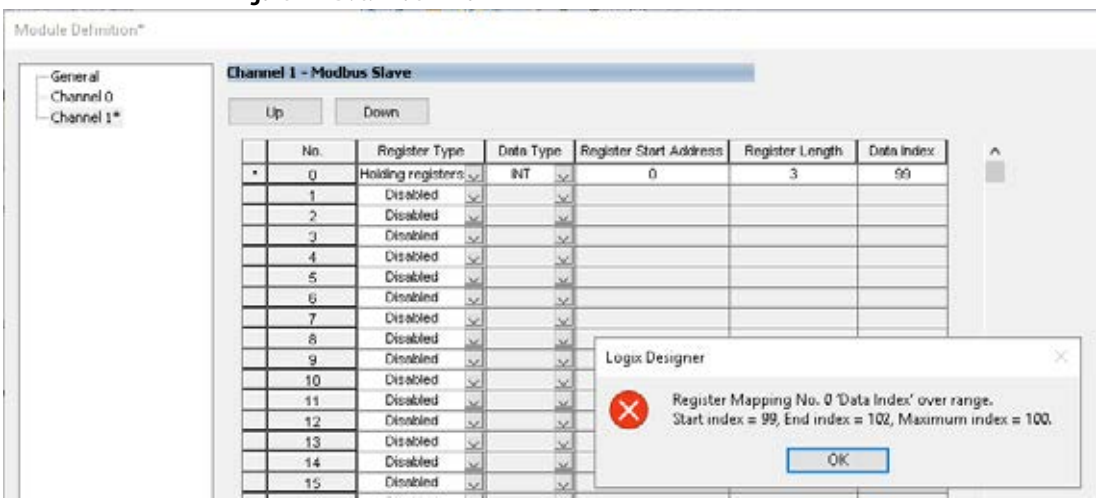
Data Index

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 it 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 11.

Figure 11 - 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

Channel 1 - Modbus Slave

Up Down

	No.	Register Type	Data Type	Register Start Address	Register Length	Data Index
	0	Holding registers	INT	3	3	0
	1	Holding registers	INT	32000	7	3
	2	Holding registers	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

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

Local:1:O1.Slave	{...}	AB:5000_ModbusSlave_...
Local:1:O1.Slave.Run	1	BOOL
Local:1:O1.Slave.SequenceNumber	0	INT
Local:1:O1.Slave.HoldingRegister	{...}	INT[100]
Local:1:O1.Slave.Coils	{...}	SINT[16]
Local:1:O1.Slave.InputRegister	{...}	INT[100]
Local:1:O1.Slave.DiscreteInput	{...}	SINT[16]

Connection Category

The Connection category lets you complete the following tasks:

- Set the RPI rate. For more information on the RPI, see on page 24.
- Inhibit the module. For more information on inhibit the module, see page 26.
- The connection over EtherNet/IP will be Unicast only.
- Configure whether a connection failure while the controller is in Run module causes a major or minor fault.

TIP The Module Fault area of the Connection category is useful during module troubleshooting. For more information on the Module Fault area, see page 71.

Generic ASCII

Figure 12 - Generic ASCII Connection

Figure 13 - Generic ASCII Module Info

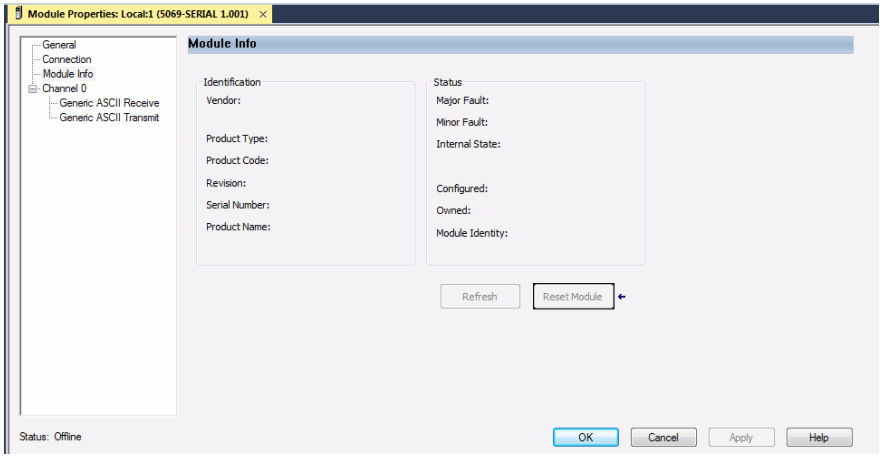
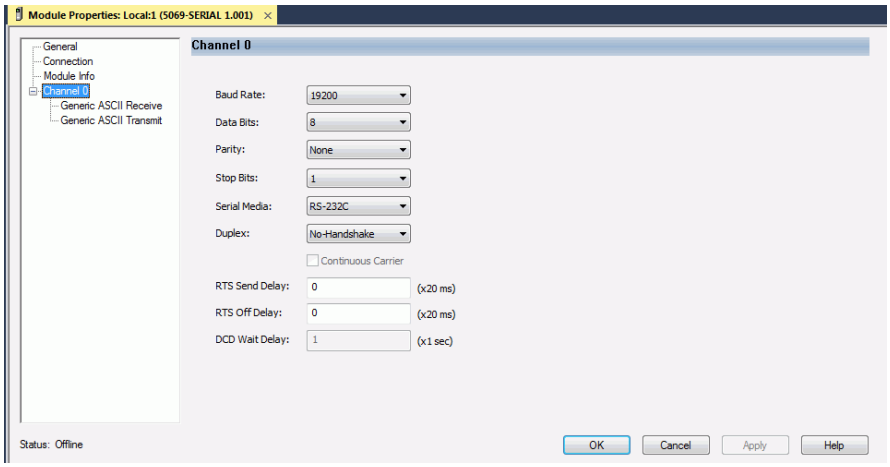


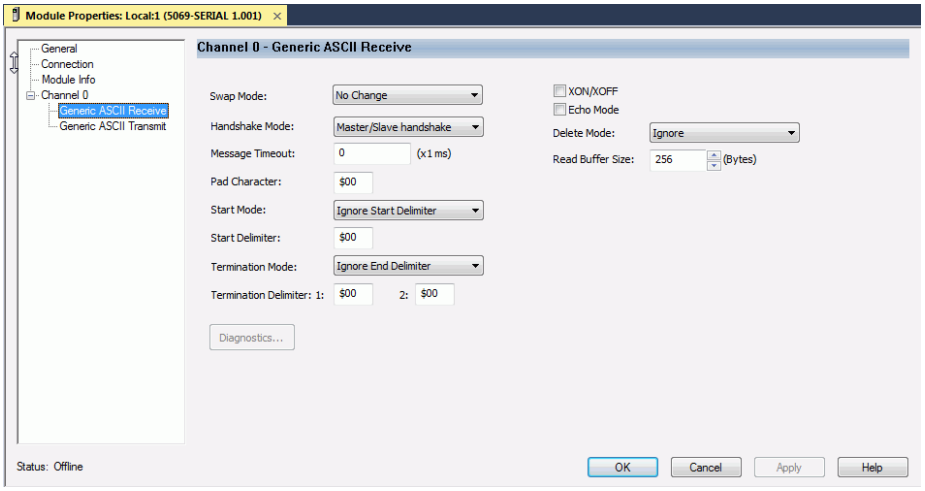
Figure 14...Figure 16 show the communication port that defines the baud rate, serial media setting, and receive and transmit settings.

Figure 14 - Generic ASCII Channel Parameters



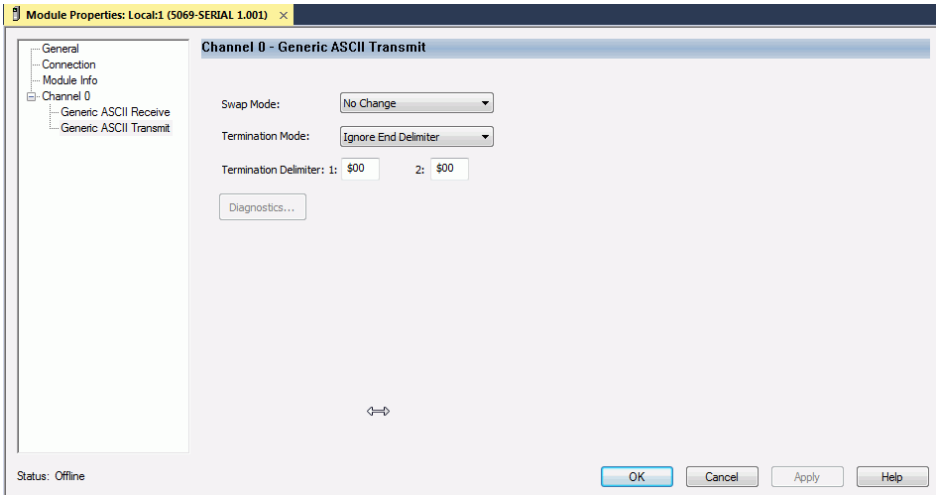
For module function definitions, see Common Module Functions on page 29

Figure 15 - Generic ASCII Receive



For more information, see Generic ASCII Receive Functions on page 33

Figure 16 - Generic ASCII Transmit



For more information, see Generic ASCII Transmit Functions on page 32

Modbus Master and Slave

Figure 17 - Modbus Master and Slave Connection

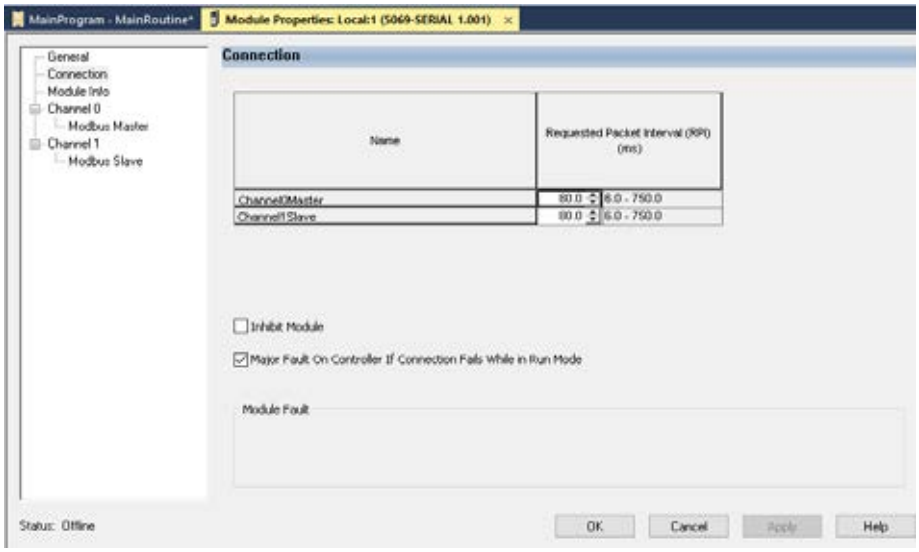
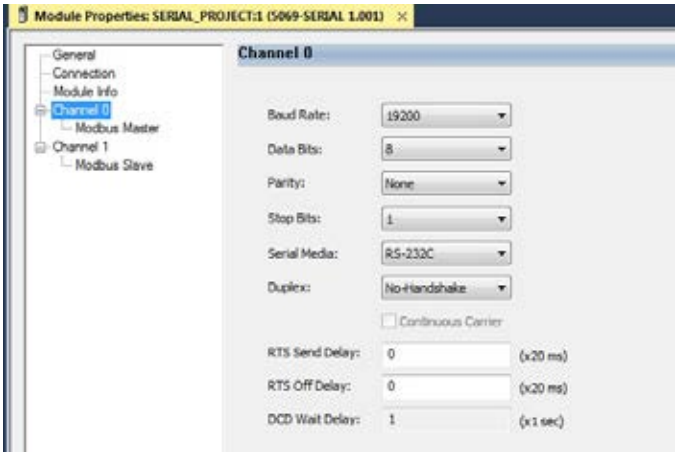


