

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Visit our website - *Click HERE*



Allen-Bradley

FlexLogix Controller System User Manual

1794-L34

Firmware Revision 16

User Manual

Rockwell Automation

Important User Information

Solid state equipment has operational characteristics differing from those of electromechanical equipment. Safety Guidelines for the Application, Installation and Maintenance of Solid State Controls (publication SGI-1.1 available from your local Rockwell Automation sales office or online at http://literature.rockwellautomation.com) describes some important differences between solid state equipment and hard-wired electromechanical devices. Because of this difference, and also because of the wide variety of uses for solid state equipment, all persons responsible for applying this equipment must satisfy themselves that each intended application of this equipment is acceptable.

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.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.
ATTENTION 	Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you to identify a hazard, avoid a hazard, and recognize the consequences.
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 be dangerous temperatures.

Allen-Bradley, FlexLogix, Logix5000, RSLogix, RSLogix 5000, Rockwell Automation, RSNetWorx, and RSLinx are trademarks of Rockwell Automation, Inc.

Trademarks not belonging to Rockwell Automation are property of their respective companies.

Table of Contents

Important User Information 2

Summary of Changes

Introduction 7
Updated Information 7

Preface

Developing FlexLogix Controller Systems

Introduction 9
Related Documentation 9

Chapter 1

Where to Start

Use This Chapter 11
Design 12
Install Hardware 13

Chapter 2

Directly Connect to the Controller via the Serial Port

Use This Chapter 15
Connect the Controller via the Serial Port 15
Configure the Serial Driver 18
Select the Controller Path 20

Chapter 3

Communicate over Networks

Use This Chapter 21
EtherNet/IP 22
- Connections over EtherNet/IP 24

ControlNet 25
- Connections over ControlNet 26

DeviceNet 28
- Define Data Blocks 30

Serial 31
- Communicate with DF1 devices 33
- Communicate with ASCII devices 35
- Modbus support 38

DH-485 39
Third Party 42
- Communication Format 42
- Connection Parameters 42

Chapter 4

Manage Controller Communications

Use This Chapter 45
Produce and Consume (Interlock) Data 45
Send and Receive Messages 47
- Determine whether to cache message connections 47

Connection Overview 49
Calculate Connection Use 50
Connections Example 52

Chapter 5

Place, Configure, and Monitor I/O

Use This Chapter 53
Select I/O Modules 53
Place Local I/O Modules 54
  Selecting a Power Supply 54
Configure I/O 55
  I/O connections 57
Configure Distributed I/O on EtherNet/IP 59
Configure Distributed I/O on ControlNet 60
Configure Distributed I/O on DeviceNet 61
Address I/O Data 62
Determine When Data Is Updated 63
Monitor I/O Modules 64
  Displaying fault data 64
  Monitor a rack-optimized connection 65
Reconfigure an I/O Module 66
  Reconfigure a module via RSLogix 5000 software 66
  Reconfigure a module via a MSG instruction 67

Chapter 6

Develop Applications

Use This Chapter 69
Manage Tasks 69
Develop Programs 70
  Defining tasks 71
  Defining programs 73
  Defining routines 73
  Sample controller projects 74
Organize Tags 75
Select a Programming Language 76
  Add-On Instructions 77
Monitor Controller Status 79
Monitor Connections 80
  Determine if communication has timed out with any device 80
  Determine if communication has timed out with a specific I/O module 80
  Interrupt the execution of logic and execute the fault handler 81
Select a System Overhead Percentage 82
Use the Event Task 85
  Prioritizing Periodic and Event Tasks 85
  Triggering the Event Task 86
  Programmatically Determine if an EVENT Instruction Triggered a Task 87
  Checklist for an EVENT Instruction Task 87

Chapter 7

Configure PhaseManager

Use This Chapter 89
PhaseManager Overview 89
State Model Overview 91
How equipment changes states 92
Manually change states 94
Compare PhaseManager to Other State Models 94
Minimum System Requirements 94
Equipment Phase Instructions 95

Chapter 8

Maintain the Battery

Using this Appendix 97
Storing Replacement Batteries 97
Estimating Battery Life 98
Replacing a Battery 99

Appendix A

FlexLogix System Status Indicators

Controller LEDs 101

Appendix B

FlexLogix Back-Up on DeviceNet

Using This Appendix 103
How the Back-up Works 104
Requirements of the Back-Up 105
Power-Up and System Start-up 106
Developing the FlexLogix Back-Up Application 108
Back-up Heartbeat Configuration Rungs 108
Reading Back-up State Rung 112
Reading Back-up Status 114
Using Indicators to Check Status 115
Development and Debugging Tips 115

Appendix C

Instruction Locator Index

Where to Find an Instruction 117

Introduction

This release of this document contains new and updated information. To find new and updated information, look for change bars, as shown next to this paragraph.

Updated Information

The document contains these changes.

Topic	Page
DF1 radio modem	31
Add-On Instructions	77
Where to Find an Instruction	117

Notes:

Developing FlexLogix Controller Systems

Introduction

Use this manual to become familiar with the FlexLogix controller and its features. This version of the manual corresponds to controller firmware revision 16.

This manual describes the necessary tasks to install, configure, program, and operate a FlexLogix system. In some cases, this manual includes references to additional documentation that provides the more comprehensive details.

Related Documentation

These core documents address the Logix5000 family of controllers:

For this information:	Use this publication:
where to start for a new user of a Logix5000 controller program and test a simple project	Logix5000 Controllers Quick Start publication 1756-QS001
how to complete standard tasks program logic using sequential function charts (SFC), ladder diagram (LD), structured text (ST), and function block diagram (FBD) languages	Logix5000 Controllers Common Procedures publication 1756-PM001 **Important:** SFC and ST Programming Languages Programming Manual, publication 1756-PM003, is an excerpt from the Logix5000 Controllers Common Procedures Manual
Logix5000 controller reference: • LED patterns • controller features • instruction set quick reference	Logix5000 Controllers System Reference publication 1756-QR107
program sequential applications ladder diagram and structured text instructions	Logix5000 Controllers General Instruction Set Reference Manual publication 1756-RM003
program process control and drives applications function block diagram instructions	Logix5000 Controllers Process Control/Drives Instruction Set Reference Manual publication 1756-RM006
program motion applications ladder diagram motion instructions	Logix5000 Controllers Motion Instruction Set Reference Manual publication 1756-RM007
configure and program motion interface modules create and configure motion groups and axes configure a coordinated system time master device	Logix5000 Motion Module Configuration and Programming Manual publication 1756-UM006

The documents address network communications:

For this information:	Use this publication:
configure and use EtherNet/IP networks communicate over EtherNet/IP	EtherNet/IP Communication Modules in Logix5000 Control Systems publication ENET-UM001
configure and use ControlNet networks communicate over ControlNet	ControlNet Communication Modules in Logix5000 Control Systems publication CNET-UM001
configure and use DeviceNet network communicate over DeviceNet	DeviceNet Communication Modules in Logix5000 Control Systems publication CNET-UM004

These documents address specific controller applications:

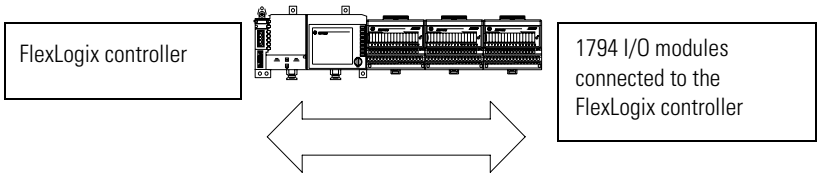
For this information:	Use this publication:
use a state model for your controller configure equipment phase programs	Logix5000 Controllers PhaseManager User Manual publication LOGIX-UM001

- To view or download manuals, visit www.rockwellautomation.com/literature.
- To obtain a hard copy of a manual, contact your local Rockwell Automation distributor or sales representative.