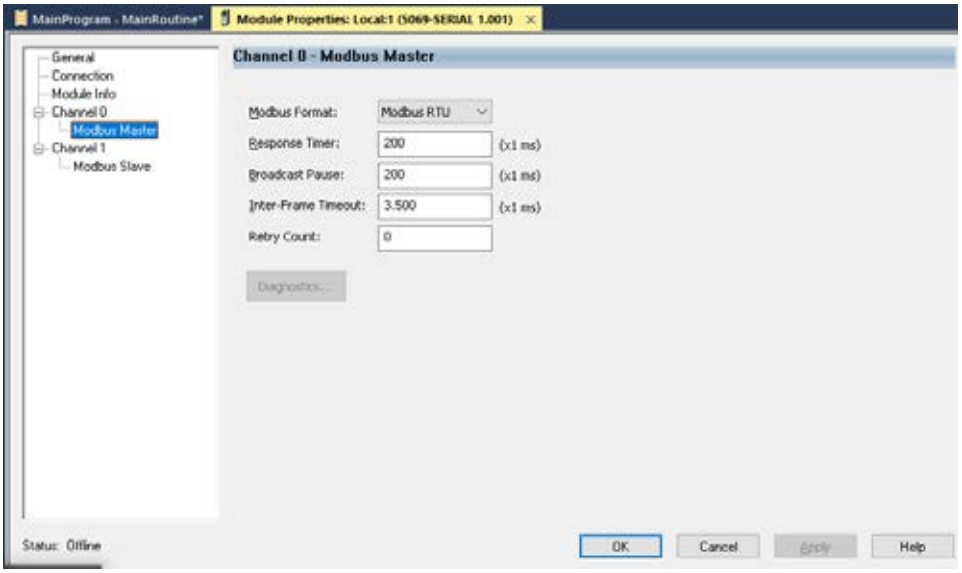
Figure 18 - Channel 0 and Channel 1 Parameters



For module function definitions, see Common Module Functions on page 29.

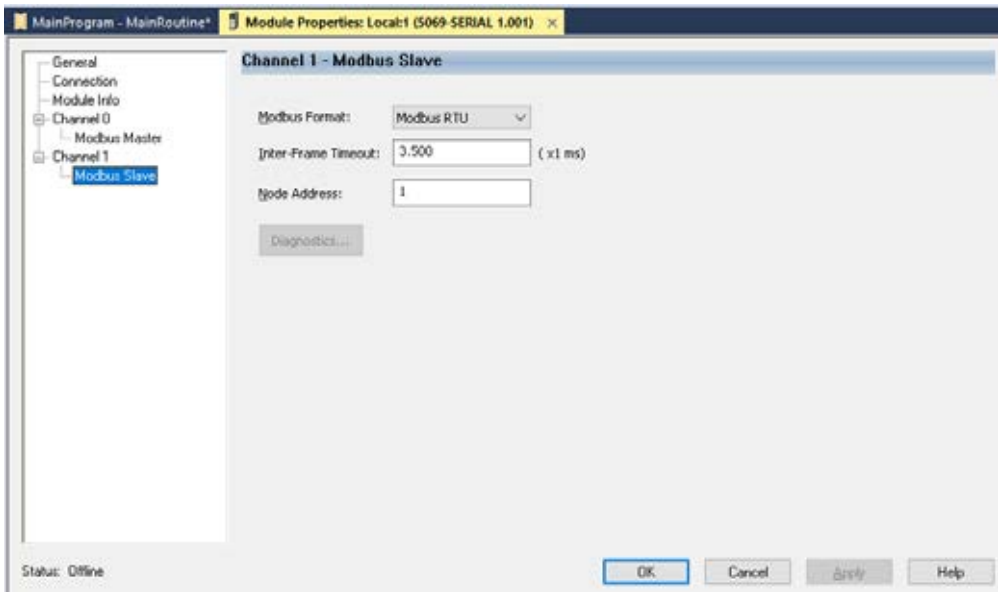
Figure 19 and Figure 20 show the detailed communication settings for Modbus Master and Modbus Slave.

Figure 19 - Modbus Master Channel Parameters



For Modbus Master function definitions, see Modbus Master Functions on page 41.

Figure 20 - Modbus Slave Channel Parameters

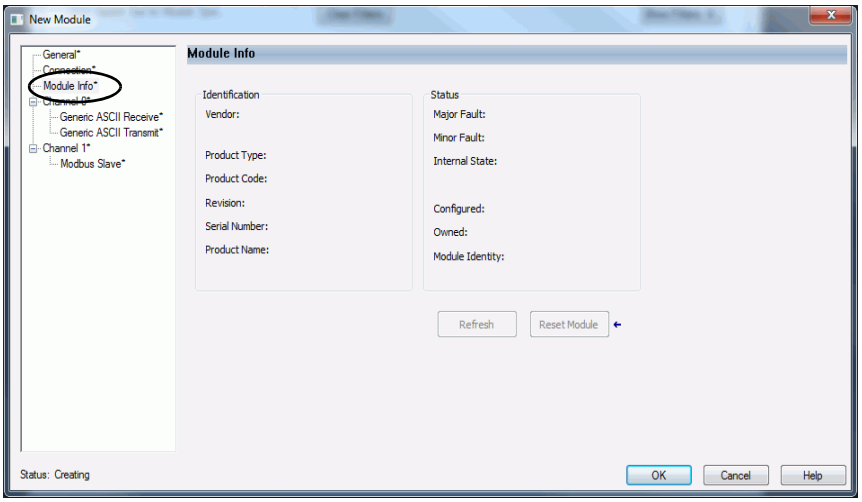


For Modbus Slave function definitions, see Modbus Slave Functions on page 42.

Module Info Category

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

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

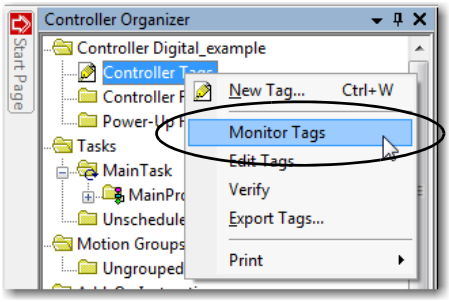


View the Module Tags

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.

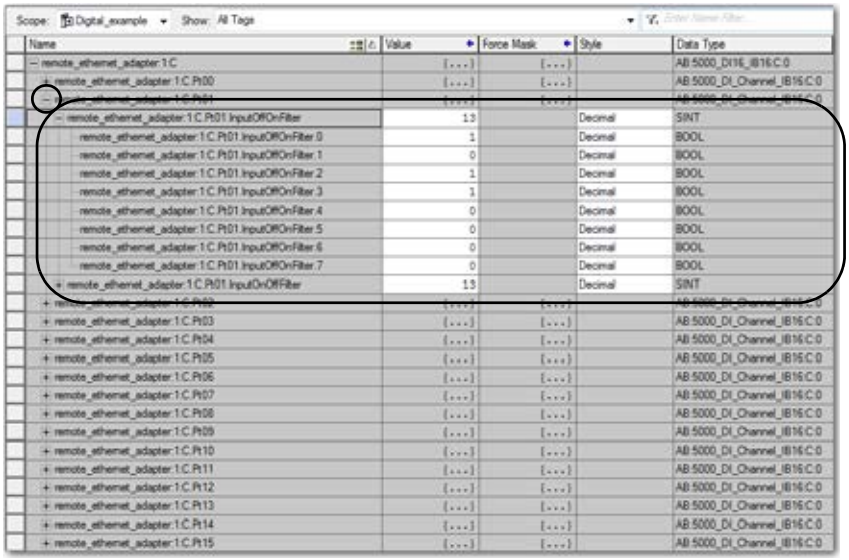
Complete the following steps to access the tags for a module.

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 Module Tags on page 75.

Notes:

Troubleshoot Your Module

Topic	Page
Module Status Indicator	71
Compact 5000 I/O Serial Module Status Indicators	73

Compact 5000™ I/O Serial modules use the following status indicators:

Module (MOD) Status Indicator - This indicator operates the same for all Compact 5000 I/O Serial modules.

Module Status Indicator

Table 15 describes the Module (MOD) Status indicator on Compact 5000 I/O Serial modules

Table 15 - Module Status Indicator - Compact 5000 I/O Module

Indicator State	Description	Recommended Action
Off	The module is not powered.	None if your application does not use the module If your application uses the module and it is expected to be operating, complete the following: • Confirm that the system is powered. • Confirm that the module is installed properly.
Steady green	The module has a connection to the owner-controller and is operating normally.	None

Table 15 - Module Status Indicator - Compact 5000 I/O Module

Indicator State	Description	Recommended Action
Flashing green	The following conditions exist: • Both channels are disabled. • The module is powering up. • The module has powered up successfully. • One of the following: – The module does not have a connection to the controller. A connection can result from missing, incomplete, or incorrect module configuration.	Complete the following: • 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.

Compact 5000 I/O Serial Module Status Indicators

Figure 21 shows the Compact 5000 I/O serial Module Status indicators.

Figure 21 - Compact 5000 I/O Serial Status Indicators

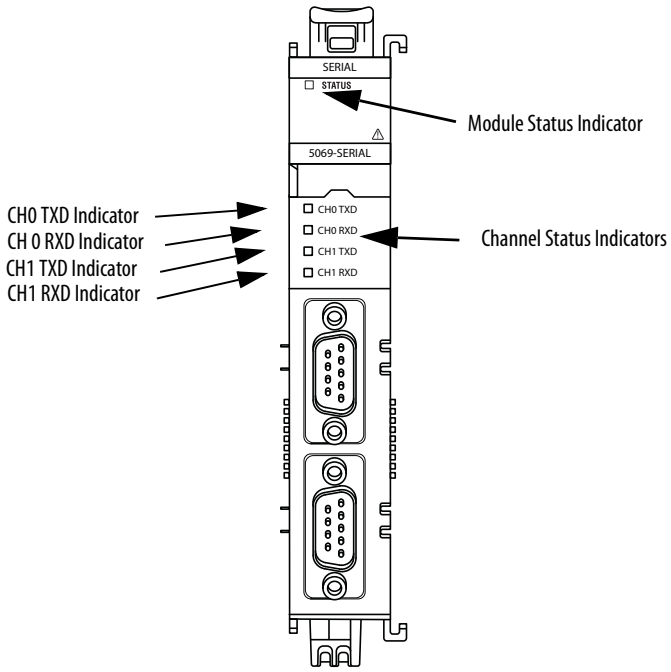


Table 16 - Channel Status Indicator Descriptions

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 serial port configuration and setup.

Notes:

Module Tags

Topic	Page
Name Conventions	75
Access the Tags	76
Channel Configured Generic ASCII Tags	77
Channel Configured for Generic ASCII	78
Channel Configured for Modbus Master	82
Channel Configured for Modbus Slave	86

Name Conventions

The module tags use defined naming conventions. The conventions are as follows:

- 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:

- SERIAL_PROJECT= name of the module
- 1 = slot number
- I0 = tag type (input) and channel number (0)
  - The possible Compact 5000 I/O serial tag types are I (input) and O (output)
  - The possible channels are channel 0 and channel 1
- RunMode = Parameter

Modbus Master Name Conventions

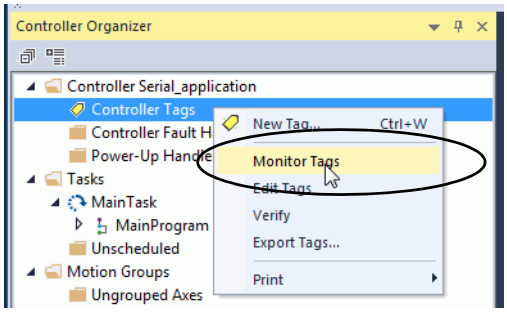
The following is an example for a Modbus Master tag name. The conventions for the example, “SERIAL_PROJECT:1:I00.RunMode”, would be the following:

- SERIAL_PROJECT= name of the module
- 1 = slot number
- I00 = tag type (input), channel number (0), number of connections (0)
  - The possible Compact 5000 I/O 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

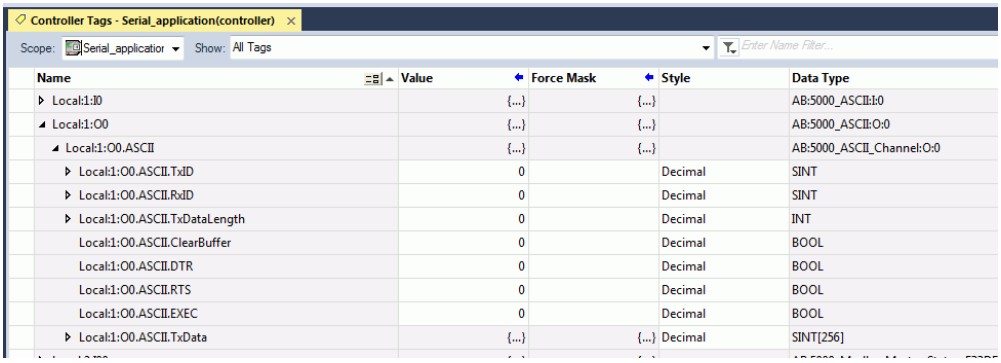
Access the Tags

You can view tags from the Tag Editor.

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

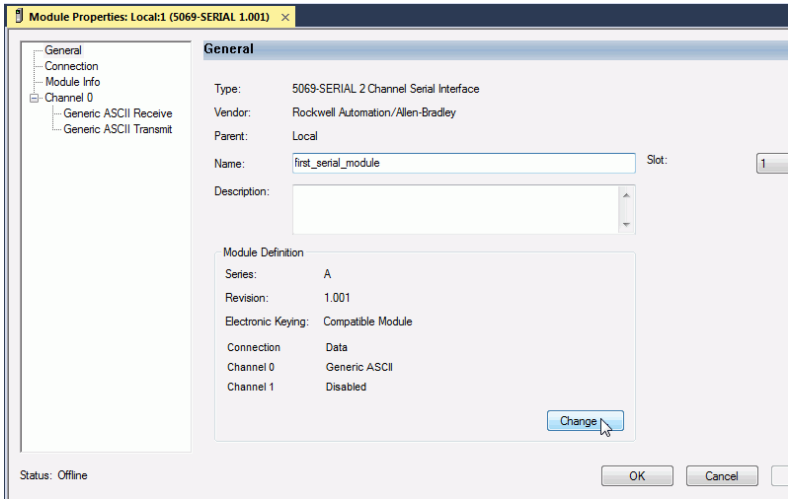


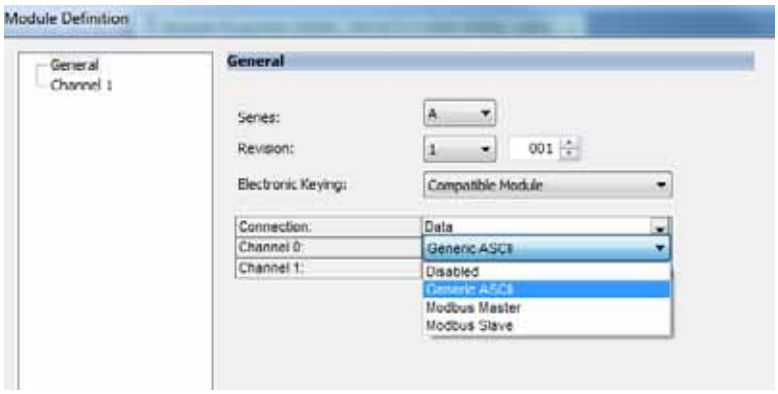
3. Open the tags as necessary to view specific tags.



Channel Configured Generic ASCII Tags

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



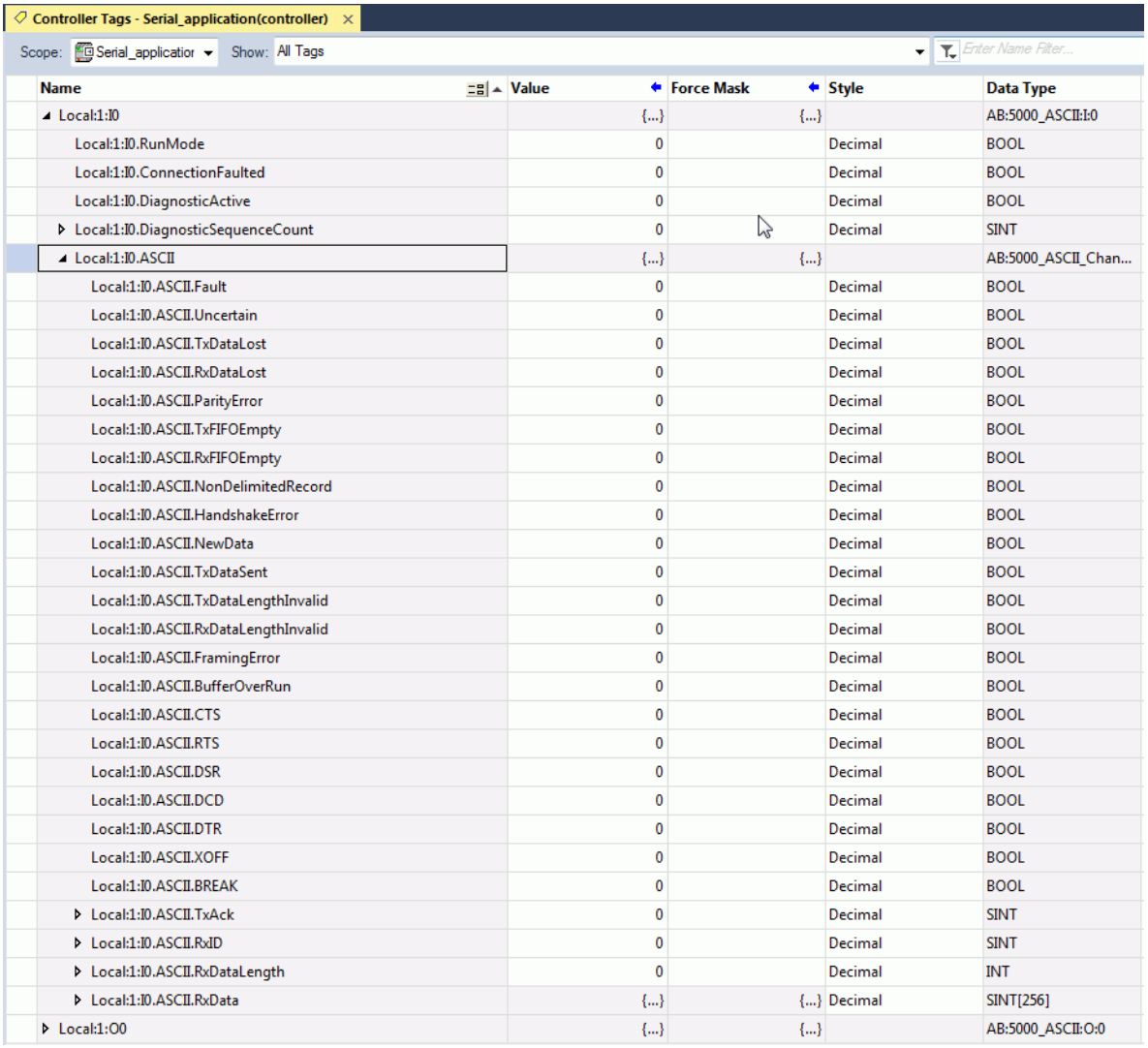


Channel Configured for Generic ASCII

Input Tags

The following image shows the tags that are described in the following table.

Table 17 describes the input tags of channel 1 configured to the Generic ASCII.



Name	Value	Force Mask	Style	Data Type
Local:1:I0	{...}	{...}		AB:5000_ASCII:I:0
Local:1:I0.RunMode	0		Decimal	BOOL
Local:1:I0.ConnectionFaulted	0		Decimal	BOOL
Local:1:I0.DiagnosticActive	0		Decimal	BOOL
Local:1:I0.DiagnosticSequenceCount	0		Decimal	SINT
Local:1:I0.ASCII	{...}	{...}		AB:5000_ASCII_Chan...
Local:1:I0.ASCII.Fault	0		Decimal	BOOL
Local:1:I0.ASCII.Uncertain	0		Decimal	BOOL
Local:1:I0.ASCII.TxDataLost	0		Decimal	BOOL
Local:1:I0.ASCII.RxDataLost	0		Decimal	BOOL
Local:1:I0.ASCII.ParityError	0		Decimal	BOOL
Local:1:I0.ASCII.TxFIFOEmpty	0		Decimal	BOOL
Local:1:I0.ASCII.RxFIFOEmpty	0		Decimal	BOOL
Local:1:I0.ASCII.NonDelimitedRecord	0		Decimal	BOOL
Local:1:I0.ASCII.HandshakeError	0		Decimal	BOOL
Local:1:I0.ASCII.NewData	0		Decimal	BOOL
Local:1:I0.ASCII.TxDataSent	0		Decimal	BOOL
Local:1:I0.ASCII.TxDataLengthInvalid	0		Decimal	BOOL
Local:1:I0.ASCII.RxDataLengthInvalid	0		Decimal	BOOL
Local:1:I0.ASCII.FramingError	0		Decimal	BOOL
Local:1:I0.ASCII.BufferOverRun	0		Decimal	BOOL
Local:1:I0.ASCII.CTS	0		Decimal	BOOL
Local:1:I0.ASCII.RTS	0		Decimal	BOOL
Local:1:I0.ASCII.DSR	0		Decimal	BOOL
Local:1:I0.ASCII.DCD	0		Decimal	BOOL
Local:1:I0.ASCII.DTR	0		Decimal	BOOL
Local:1:I0.ASCII.XOFF	0		Decimal	BOOL
Local:1:I0.ASCII.BREAK	0		Decimal	BOOL
Local:1:I0.ASCII.TxAck	0		Decimal	SINT
Local:1:I0.ASCII.RxID	0		Decimal	SINT
Local:1:I0.ASCII.RxDataLength	0		Decimal	INT
Local:1:I0.ASCII.RxData	{...}	{...}	Decimal	SINT[256]
Local:1:O0	{...}	{...}		AB:5000_ASCII:O:0