Where to Start

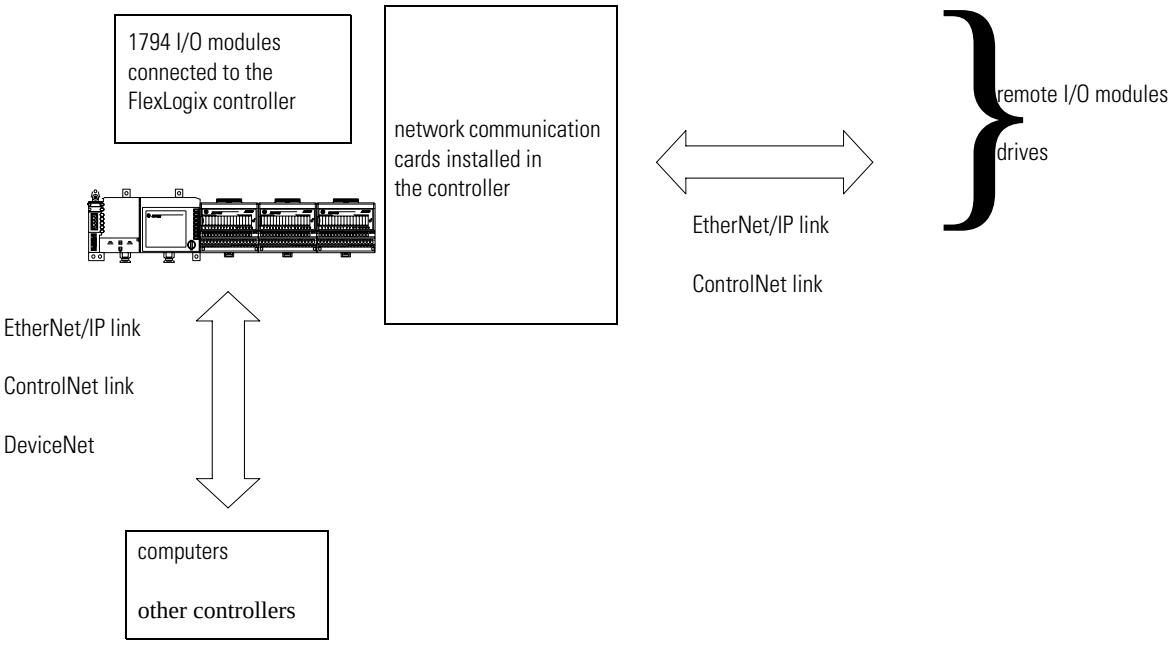
Use This Chapter

The FlexLogix controller offers state-of-art control, communications, and I/O elements in a distributed control package..



For a more flexible system, use:

- multiple controllers joined across networks
- I/O from multiple platforms that is distributed in many locations and connected over multiple I/O links



The FlexLogix controller, part of the Logix family of controllers, provides a small, powerful, cost-effective system built on the following components:

- 1794-L34 FlexLogix controller available in 512 Kbytes of user memory.
- FlexLogix controller that supports the Logix instructions.
- RSLogix 5000 programming software that supports every Logix controller.
- FLEX I/O modules that provide a compact, DIN-rail mounted I/O system.
- 1788 communication daughtercard that provides communication over standard-based ControlNet, DeviceNet or EtherNet/IP networks. The controller allows the insertion of daughtercards for up to 2 networks (e.g., one for DeviceNet and one for EtherNet/IP).

Design

See:

- *FlexLogix Selection Guide*, 1794-SG001
- *Logix5000 Controller Design Considerations Reference Manual*, 1756-RM094

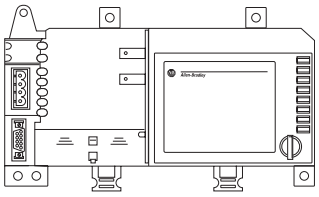
When designing a FlexLogix system, determine the network configuration and the placement of components in each location. Make these decisions as you design your system:

	Design Step
O **1.**	Select I/O devices
O **2.**	Select communication cards
O **3.**	Select controllers
O **4.**	Select power supplies
O **5.**	Select software

Install Hardware

See:

- *FlexLogix Controller Installation*



To install a FlexLogix controller, follow these steps:

	Installation Step
O **1.**	Install a DIN rail
O **2.**	Use DIN rail locks that came with your controller
O **3.**	Mount an appropriate power supply on the DIN rail
O **4.**	Install the battery in the controller See 8 "Maintain the Battery."
O **5.**	Install the communication cards in the controller See Chapter 3 "Communicate over Networks"
O **6.**	Install the controller on the DIN rail
O **7.**	Install the extended-local adapter (optional)
O **8.**	Make serial connections See Chapter 2 "Directly Connect to the Controller via the Serial Port "
O **9.**	Load controller firmware

Notes:

Directly Connect to the Controller via the Serial Port

Use This Chapter

See:

- *EtherNet/IP Modules in Logix5000 Control Systems User Manual*, ENET-UM001
- *ControlNet Modules in Logix5000 Control System User Manual*, CNET-UM001
- *DeviceNet Modules in Logix5000 Control System User Manual*, DNET-UM004

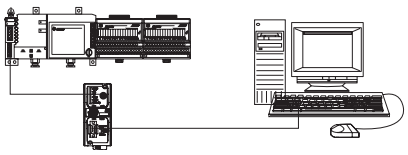
This chapter describes how to connect to the controller via the serial port so you can configure the controller and upload and/or download a project to the controller.

For this information	See
Connect the Controller via the Serial Port	15
Configure the Serial Driver	18
Select the Controller Path	20

For the FlexLogix controller to operate on a serial network, you need:

- a workstation with a serial port
- RSLinx software to configure the serial communication driver
- RSLogix5000 programming software to configure the serial port of the controller

Connect the Controller via the Serial Port

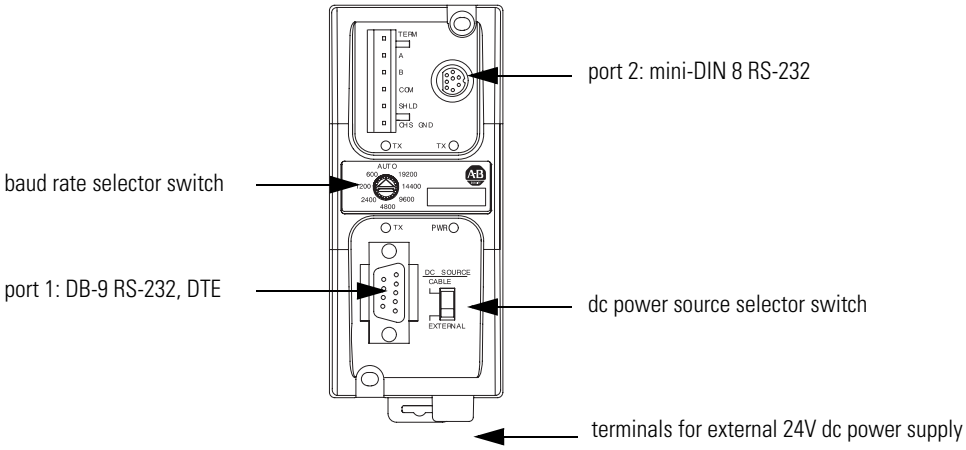


The RS-232 port is a non-isolated serial port built-in to the front of the FlexLogix controller.

1. Determine whether you need an isolator.

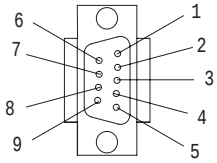
   If you connect the controller to a modem or an ASCII device, consider installing an isolator between the controller and modem or ASCII device. An isolator is also recommended when connecting the controller directly to a programming workstation.

One possible isolator is the 1761-NET-AIC interface converter.

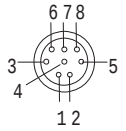


2. Select the appropriate cable.

If you are using an isolator:	Use this cable:
yes	The 1761-CBL-AP00 cable (right-angle bend connector to controller) or the 1761-CBL-PM02 cable (straight connector to the controller) attaches the controller to port 2 on the 1761-NET-AIC isolator. The 8-pin mini-DIN connector is not commercially available, so you cannot make this cable.



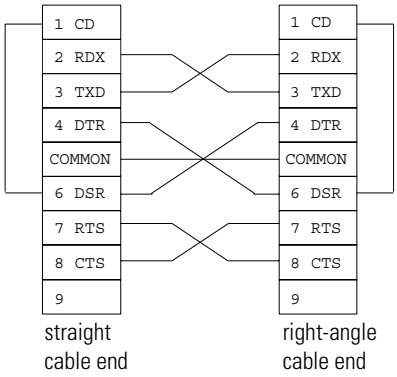
DB-9 right-angle or straight cable end



8-pin, mini-DIN cable end

Pin:	DB-9 end:	Mini-DIN end:
1	DCD	DCD
2	RxD	RxD
3	TxD	TxD
4	DTR	DTR
5	ground	ground
6	DSR	DSR
7	RTS	RTS
8	CTS	CTS
9	na	na

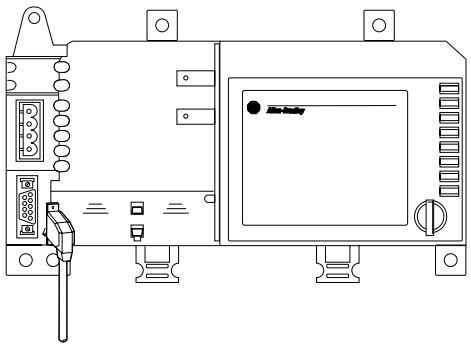
If you are using an isolator:	Use this cable:
no	The 1756-CP3 cable attaches the controller directly to the RS-232 device.



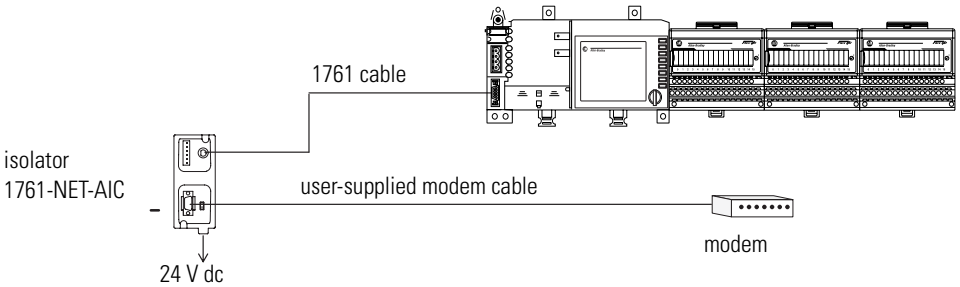
If you make your own cable, it must be shielded and the shields must be tied to the metal shell (that surrounds the pins) on both ends of the cable.

You can also use a 1747-CP3 cable from the SLC product family. This cable has a larger right-angle connector than the 1756-CP3 cable.

3. Connect the appropriate cable to the serial port on the controller.



4. If necessary, attach the controller to the isolator.



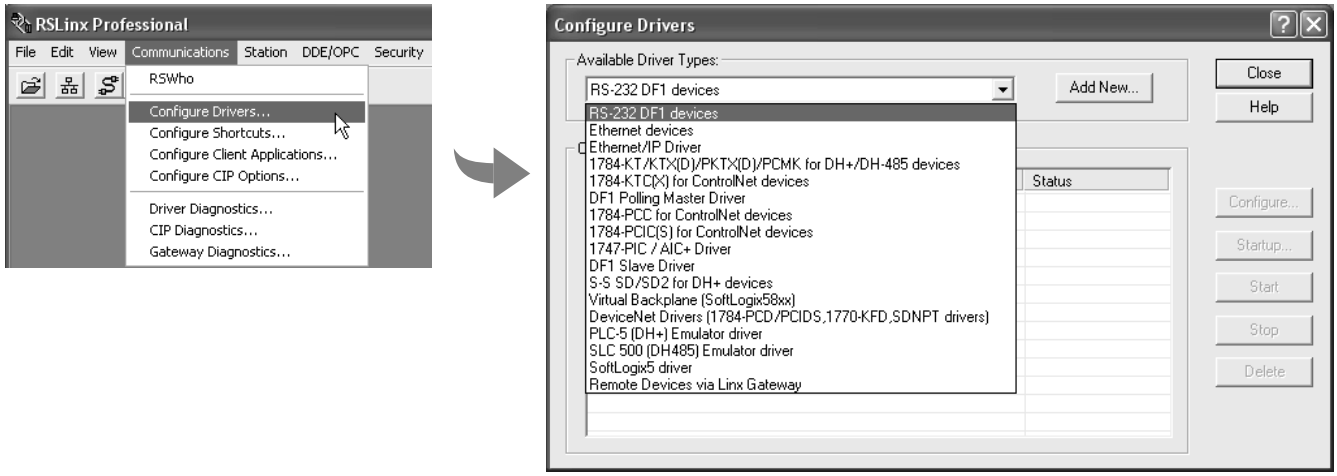
ATTENTION

The FlexLogix controller is grounded through its DIN rail. It is important that you understand the workstation's grounding system before connecting it to the controller. An isolator is recommended between the controller and the workstation.

Configure the Serial Driver

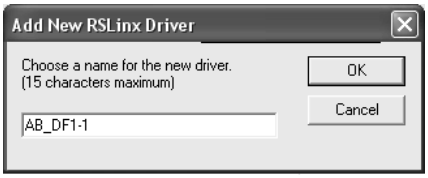
Use RSLinx software to configure the RS-232 DF1 Device driver for serial communications. To configure the driver:

1. From the Communications menu in RSLinx software, select Configure Drivers. Choose the RS-232 DF1 Device driver.



2. Click Add New to add the driver.

3. Specify the driver name and click OK.



4. Specify the serial port settings:
   a. From the Comm Port drop-down list, select the serial port (on the workstation) that the cable is connected to.
   b. From the Device drop-down list, select Logix 5550-Serial Port.
   c. Click Auto-Configure.



5. Does the dialog box display the following message:

   Auto Configuration Successful!

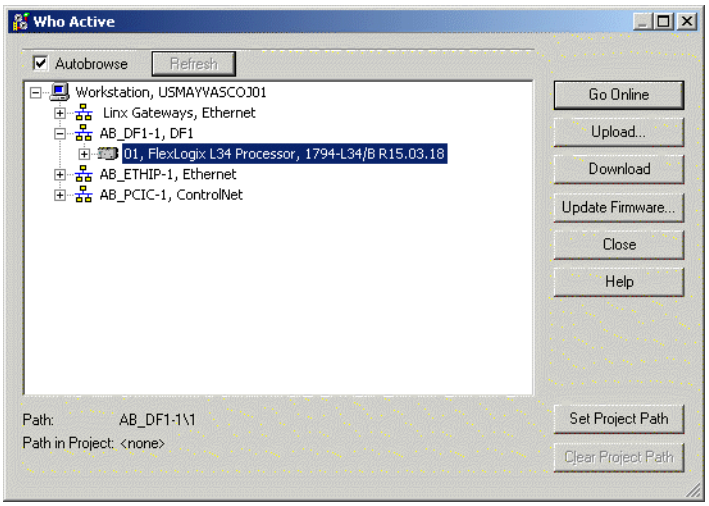
If:	Then:
Yes	Click *OK*.
No	Go to step 4. and verify that you selected the correct Comm Port.