Table 17 - Generic ASCII Input Module Tags

Name	Data Type	Definition	Valid Values
I*x*.RunMode	BOOL	Channel's operating state	• 0 = Idle • 1 = Run
I*x*.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
I*x*.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
I*x*.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.
I*x*.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.
I*x*.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.
I*x*.ASCII.TxDataLost	BOOL	The transmitted data was lost. Until Clear Buffer, this bit continues to set.	• 0 = Non-occurrence • 1 = Occurrence
I*x*.ASCII.RxDataLost	BOOL	The received data was lost. Until Clear Buffer, this bit continues to set.	• 0 = Non-occurrence • 1 = Occurrence
I*x*.ASCII.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
I*x*.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
I*x*.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
I*x*.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
I*x*.ASCII.HandshakeError	BOOL	Handshake error. Used for Handshake mode only.	• 0 = Non-occurrence • 1 = Occurrence
I*x*.ASCII.NewData	BOOL	New data. Used for Handshake mode only.	• 0 = None • 1 = New Data
I*x*.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
I*x*.ASCII.TxDataLengthInvalid	BOOL	Indicates whether the O*x*.ASCII.TxDataLength is valid.	• 0 = Correct • 1 = Length of TxDataLength is illegal
I*x*.ASCII.RxDataLengthInvalid	BOOL	Indicates whether the Ox.ASCII.RxDataLength is valid	• 0 = Correct • 1 = Length of RxDataLength is illegal[(2)]

Table 17 - Generic ASCII Input Module Tags

Name	Data Type	Definition	Valid Values
Ix.ASCII.FramingError	BOOL	Indicates whether an error in framing has occurred.	• 0 = Non-occurrence • 1 = Occurrence
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](1)	SINT	Received data from serial port.	• -128…127

(1) X represents any possible value 0…255

(2) Length greater than the Max buffer length defined in the module definition page.

Output Tags

This screen capture shows the tags that are described in the following table.

Table 18 describes the output tags of channel 1 configured to the Generic ASCII

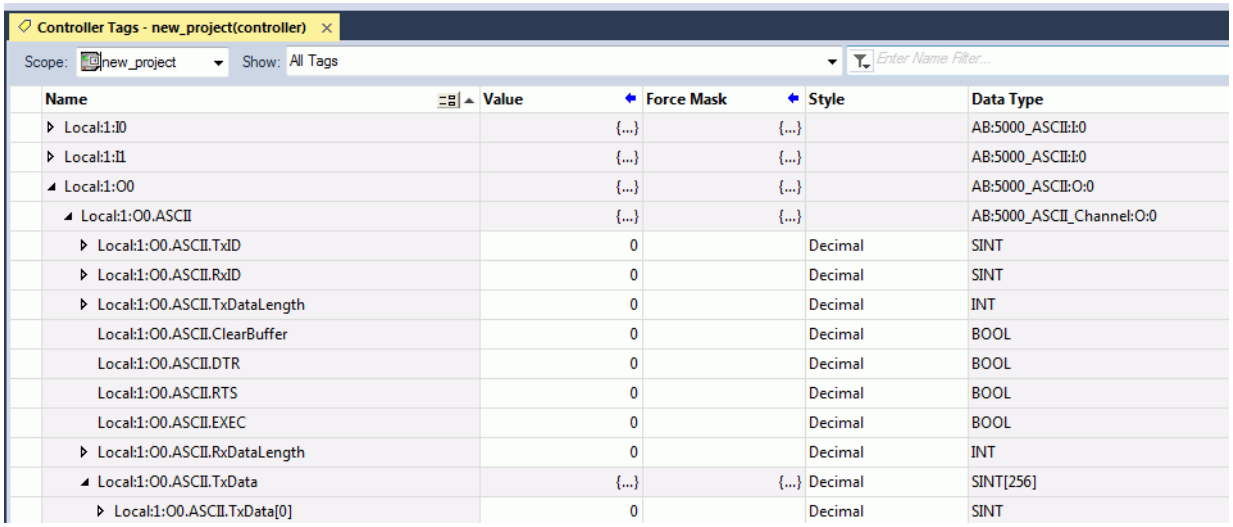


Table 18 - Generic ASCII Output Module Tags

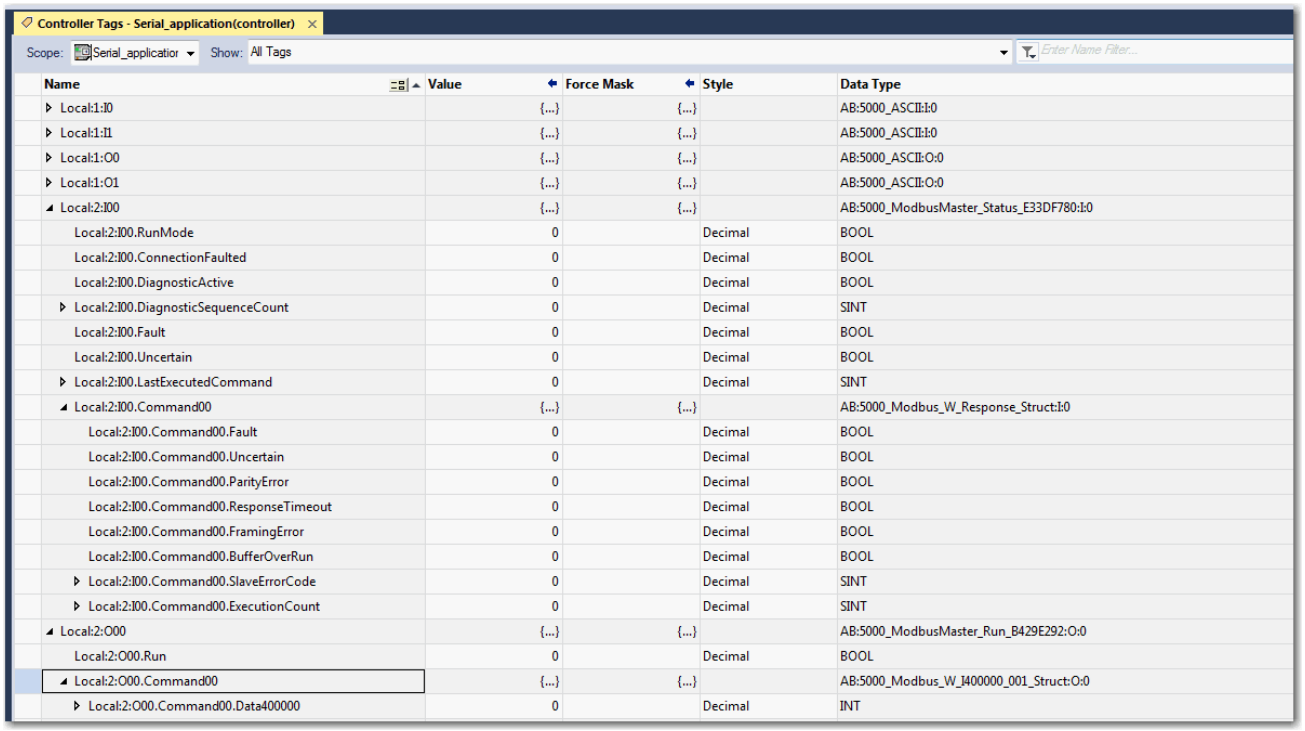
Name	Data Type	Definition	Valid Values
0x.ASCII.TxID	SINT	This ID is incremented when you want to transmit data from the serial port.	• -128…127[2]
0x.ASCII.RxID	SINT	This ID is incremented when you receive data from the serial port. It is only used in Master/Slave Handshake Mode.	• -128…127[2]
0x.ASCII.TxDataLength	INT	Length of Transmitted Data of each channel.	• 1…256
0x.ASCII.RxDataLength	INT	Length of Received Data of each channel.	• 1…256
0x.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
0x.ASCII.DTR	BOOL	Signal level of DTR line sent out when rising edge of EXEC bit is detected.	• 0 = Inactive • 1 = Active
0x.ASCII.RTS	BOOL	Signal level of the RTS line sent out when rising edge of EXEC bit is detected.	• 0 = Inactive • 1 = Active
0x.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
0x.ASCII.TxData[x][1]	SINT	Output data from module.	• -128…127

(1) 0...255

(2) The value of 0 must be skipped except during module power-up.

Channel Configured for Modbus Master

This screen capture shows the tags that are described in the following tables.



Name	Value	Force Mask	Style	Data Type
Local:1:I0	{...}	{...}		AB:5000_ASCII:I:0
Local:1:I1	{...}	{...}		AB:5000_ASCII:I:0
Local:1:O0	{...}	{...}		AB:5000_ASCII:O:0
Local:1:O1	{...}	{...}		AB:5000_ASCII:O:0
Local:2:I00	{...}	{...}		AB:5000_ModbusMaster_Status_E33DF780:I:0
Local:2:I00.RunMode	0		Decimal	BOOL
Local:2:I00.ConnectionFaulted	0		Decimal	BOOL
Local:2:I00.DiagnosticActive	0		Decimal	BOOL
Local:2:I00.DiagnosticSequenceCount	0		Decimal	SINT
Local:2:I00.Fault	0		Decimal	BOOL
Local:2:I00.Uncertain	0		Decimal	BOOL
Local:2:I00.LastExecutedCommand	0		Decimal	SINT
Local:2:I00.Command00	{...}	{...}		AB:5000_Modbus_W_Response_Struct:I:0
Local:2:I00.Command00.Fault	0		Decimal	BOOL
Local:2:I00.Command00.Uncertain	0		Decimal	BOOL
Local:2:I00.Command00.ParityError	0		Decimal	BOOL
Local:2:I00.Command00.ResponseTimeout	0		Decimal	BOOL
Local:2:I00.Command00.FramingError	0		Decimal	BOOL
Local:2:I00.Command00.BufferOverRun	0		Decimal	BOOL
Local:2:I00.Command00.SlaveErrorCode	0		Decimal	SINT
Local:2:I00.Command00.ExecutionCount	0		Decimal	SINT
Local:2:O00	{...}	{...}		AB:5000_ModbusMaster_Run_B429E292:O:0
Local:2:O00.Run	0		Decimal	BOOL
Local:2:O00.Command00	{...}	{...}		AB:5000_Modbus_W_I400000_001_Struct:O:0
Local:2:O00.Command00.Data400000	0		Decimal	INT

Input Tags

In the following table, 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.

TIP The yy in the tag names represents the Modbus Master command number and the zzzzzz represents the Modbus data address.