   Then click Close.

Select the Controller Path

In RSLogix 5000 software, select the controller path on the network.

1. Open an RSLogix 5000 project for the controller.
2. From the Communications menu, select Who Active.
3. Expand the communication driver to the level of the controller.



4. Select the controller.

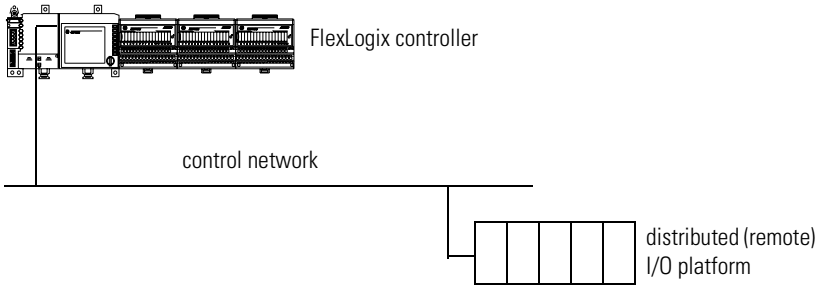
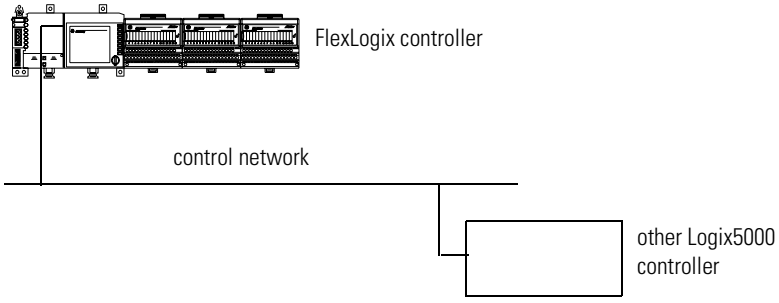
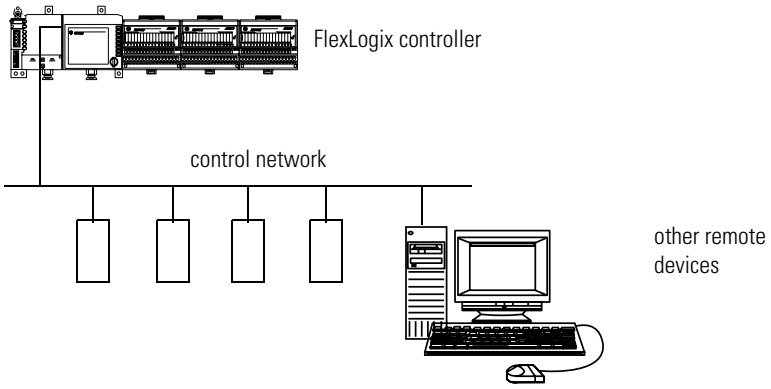
To:	Choose:
monitor the project in the controller	Go Online
transfer a copy of the project from the controller to RSLogix 5000 software	Upload
transfer the open project to the controller	Download

You may have to confirm the action.

Communicate over Networks

Use This Chapter

The FlexLogix controller supports additional networks so that the controller can:

Supported networks for:	Example:
Control distributed (remote) I/O • EtherNet/IP • ControlNet • DeviceNet	
Produce/consume (interlock) data between controllers • EtherNet/IP • ControlNet	
Send and receive messages to and from other devices (this includes access to the controller via RSLogix 5000 programming software) • EtherNet/IP • ControlNet • DeviceNet (to devices only) • serial • DH-485	

This chapter summarizes the FlexLogix controller's communications capabilities:

For this information:	See:
EtherNet/IP	22
ControlNet	25
DeviceNet	28
Serial	31
DH-485	39
Third Party	42

EtherNet/IP

See:

- *EtherNet/IP Modules in Logix5000 Control Systems User Manual*, ENET-UM001
- *EtherNet/IP Web Server Module User Manual*, ENET-UM527
- *EtherNet/IP Performance Application Guide*, ENET-AP001
- *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094
- *EtherNet/IP Daughtercard Installation Instructions*, 1788-IN054

For EtherNet/IP communications, install a 1788-ENBT communication card in your FlexLogix controller.

Use these software products when you use a FlexLogix controller on EtherNet/IP:

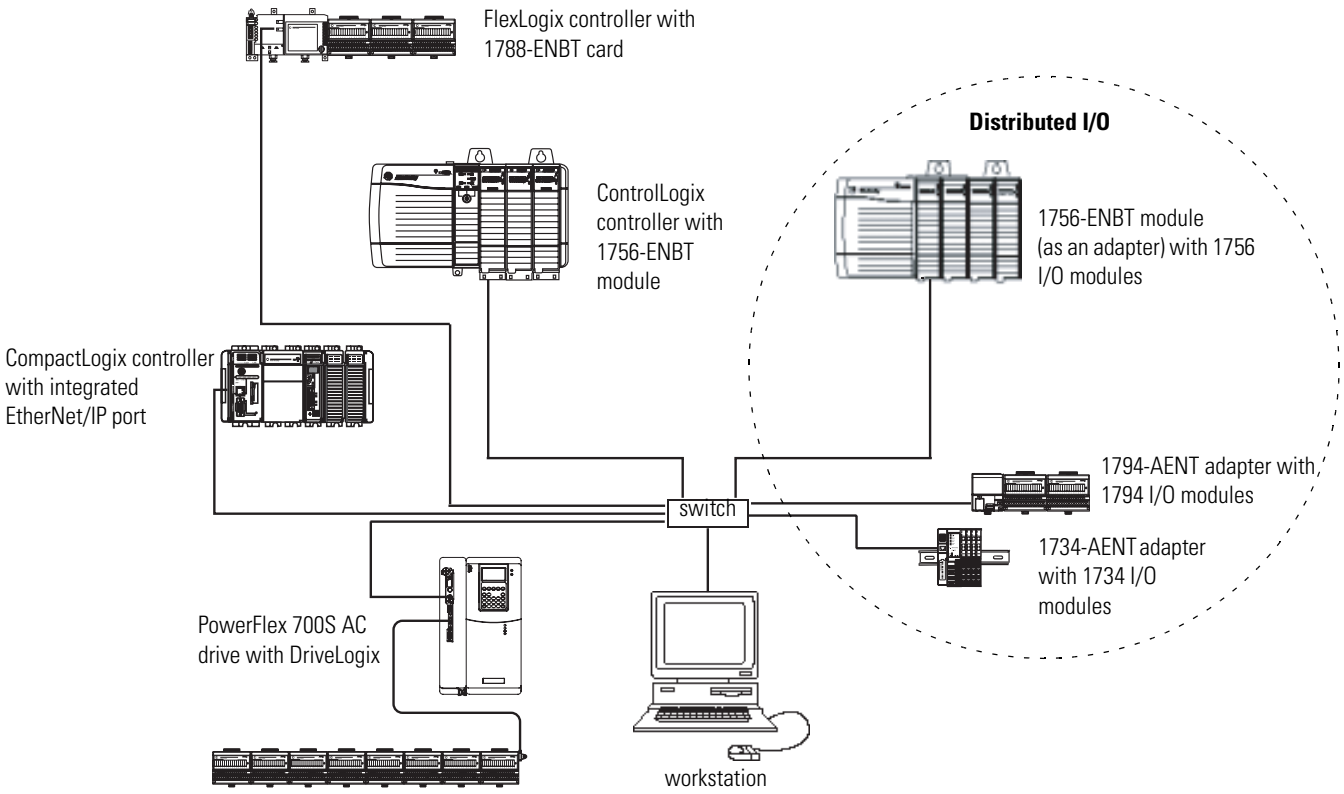
Software	Use	Required/optional
RSLogix 5000 programming software	Use this to configure the FlexLogix project and define EtherNet/IP communications.	Required
BOOTP/DHCP Utility	This utility comes with RSLogix 5000 software. Use this utility to assign IP addresses to devices on an EtherNet/IP network.	Optional
RSNetWorx for EtherNet/IP	Use this software to configure EtherNet/IP devices by IP addresses and/or host names.	Optional

The EtherNet/IP communication modules:

- support messaging, produced/consumed tags, HMI, and distributed I/O
- encapsulate messages within standard TCP/UDP/IP protocol
- share a common application layer with ControlNet and DeviceNet
- interface via RJ45, category 5, unshielded, twisted-pair cable
- support half/full duplex 10 Mbps or 100 Mbps operation
- support standard switches
- require no network scheduling
- require no routing tables

In this example:

- The controllers can produce and consume tags among each other.
- The controllers can initiate MSG instructions that send/receive data or configure devices.
- The personal computer can upload/download projects to the controllers.
- The personal computer can configure devices on EtherNet/IP.



Connections over EtherNet/IP

You indirectly determine the number of connections the controller uses by configuring the controller to communicate with other devices in the system. Connections are allocations of resources that provide more reliable communications between devices than unconnected messages.

All EtherNet/IP connections are unscheduled. An unscheduled connection is a message transfer between controllers that is triggered by the requested packet interval (RPI) or the program (such as a MSG instruction). Unscheduled messaging lets you send and receive data when needed.

The 1788-ENBT card supports 32 CIP connections over an EtherNet/IP network. With these controllers, the number of end-node connections they effectively support is dependent on the RPI of the connection:

If the RPI is:	The EtherNet/IP card effectively supports a maximum of this many communication connections:
2 ms	2
4 ms	5
8 ms	10
16 ms	18
32 ms +	25

In the table above, with an RPI of 32 ms and greater, the EtherNet/IP card effectively supports 25 communications connections. In this case, the remaining 7 connections can be used for non-I/O purposes.

For more information...

The *EtherNet/IP Modules in Logix5000 Control Systems User Manual*, ENET-UM001 provides information on how to:

- configure an EtherNet/IP communication module
- control I/O over EtherNet/IP
- send a message over EtherNet/IP
- produce/consume a tag over EtherNet/IP
- monitor diagnostics
- calculate controller connections over EtherNet/IP

The *Logix5000 Controllers Design Guidelines Reference Manual*, 1756-RM094 provides guidelines on optimizing a control application on an EtherNet/IP network.

ControlNet

See:

- *ControlNet Modules in Logix5000 Control Systems User Manual*, CNET-UM001
- *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094
- *ControlNet Daughtercard Installation Instructions*, 1788-IN002
- *ControlNet Daughtercard Installation Instructions*,

For ControlNet communications, install a ControlNet communication card in your FlexLogix controller:

If you are using	Use this card
fiber media	1788-CNF, 1788-CNFR
coaxial media	1788-CNC, 1788-CNCR

Use these software products when you use a FlexLogix controller on ControlNet:

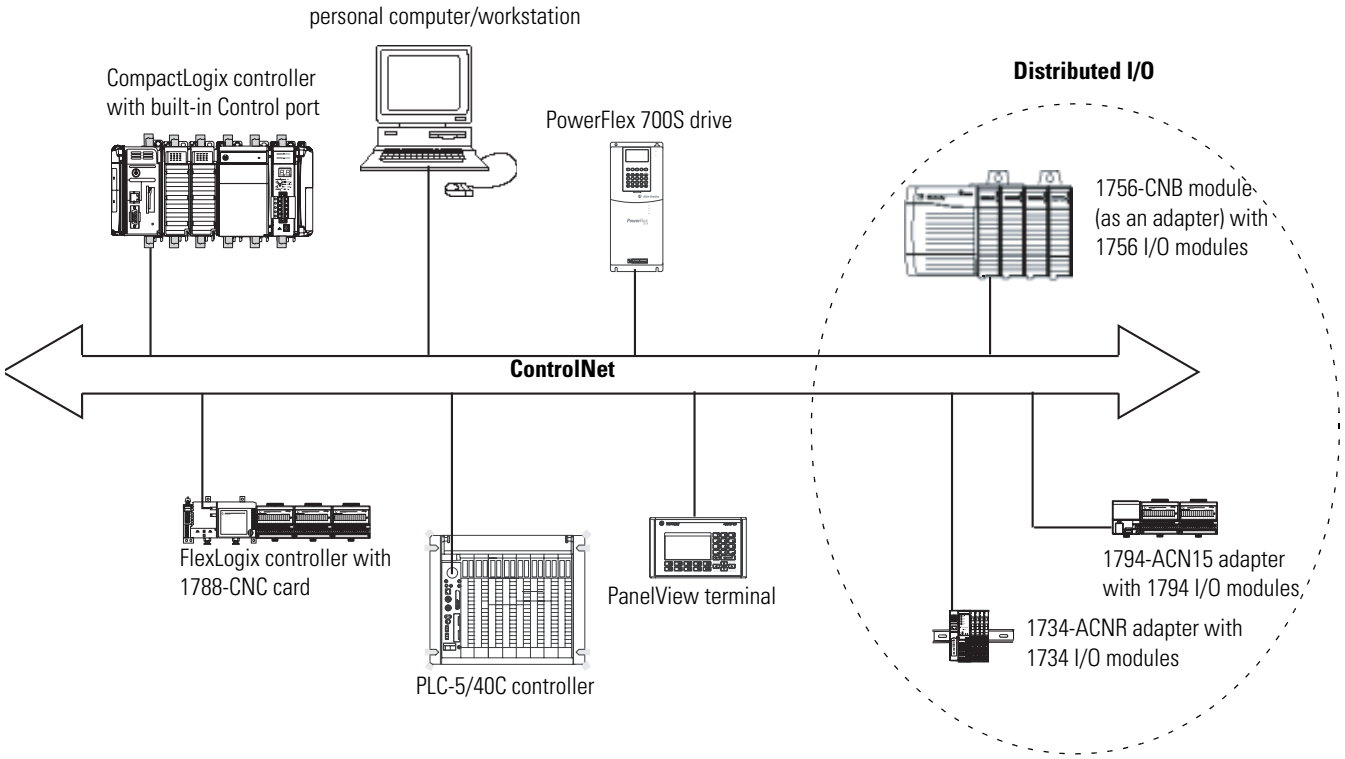
Software	Use	Required/optional
RSLogix 5000 programming software	Use this to configure the FlexLogix project and define ControlNet communications.	Required
RSLinx	Use this software to configure the ControlNet communication driver.	Required
RSNetWorx for ControlNet	Use this software to configure the ControlNet network, define the NUT (Network update time), and schedule the ControlNet network.	Required

The ControlNet communications modules:

- support messaging, produced/consumed tags and distributed I/O
- share a common application layer with DeviceNet and EtherNet/IP
- require no routing tables
- support the use of coax and fiber repeaters for isolation and increased distance

In this example:

- The controllers can produce and consume tags among each other.
- The controllers can initiate MSG instructions that send/receive data or configure devices.
- The personal computer can upload/download projects to the controllers.
- The personal computer can configure devices on ControlNet, and it can configure the network itself.



Connections over ControlNet

You indirectly determine the number of connections the controller uses by configuring the controller to communicate with other devices in the system. Connections are allocations of resources that provide

more reliable communications between devices compared to unconnected messages.