Table 19 - Modbus Master Input Tags

Name	Data Type	Definition	Valid Values
I*xx*.RunMode	BOOL	Channel's operating state	• 0 = Idle • 1 = Run
I*xx*.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
I*xx*.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

Table 19 - Modbus Master Input Tags

Name	Data Type	Definition	Valid Values
I*xx*.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.
I*xx*.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.
I*xx*.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.
I*xx*.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 shown prior to any command) • 0…49
I*xx*.Command.*yy*.Fault	BOOL	Indicates that channel data is inaccurate and cannot be trusted for use in the application. The number of Commandyy tags (where yy can be 0…49) are dynamically created depending on the command list you create.	• 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.
I*xx*.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.
I*xx*.Command.*yy*.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
I*xx*.Command.*yy*.ResponseTimeout	BOOL	Shows if there was a response timeout.	• 0 = Non-occurrence • 1 = Occurrence
I*xx*.Command.*yy*.FramingError	BOOL	Shows if there was a framing error.	• 0 = Non-occurrence • 1 = Occurrence
I*xx*.Command.*yy*.BufferOverRun	BOOL	Shows if a buffer overrun has occurred.	• 0 = Non-occurrence • 1 = Occurrence

Table 19 - Modbus Master Input Tags

Name	Data Type	Definition	Valid Values
I*xx*.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
I*xx*.Command.*yy*.ExecutionCount	SINT	The number of times the command is executed.	• -128...127
Ixx.Commandxx.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

In the following table, the *xx* in the tag name represents the channel number because the module has two channels, and both channels support the use of Modbus Master.‘

TIP The yy in the tag names represents the Modbus Master command number and the zzzzzz represents the Modbus data address.

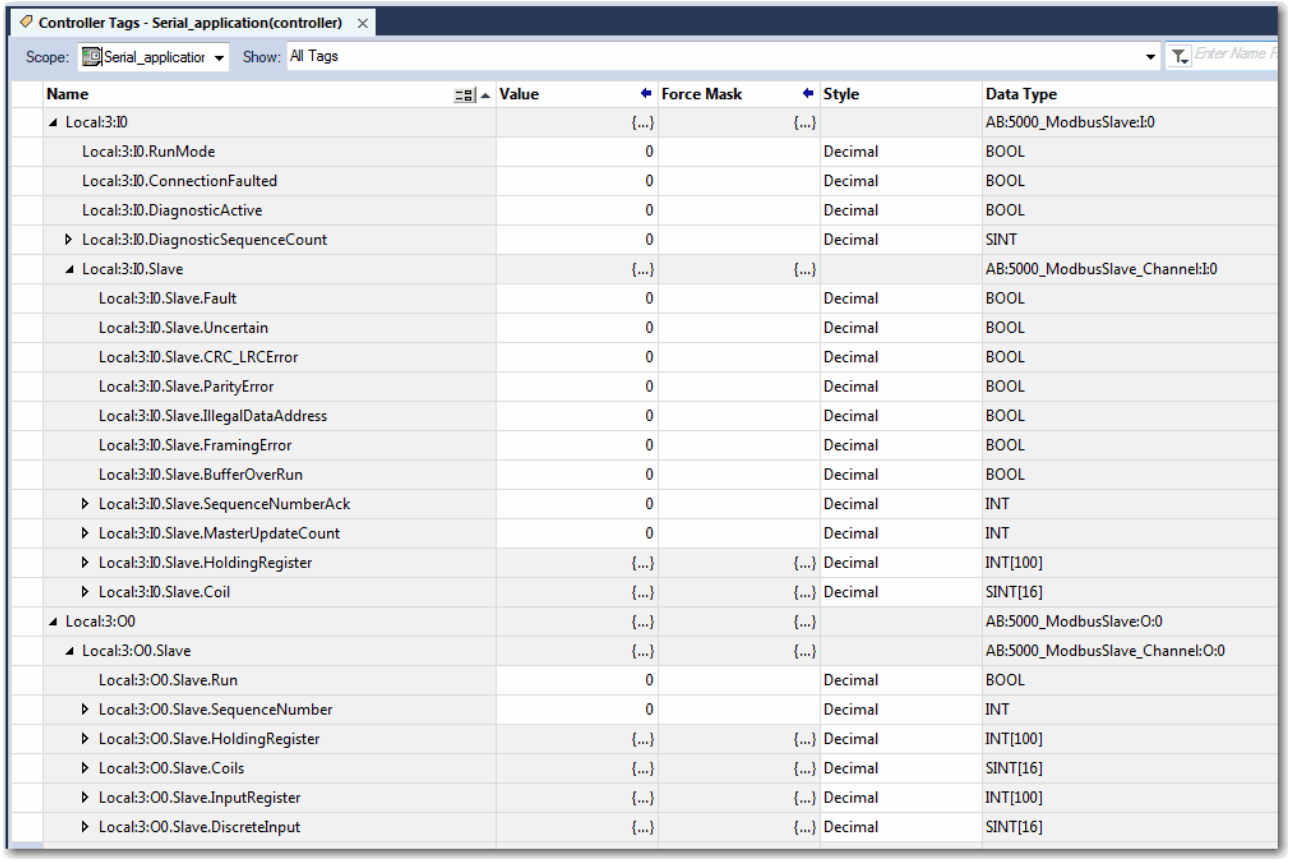
Table 20 - Modbus Master Output Tags

Name	Data Type	Definition	Valid Values
O*xx*.Run	BOOL	Channel’s operating state.	• 0 = Idle • 1 = Run(1)
O*xx*.Command*yy*.Datazzzzzz	SINT, INT, or REAL	Command write data. The number of Commandyy tags (where yy can be 0…49) are dynamically created depending on the command list you create.	• -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 screen capture shows the tags that are described in the following tables.



Name	Value	Force Mask	Style	Data Type
Local:3:I0	{...}	{...}		AB:5000_ModbusSlave:I:0
Local:3:I0.RunMode	0		Decimal	BOOL
Local:3:I0.ConnectionFaulted	0		Decimal	BOOL
Local:3:I0.DiagnosticActive	0		Decimal	BOOL
Local:3:I0.DiagnosticSequenceCount	0		Decimal	SINT
Local:3:I0.Slave	{...}	{...}		AB:5000_ModbusSlave_Channel:I:0
Local:3:I0.Slave.Fault	0		Decimal	BOOL
Local:3:I0.Slave.Uncertain	0		Decimal	BOOL
Local:3:I0.Slave.CRC_LRCError	0		Decimal	BOOL
Local:3:I0.Slave.ParityError	0		Decimal	BOOL
Local:3:I0.Slave.IllegalDataAddress	0		Decimal	BOOL
Local:3:I0.Slave.FramingError	0		Decimal	BOOL
Local:3:I0.Slave.BufferOverRun	0		Decimal	BOOL
Local:3:I0.Slave.SequenceNumberAck	0		Decimal	INT
Local:3:I0.Slave.MasterUpdateCount	0		Decimal	INT
Local:3:I0.Slave.HoldingRegister	{...}	{...}	Decimal	INT[100]
Local:3:I0.Slave.Coil	{...}	{...}	Decimal	SINT[16]
Local:3:O0	{...}	{...}		AB:5000_ModbusSlave:O:0
Local:3:O0.Slave	{...}	{...}		AB:5000_ModbusSlave_Channel:O:0
Local:3:O0.Slave.Run	0		Decimal	BOOL
Local:3:O0.Slave.SequenceNumber	0		Decimal	INT
Local:3:O0.Slave.HoldingRegister	{...}	{...}	Decimal	INT[100]
Local:3:O0.Slave.Coils	{...}	{...}	Decimal	SINT[16]
Local:3:O0.Slave.InputRegister	{...}	{...}	Decimal	INT[100]
Local:3:O0.Slave.DiscreteInput	{...}	{...}	Decimal	SINT[16]

Input Tags

In the following table, 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.

Table 21 - Modbus Slave Input Tags

Name	Data Type	Definition	Valid Values
Ix.RunMode	BOOL	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.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.
Ix.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.
Ix.Slave.CRC_LRCError	BOOL	Status shows that CRC (LRC) Error is occurred or not	• 0 = Non-occurrence • 1 = Occurrence
Ix.Slave.ParityError	BOOL	Status that shows whether a parity error has occurred or not.	• 0 = Non-occurrence • 1 = Occurrence
Ix.Slave.IllegalDataAddress	BOOL	Status shows that user requests out of Modbus Register Address	• 0 = Non-occurrence • 1 = Occurrence
Ix.Slave.BufferOverRun	BOOL	Status shows that whether Over Run is occurred or not in ASIC.	• 0 = Non-occurrence • 1 = Occurrence
Ix.Slave.FramingError	BOOL	Shows if there was a framing error.	• 0 = Non-occurrence • 1 = Occurrence
Ix.Slave.SequenceNumberAck	INT	Acknowledges the sequence number.	• -32768…32767
Ix.Slave.MasterUpdateCount	INT	Any Modbus change will update this counter.	• -32768…32767
Ix.Slave.HoldingRegister[x](1)	INT	Produce Data that are written by Modbus Master as Data in Produce Tag.	• -32768…32767
Ix.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 0…99

(2) X represents any possible value 0…15

Output Tags

In the following table, the *xx* in the tag name represents the channel number because the module has two channels, and both channels support the use of Modbus Slave.

Table 22 - Modbus Slave Output Tags

Name	Data Type	Definition	Valid Values
0*x*.Slave.Run	BOOL	Channel's operating state	• 0 = Idle • 1 = Run(5)
0*x*.Slave.SequenceNumber	INT	Sequence number for updating slave data from controller	• -32768…32767
0*x*.Slave.HoldingRegister[x](1)	INT	Location of holding register values defined by user for the serial module	• -32768…32767
0*x*.Slave.Coils[x](2)	SINT	Location of slave coil values defined by user for the serial module	• -128…127
0*x*.Slave.InputRegister[x](3)	INT	Location of input register values defined by user for the serial module	• -32768…32767
0*x*.Slave.DiscreteInput[x](4)	SINT	Location of discrete input values defined by user for the serial module	• -128…127

(1) X represents any possible value 0…99

(2) X represents any possible value 0…15

(3) X represents any possible value 0…99

(4) X represents any possible value 0…15

(5) The Run bit is to start the update of the output (0) 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 0 tag.

Notes:

Master Command List

Topic	Page
Read Coil Status (Function Code 01)	91
Read Input Status (Function Code 02)	93
Read Holding Registers (Function Code 03)	94
Read Input Registers (Function Code 04)	95
Force Single Coil (Function Code 05)	96
Preset Single Register (Function Code 06)	98
Force Multiple Coils (Function Code 15)	99
Preset Multiple Registers (Function Code 16)	100

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.

TIP 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 left most 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.

TIP 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 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 left-most 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.

TIP 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	Data Number of Registers High	Data Number of Registers Low	Error Check Field (2 bytes)
0B	02	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.

TIP 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.

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.

TIP Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.

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

TIP 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 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.

TIP
- 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	Present 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.

TIP Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.

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.

TIP 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	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	Error Check Field (2 bytes)
0B	0F	00	13	00	0A	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.

TIP Functions 5, 6, 15, and 16 are the only messages that are recognized as valid for broadcast.