ControlNet connections can be:

Connection method:	Description:
scheduled (unique to ControlNet)	A scheduled connection is unique to ControlNet communications. A scheduled connection lets you send and receive data repeatedly at a predetermined interval, which is the requested packet interval (RPI). For example, a connection to an I/O module is a scheduled connection because you repeatedly receive data from the module at a specified interval. Other scheduled connections include connections to: • communication devices • produced/consumed tags On a ControlNet network, you must use RSNetWorx for ControlNet to enable all scheduled connections and establish a network update time (NUT). Scheduling a connection reserves network bandwidth to specifically handle the connection.
unscheduled	An unscheduled connection is a message transfer between controllers that is triggered by ladder logic or the program (such as a MSG instruction). Unscheduled messaging lets you send and receive data when needed. Unscheduled messages use the remainder of network bandwidth after scheduled connections are allocated.

The FlexLogix controller supports 100 connections. However, the controller is limited by the number of connections each ControlNet communication card supports. The 1788-CNx cards support 32 total ControlNet connections, 22 of which can be scheduled and used for producing and consuming tags. With these controllers, the number of end-node connections they effectively support is dependent on the application's NUT and RPI:

If the NUT and RPI are each:	The controllers effectively support this many communication connections
5 ms	3
10 ms	6
20 ms	13
40 ms +	22

In the table above, with a NUT and RPI of 40 ms and greater, the ControlNet card supports 22 communications connections. In this case, the remaining 10 connections can be used for unscheduled connections.

For more information... The *ControlNet Modules in Logix5000 Control Systems User Manual*, CNET-UM001 provides information on how to:

- configure a ControNet communication module
- control I/O over ControlNet
- send a message over ControlNet
- produce/consume a tag over ControlNet
- calculate controller connections over ControlNet

The *Logix5000 Controllers Design Guidelines Reference Manual*, 1756-RM094 provides guidelines on optimizing a control application on a ControlNet network.

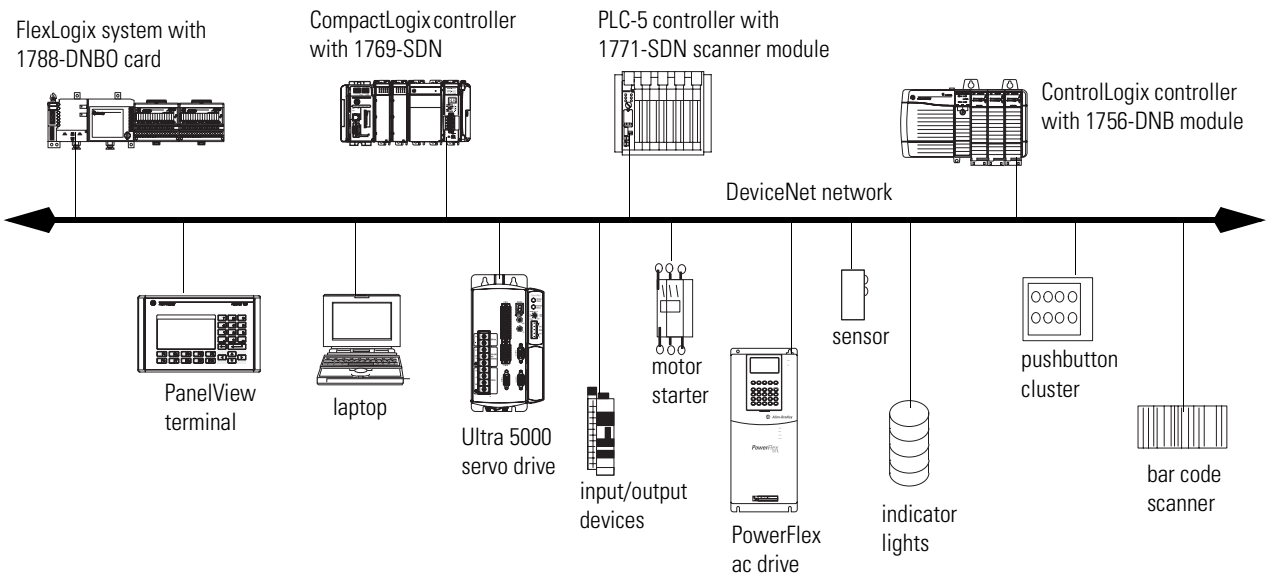
DeviceNet

See:

- *DeviceNet Modules in Logix5000 Control Systems User Manual*, DNET-UM004
- *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094
- *DeviceNet Daughtercard*

The DeviceNet network uses the Common Industrial Protocol (CIP) to provide the control, configuration, and data collection capabilities for industrial devices.

For DeviceNet communications, install a 1788-DNBO communication card in your FlexLogix controller.



Use these software products when you use a FlexLogix controller on DeviceNet:

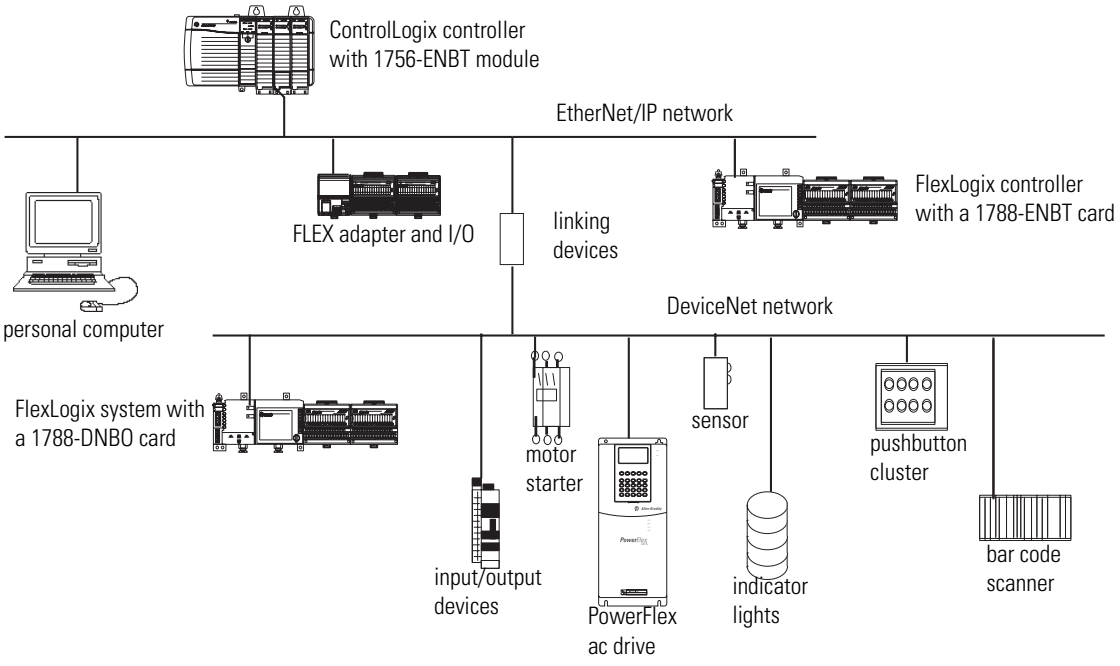
Software	Use	Required/optional
RSLogix 5000 programming software	Use this to configure the FlexLogix project and define DeviceNet communications.	Required
RSNetWorx for DeviceNet	Use this software to configure DeviceNet devices and define the scan list for those devices.	Required

The DeviceNet communications module:

- supports messaging to devices (not controller to controller)
- shares a common application layer with ControlNet and EtherNet/IP
- offers diagnostics for improved data collection and fault detection
- requires less wiring than traditional, hardwired systems

You can use a linking device as a:

- gateway to connect information- or control-level networks to device-level networks for programming, configuration, control or data collection
- router/bridge to connect the EtherNet/IP or ControlNet network to the DeviceNet network



Define Data Blocks

How you configure the DeviceNet devices determines how many words you use per device. The 1788-DNBO card supports a maximum of:

- 124 32-bit words of input data
- 123 32-bit words of output data
- 32 32-bit words of status data

Most DeviceNet devices support 16-bit words. Take care how you map these into the 32-bit words used in RSLogix 5000 programming software. RSNetWorx for DeviceNet lets you DINT-align the device data. While this might simplify the organization of the data, it might also limit the data you have available.