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

TIP 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 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

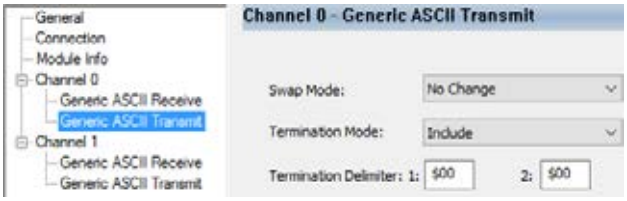
Programming Example

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

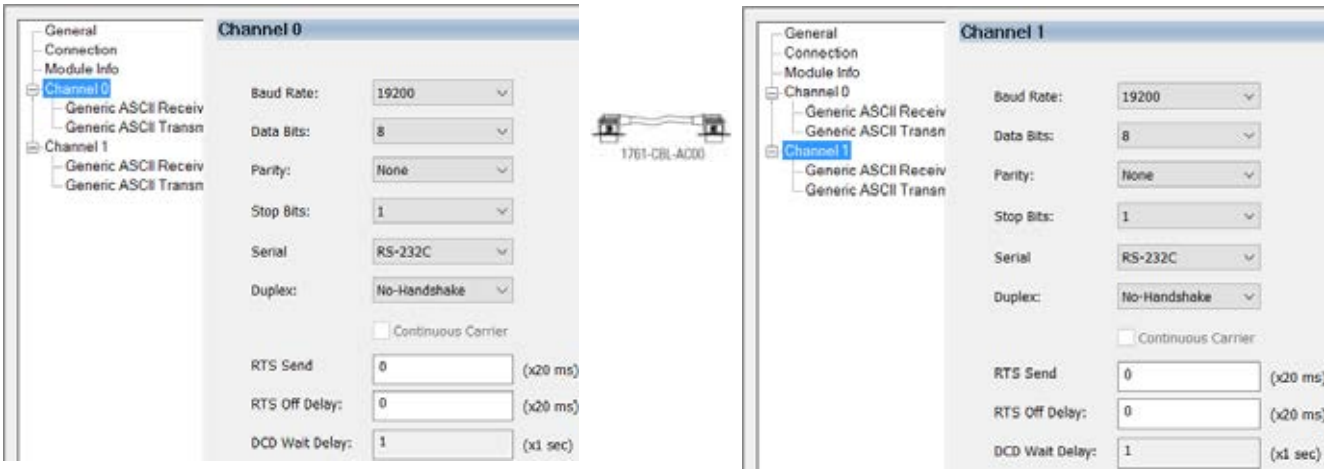
The following image shows the Generic ASCII Transmit Configuration.



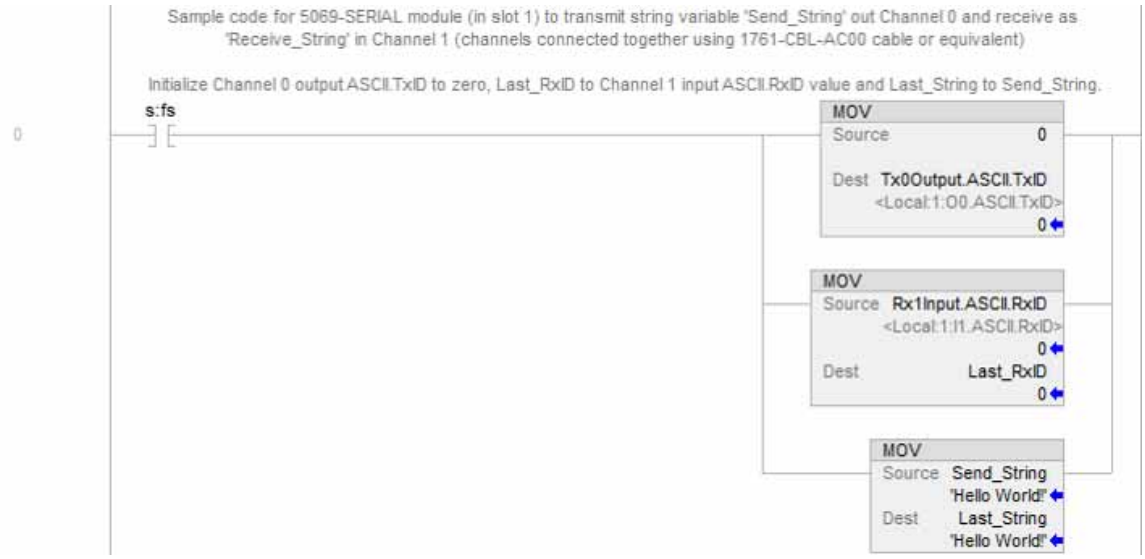
The following image shows the Generic ASCII Receive Configuration

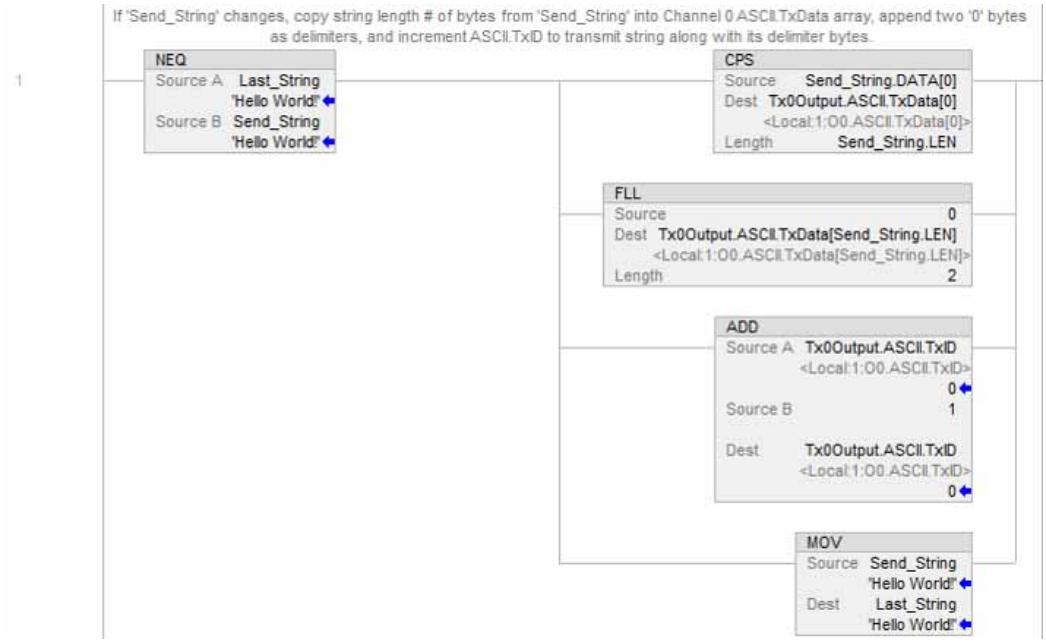
To test the ASCII capabilities by transmitting out Channel 0 and receiving in Channel 1, connect the two serial ports together using a 1761-CBL-AC00 cable. On the Module Properties Channel screens, verify that both channels are configured identically for RS-232C. See the following images.

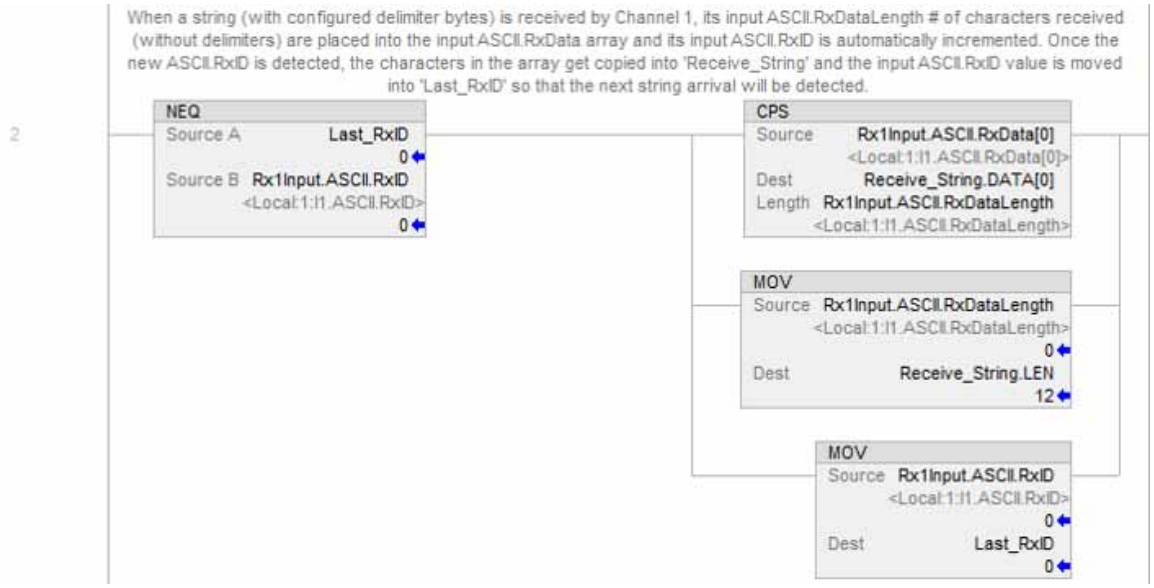
Figure 22 - Channel 0 to Channel 1 Communication



Generic ASCII Sample Code



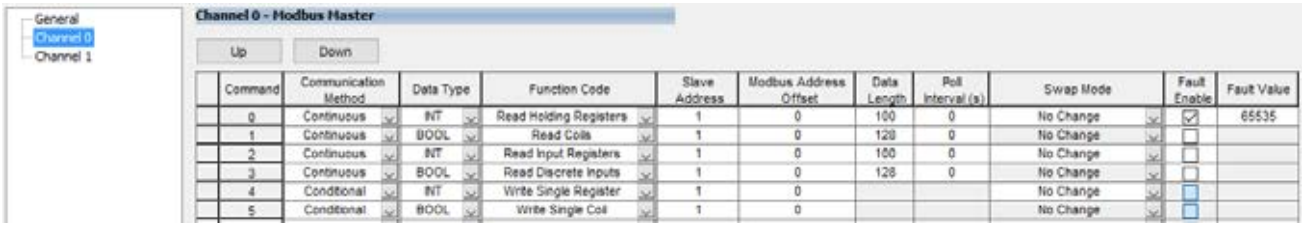

If 'Send_String' changes, copy string length # of bytes from 'Send_String' into Channel 0 ASCII.TxData array, append two '0' bytes as delimiters, and increment ASCII.TxID to transmit string along with its delimiter bytes.
1
NEQ
Source A Last_String
'Hello World'
Source B Send_String
'Hello World'
CPS
Source Send_String.DATA[0]
Dest Tx0Output.ASCII.TxData[0]
<Local:1:O0.ASCII.TxData[0]>
Length Send_String.LEN
FLL
Source 0
Dest Tx0Output.ASCII.TxData[Send_String.LEN]
<Local:1:O0.ASCII.TxData[Send_String.LEN]>
Length 2
ADD
Source A Tx0Output.ASCII.TxID
<Local:1:O0.ASCII.TxID>
0
Source B 1
Dest Tx0Output.ASCII.TxID
<Local:1:O0.ASCII.TxID>
0
MOV
Source Send_String
'Hello World'
Dest Last_String
'Hello World'