For more information...

The *DeviceNet Modules in Logix5000 Control Systems User Manual*, DNET-UM004 provides information on how to:

- configure DeviceNet communication module
- control devices on DeviceNet

The *Logix5000 Controllers Design Guidelines Reference Manual*, 1756-RM094 provides guidelines on optimizing a control application on a DeviceNet network.

Serial

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001

The RS-232 port is a non-isolated serial port built-in to the front of the FlexLogix controller. You can configure the serial port of the controller for these modes:

Use this mode:	For:
DF1 point-to-point	communication between the controller and one other DF1-protocol-compatible device. This is the default system mode. Default parameters are: • Baud Rate: 19200 • Data Bits: 8 • Parity: None • Stop Bits: 1 • Control Line: No Handshake • RTS send Delay: 0 • RTS Off Delay: 0 This mode is typically used to program the controller through its serial port.
DF1 master mode	control of polling and message transmission between the master and slave nodes. The master/slave network includes one controller configured as the master node and as many as 254 slave nodes. Link slave nodes using modems or line drivers. A master/slave network can have node numbers from 0 to 254. Each node must have a unique node address. Also, at least 2 nodes must exist to define your link as a network (1 master and 1 slave station are the two nodes).
DF1 radio modem	• Compatible with SLC500 and MicroLogix1500 controllers • This mode supports master and slave, and store and forward modes

Use this mode:	For:
DF1 slave mode	using a controller as a slave station in a master/slave serial communication network. When there are multiple slave stations on the network, link slave stations using modems or line drivers to the master. When you have a single slave station on the network, you do not need a modem to connect the slave station to the master. You can configure the control parameters for no handshaking. You can connect 2 to 255 nodes to a single link. In DF1 slave mode, a controller uses DF1 half-duplex protocol. One node is designated as the master and it controls who has access to the link. All the other nodes are slave stations and must wait for permission from the master before transmitting.
User mode (channel 0 only)	communicating with ASCII devices. This requires your program to use ASCII instructions to read and write data from and to an ASCII device.
DH-485	communicating with other DH-485 devices multi-master, token passing network allowing programming and peer-to-peer messaging.

1. Determine whether you need an isolator.

   For more information on determining if you need an isolator, see
   page 15.

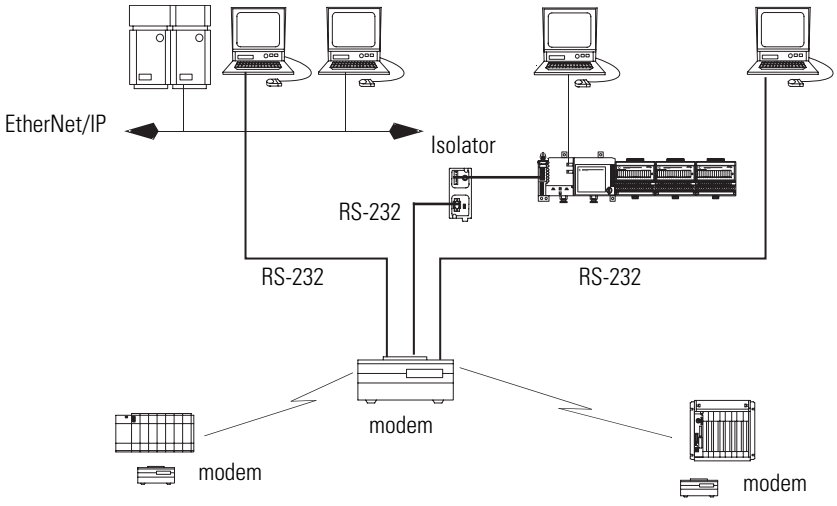
2. Select the appropriate cable.

3. Connect the appropriate cable to the serial port.

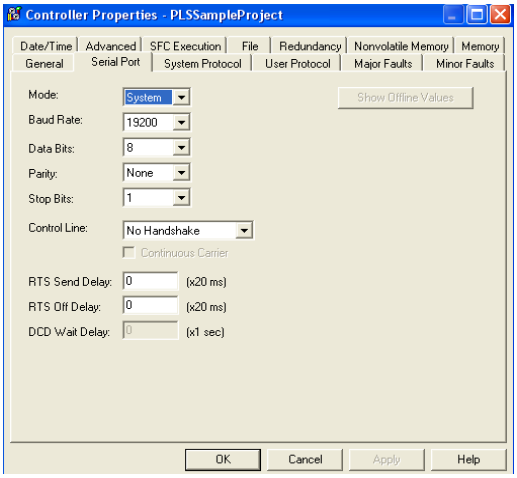
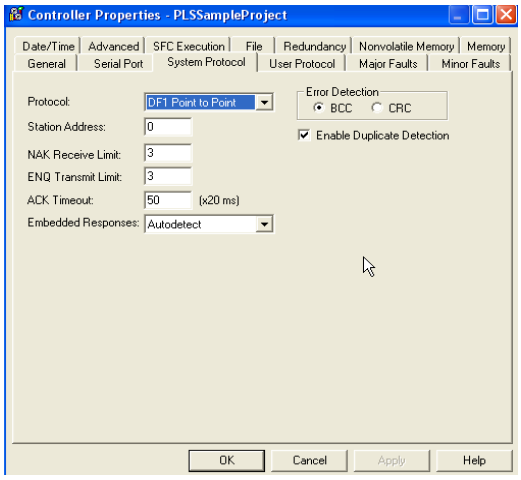
Communicate with DF1 devices

You can configure the controller as a master or slave on a serial communication network. Use serial to get information to and from remote controllers (stations) when:

- the system contains three or more stations
- communications occur on a regular basis and require leased-line, radio, or power-line modems



To configure the controller for DF1 communications:

On this tab	Do this
	1. Select System Mode 2. Specify communication settings
	1. Select DF1 protocol 2. Specify DF1 settings

For more information... The *Logix5000 Controllers General Instructions Reference Manual*, 1756-RM003 defines the instructions you can use to manipulate ASCII characters.