When a string (with configured delimiter bytes) is received by Channel 1, its input ASCII.RxDataLength # of characters received (without delimiters) are placed into the input ASCII.RxData array and its input ASCII.RxID is automatically incremented. Once the new ASCII.RxID is detected, the characters in the array get copied into 'Receive_String' and the input ASCII.RxID value is moved into 'Last_RxID' so that the next string arrival will be detected.
2
NEQ
Source A Last_RxID
0
Source B Rx1Input.ASCII.RxID
<Local:1:I1.ASCII.RxID>
0
CPS
Source Rx1Input.ASCII.RxData[0]
<Local:1:I1.ASCII.RxData[0]>
Dest Receive_String.DATA[0]
Length Rx1Input.ASCII.RxDataLength
<Local:1:I1.ASCII.RxDataLength>
MOV
Source Rx1Input.ASCII.RxDataLength
<Local:1:I1.ASCII.RxDataLength>
0
Dest Receive_String.LEN
12
MOV
Source Rx1Input.ASCII.RxID
<Local:1:I1.ASCII.RxID>
0
Dest Last_RxID
0

Modbus Sample Code Configuration

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

Modbus Master Command List



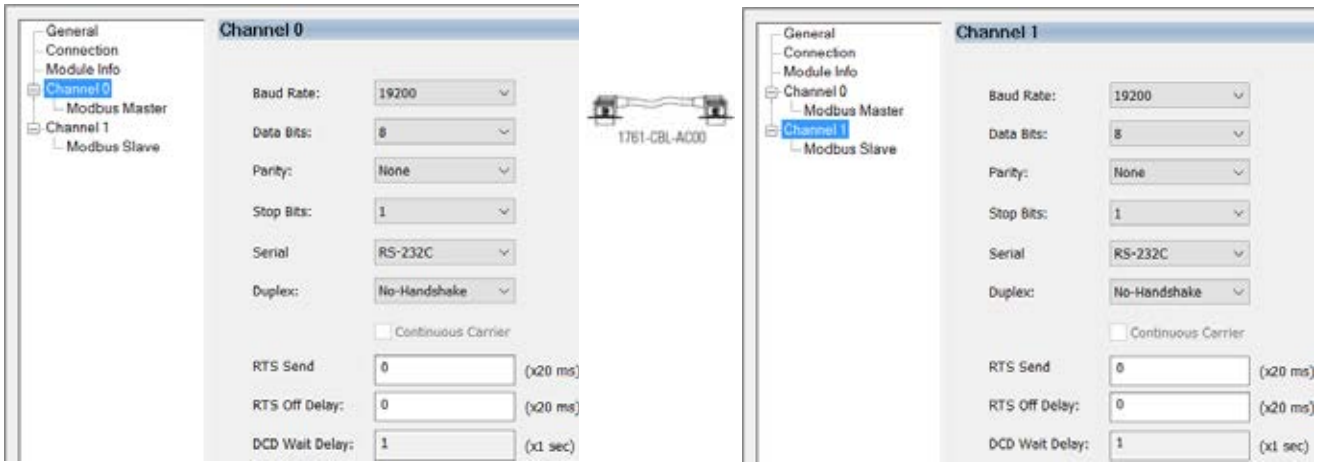
	Command	Communication Method	Data Type	Function Code	Slave Address	Modbus Address Offset	Data Length	Poll Interval (s)	Swap Mode	Fault Enable	Fault Value
	0	Continuous	INT	Read Holding Registers	1	0	100	0	No Change	☑	65535
	1	Continuous	BOOL	Read Coils	1	0	128	0	No Change	☐	
	2	Continuous	INT	Read Input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

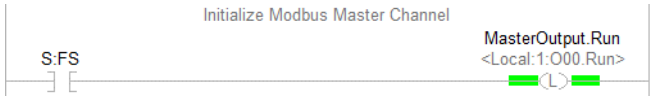
	No.	Register Type	Data Type	Register Start Address	Register Length	Data Index
	0	Holding registers	INT	0	100	0
	1	Input registers	INT	0	100	0
	2	Coils	BOOL	0	128	0
	3	Discrete inputs	BOOL	0	128	0

Modbus Sample Code Configuration Example

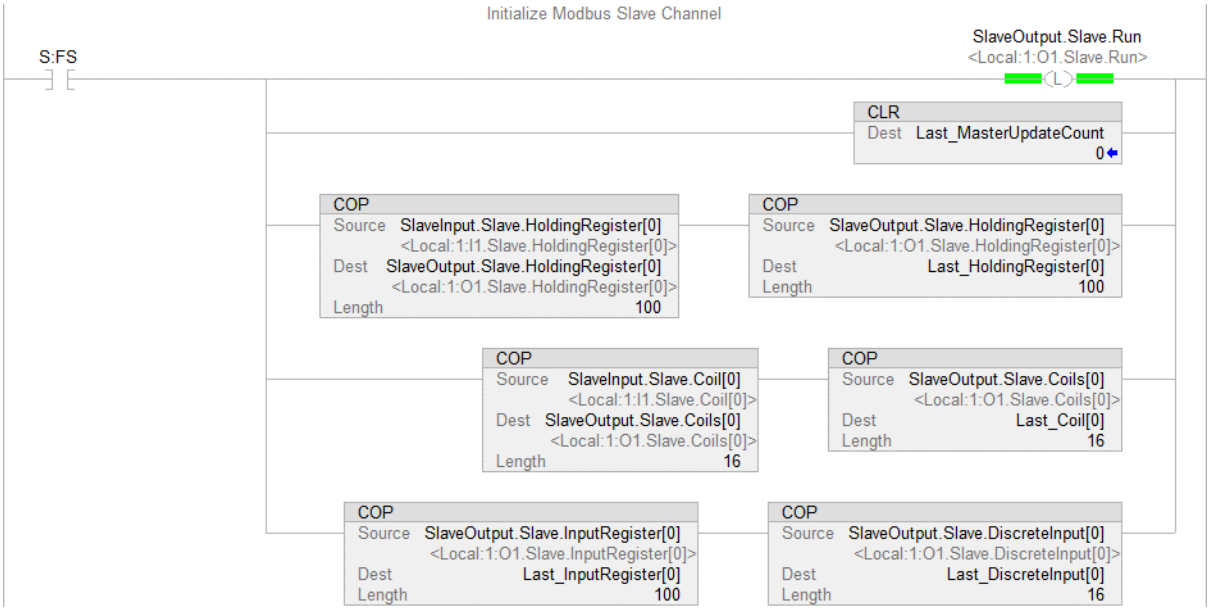
To test the Modbus capabilities by configuring Channel 0 for Master and Channel 1 for Slave, connect the two serial ports together using a 1761-CBL-AC00 cable. On the Module Properties Channel screens, verify that both channels are configured identically for RS-232C.

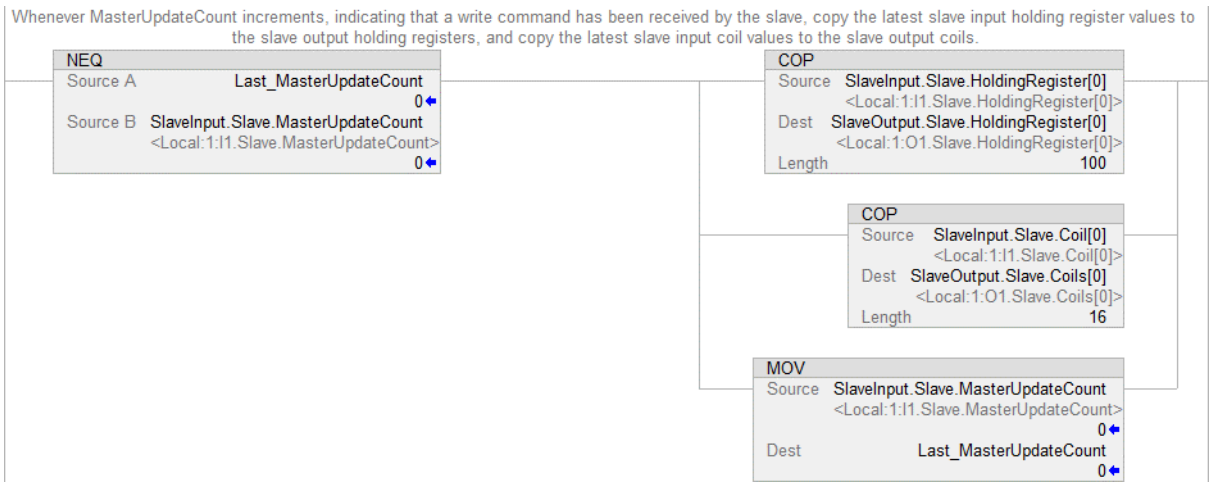


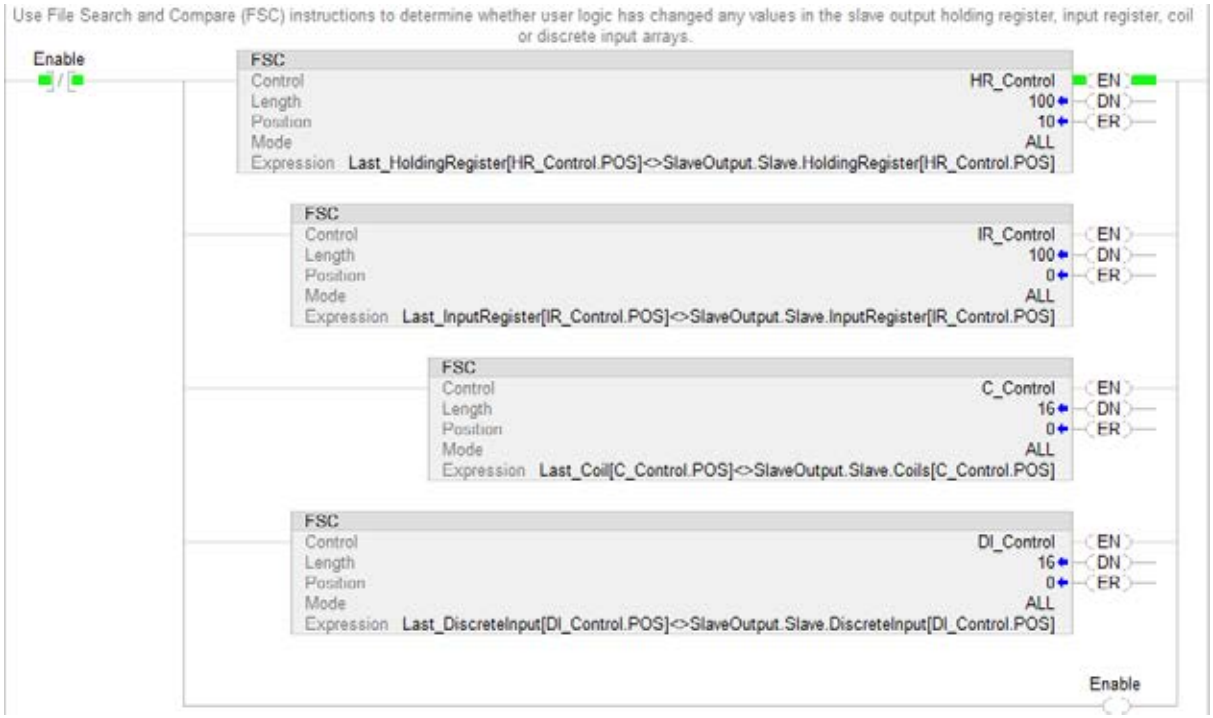
Modbus Master Sample Code

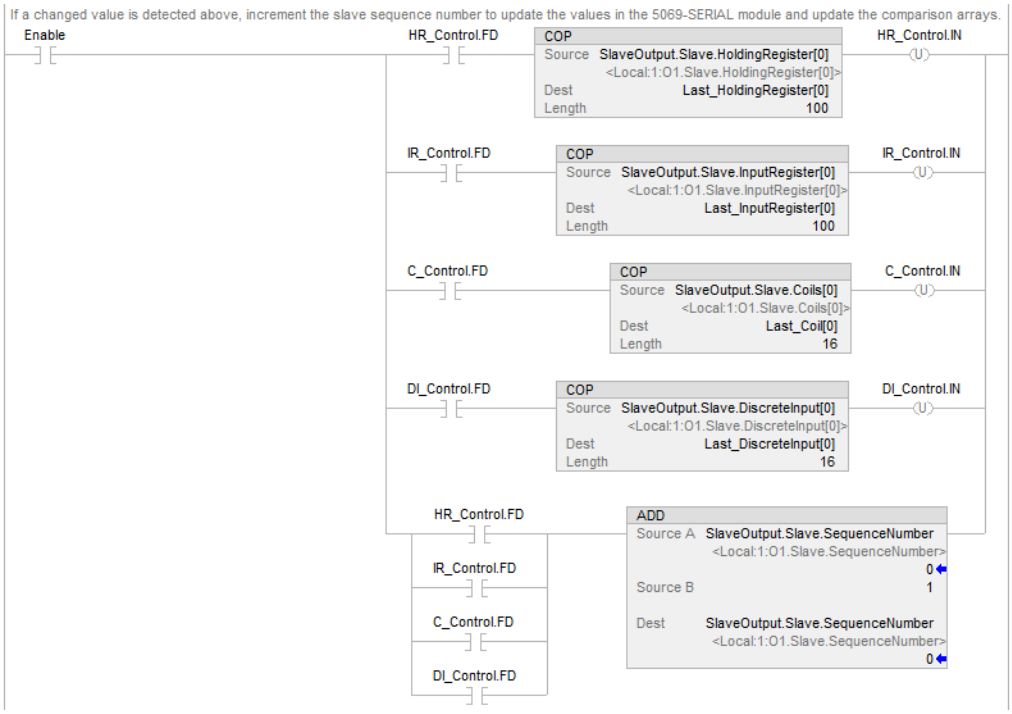


Modbus Slave Sample Code






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.Coils[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]
Enable


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]
<Local:1:O1.Slave.HoldingRegister[0]>
Dest Last_HoldingRegister[0]
Length 100
HR_Control.IN
IR_Control.FD
COP
Source SlaveOutput.Slave.InputRegister[0]
<Local:1:O1.Slave.InputRegister[0]>
Dest Last_InputRegister[0]
Length 100
IR_Control.IN
C_Control.FD
COP
Source SlaveOutput.Slave.Coils[0]
<Local:1:O1.Slave.Coils[0]>
Dest Last_Coil[0]
Length 16
C_Control.IN
DI_Control.FD
COP
Source SlaveOutput.Slave.DiscreteInput[0]
<Local:1: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
<Local:1:O1.Slave.SequenceNumber>
0
Source B 1
Dest SlaveOutput.Slave.SequenceNumber
<Local:1:O1.Slave.SequenceNumber>
0

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 23 - 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	25	31	19	0011001	Control Y, EM
1	1	01	0000001	Control A, SOH	26	32	1A	0011010	Control Z, SUB
2	2	02	0000010	Control B, STX	27	33	1B	0011011	Control Shift K, ESC
3	3	03	0000011	Control C, ETX	28	34	1C	0011100	Control Shift L, FS
4	4	04	0000100	Control D, EOT	29	35	1D	00141101	Control Shift M, GS
5	5	05	0000101	Control E, ENQ	30	36	1E	0011110	Control Shift N, RS
6	6	06	0000110	Control F, ACK	31	37	1F	0011111	Control Shift O, US
7	7	07	0000111	Control G, Rings bell	32	40	20	0100000	Space, SP
8	10	08	0001000	Control H, Backspace on some terminals	33	41	21	0100001	!
9	11	09	0001001	Control I, Horizontal tab on some terminals	34	42	22	0100010	“
10	12	0A	0001010	Control J, Line feed	35	43	23	0100011	#
11	13	0B	0001011	Control K, VT	36	44	24	0100100	$
12	14	0C	0001000	Control L, Form feed on some terminals	37	45	25	0100101	%
13	15	0D	0001101	Control M, Carraige return	38	46	26	0100110	&
14	16	0E	0001110	Control N, SO	39	47	27	0100111	’
15	17	0F	0001111	Control O, SI	40	50	28	0101000	(
16	20	10	0010000	Control P, DLE	41	51	29	0101001	)
17	21	11	0010001	Control Q, DC1	42	52	2A	0101010	*
18	22	12	0010010	Control R, DC2	43	53	2B	0101011	+
19	23	13	0010011	Control S, DC3	44	54	2C	0101100	,
20	24	14	0010100	Control T, DC4	45	55	2D	0101101	-
21	25	15	0010101	Control U, NAK	46	56	2E	0101110	.
22	26	16	0010110	Control V, SYN	47	57	2F	0101111	/
23	27	17	0010111	Control W, EB	48	60	30	0110000	0
24	30	18	0011000	Control X, CAN	49	61	31	0110001	1
50	62	32	0110010	2	85	125	55	1010101	U
51	63	33	0110011	3	86	126	56	1010110	V
52	64	34	0110100	4	87	127	57	1010111	W

Table 23 - ASCII Conversions (continued)

Decimal	Octal	Hex	Binary	ASCII Character or Control	Decimal	Octal	Hex	Binary	ASCII Character or Control
53	65	35	0110101	5	88	130	58	1011000	X
54	66	36	0110110	6	89	131	59	1011001	Y
55	67	37	0110111	7	90	132	5A	1011010	Z
56	70	38	0111000	8	91	133	5B	1011011	[
57	71	39	0111001	9	92	134	5C	1011100	\
58	72	3A	0111010	:	93	135	5D	1011101	]
59	73	3B	0111011	;	94	136	5E	1011110	^
60	74	3C	0111100	<	95	137	5F	1011111	_
61	75	3D	0111101	=	96	140	60	1100000	\
62	76	3E	0111110	>	97	141	61	1100001	a
63	77	3F	0111111	?	98	142	62	1100010	b
64	100	40	1000000	@	99	143	63	1100011	c
65	101	41	1000001	A	100	144	64	1100100	d
66	102	42	1000010	B	101	145	65	1100101	e
67	103	43	1000011	C	102	146	66	1100110	f
68	104	44	1000100	D	103	147	67	1100111	g
69	105	45	1000101	E	104	150	68	1101000	h
70	106	46	1000110	F	105	151	69	1101001	i
71	107	47	1000111	G	106	152	6A	1101010	j
72	110	48	1001000	H	107	153	6B	11010111	k
73	111	49	1001001	I	108	154	6C	1101100	l
74	112	4A	1001010	J	109	155	6D	1101101	m
75	113	4B	1001011	K	110	156	6E	1101110	n
76	114	4C	1001100	L	111	157	6F	1101111	o
77	115	4D	1001101	M	112	160	70	1110000	p
78	116	4E	1001110	N	113	161	71	1110001	q
79	117	4F	1001111	O	114	162	72	1110010	r
80	120	50	1010000	P	115	163	73	1110011	s
81	121	51	1010001	Q	116	164	74	1110100	t
82	122	52	1010010	R	117	165	75	1110101	u
83	123	53	1010011	S	118	166	76	1110110	v
84	124	54	1010100	T	119	167	77	1110111	w
120	170	78	1111000	X	124	174	7C	1111100	|
121	171	79	1111001	y	125	175	7D	1111101	}
122	172	7A	1111010	z	126	176	7E	1111110	~
123	173	7B	1111011	{	127	177	7F	1111111	DEL

Numerics

5069-ARM address reserve module 21
5069-FPD field potential distributor 22

A

access module tags 76
additional resources 7
ASCII conversion tables 107

C

configuration parameters 29
  generic ASCII 32
  modbus master 41
  modbus slave 42
configure the module 18, 43
  connections 18
    multiple connections 19
  create a new module 44
    discover local I/O modules 44
    discover remote I/O modules 49
    new local I/O modules 47
    new remote I/O module 51
connections 18
  multiple connections 19
construct the system 16
controller compatibility 13
controller organizer
  monitor tag 69
  view module tag 69
create a new module 44

D

data exchange
  handshake mode 36
  received from serial port 35
  sent to serial port 31
disable keying 27

E

edit the module configuration 54
  connection category 63
  general category 54
    module definition 55
  module info category 68
electronic keying 27
  compatible module 27
  disable keying 27
  exact match 27

F

fault reporting 25
function codes 91
  force multiple coils 99
  force single coil 96
  preset single register 98
  read coil status 91
  read holding registers 94
  read input registers 95
  read input status 93

G

generic ASCII
  input tags 78
  output tags 81

L

local I/O module 14
Logix Designer application
  tag editor 76
  view module tag 69

M

master command list 91
modbus master
  data exhange 37
    read command 38
    write command 37
  input tags 82
  output tag 85
modbus master sample code 105
modbus sample code configuration 31
modbus slave
  data exhange 39
    read command 40
    write command 39
  input tags 87
  output tags 88
modbus slave sample code 105
module definition
  dialog box 55
module overview 11
module status indicator 71
  description, digital I/O module 71 . . . 72
module tag
  viewing 69
module tags
  tag editor 76

N

name conventions 75
new local I/O modules 47

O

ownership 18

P

Power Compact 5000 I/O Serial Module 17
field-side power 17
system-side power 17
programming example 101

R

remote I/O module 15

S

serial module
diagram 12
parts 12
serial module diagram 12
serial module status indicators 73
software compatibility 13
software configurable 24
status reporting 25

T

troubleshoot 71

V

view the module tags 69

Rockwell Automation Support

Use the following resources to access support information.

Technical Support Center	Knowledgebase Articles, How-to Videos, FAQs, Chat, User Forums, and Product Notification Updates.	https://rockwellautomation.custhelp.com/
Local Technical Support Phone Numbers	Locate the phone number for your country.	http://www.rockwellautomation.com/global/support/get-support-now.page
Direct Dial Codes	Find the Direct Dial Code for your product. Use the code to route your call directly to a technical support engineer.	http://www.rockwellautomation.com/global/support/direct-dial.page
Literature Library	Installation Instructions, Manuals, Brochures, and Technical Data.	http://www.rockwellautomation.com/global/literature-library/overview.page
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	http://www.rockwellautomation.com/global/support/pcdc.page

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete the How Are We Doing? form at http://literature.rockwellautomation.com/idc/groups/literature/documents/du/ra-du002_-en-e.pdf.

Rockwell Automation maintains current product environmental information on its website at http://www.rockwellautomation.com/rockwellautomation/about-us/sustainability-ethics/product-environmental-compliance.page.

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

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

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 5069-UM003A-EN-P - May 2018