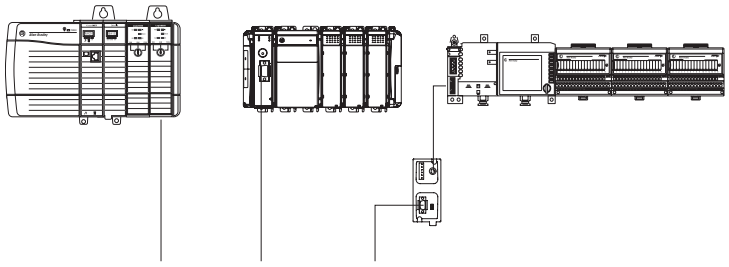
The *SCADA System Application Guide*, AG-UM008 provides information on how to:

- select a polling mode
- configure controllers, modems, and software
- troubleshoot basic DF1 protocol issues

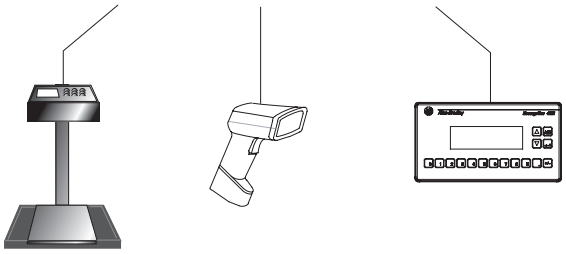
Communicate with ASCII devices

When configured for user mode, you can use the serial port to interface with ASCII devices. For example, you can use the serial port to:

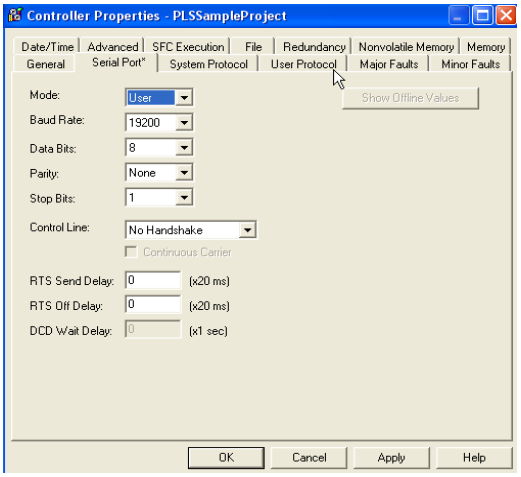
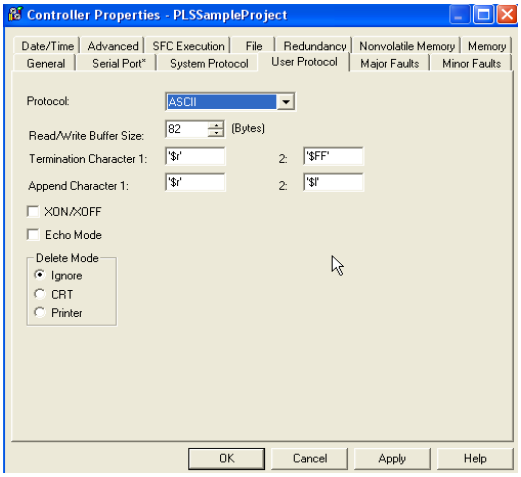
- read ASCII characters from a weigh scale module or bar code reader
- send and receive messages from an ASCII triggered device, such as a MessageView terminal.



connection from the serial port of the controller to the ASCII device



To configure the controller for ASCII communications:

On this tab	Do this
	1. Select User Mode 2. Specify communication settings
	1. Select ASCII protocol 2. Specify ASCII character settings

The controller supports several instructions to manipulate ASCII characters. The instructions are available in ladder diagram (LD) and structured text (ST).

Read and write ASCII characters

If you want to:	Use this instruction:
determine when the buffer contains termination characters	ABL
count the characters in the buffer	ACB
clear the buffer	ACL
clear out ASCII Serial Port instructions that are currently executing or are in the queue	
obtain the status of the serial port control lines	AHL
turn on or off the DTR signal	
turn on or off the RTS signal	
read a fixed number of characters	ARD
read a varying number of characters, up to and including the first set of termination characters	ARL
send characters and automatically append one or two additional characters to mark the end of the data	AWA
send characters	AWT

Create and modify strings of ASCII characters

If you want to:	Use this instruction:
add characters to the end of a string	CONCAT
delete characters from a string	DELETE
determine the starting character of a sub-string	FIND
insert characters into a string	INSERT
extract characters from a string	MID

Convert data to or from ASCII characters

If you want to:	Use this instruction:
convert the ASCII representation of an integer value to a SINT, INT, DINT, or REAL value	STOD
convert the ASCII representation of a floating-point value to a REAL value	STOR
convert a SINT, INT, DINT, or REAL value to a string of ASCII characters	DTOS
convert a REAL value to a string of ASCII characters	RTOS
convert the letters in a string of ASCII characters to upper case	UPPER
convert the letters in a string of ASCII characters to lower case	LOWER

For more information... The *Logix5000 Controllers General Instructions Reference Manual*, 1756-RM003 defines the instructions you can use to manipulate ASCII characters.

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- communicate with an ASCII device
- transmit/receive ASCII characters

Modbus support

See:

- *Logix5000 Controllers as Masters or Slaves on Modbus Application Solution*, CIG-AP129

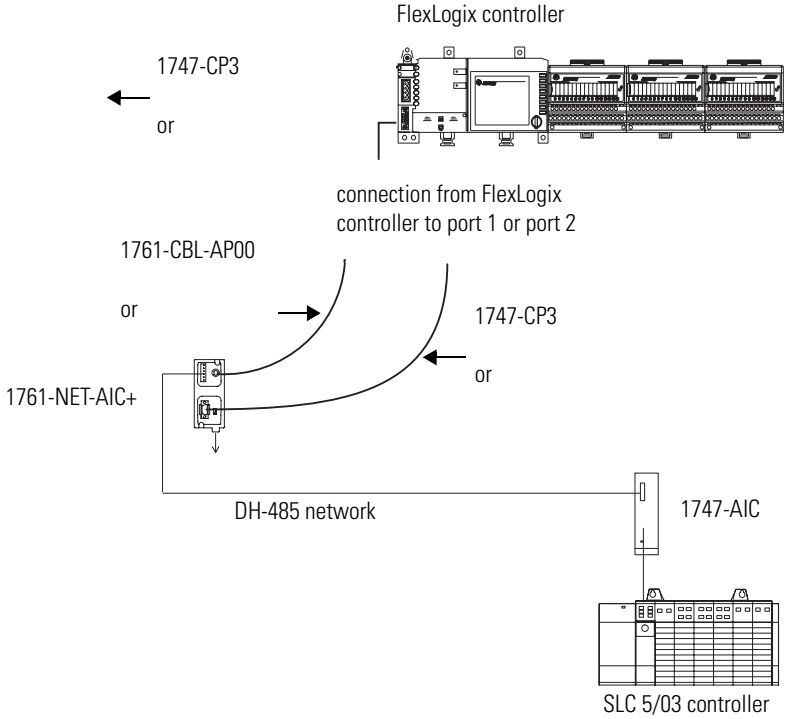
To use Logix5000 controllers on Modbus, you connect through the serial port and execute specific ladder logic routines. A sample controller project is available with RSLogix 5000 Enterprise programming software. From RSLogix 5000 software, select Help → Vendor Sample Projects to display a list of available, sample projects.

DH-485

For DH-485 communication, use the serial port of the controller. However, when using a FlexLogix controller, it is recommended that you use NetLinx networks (EtherNet/IP, ControlNet, or DeviceNet) because excessive traffic on a DH-485 network may make it impractical to connect to a controller with RSLogix 5000 programming software.

If your application uses:	Select:
• connections to existing DH-485 networks	built-in serial port

The DH-485 protocol uses RS-485 half-duplex as its physical interface. (RS-485 is a definition of electrical characteristics; it is not a protocol.) You can configure the RS-232 port of the FlexLogix controller to act as a DH-485 interface. By using a 1761-NET-AIC and the appropriate RS-232 cable (1756-CP3 or 1747-CP3), a FlexLogix controller can send and receive data on a DH-485 network.



On the DH-485 network, the FlexLogix controller can send and receive messages to and from other controllers on the network.

IMPORTANT A DH-485 network consists of multiple cable segments. Limit the total length of all the segments to 1219m (4000 ft.).

For the controller to operate on a DH-485 network, you need:

- a 1761-NET-AIC interface converter for each controller you want to put on the DH-485 network.

  You can have two controllers for each 1761-NET-AIC converter, but you need a different cable for each controller.

  a. Connect the serial port of the controller to either port 1 or port 2 of the 1761-NET-AIC converter.
  b. Use the RS-485 port to connect the converter to the DH-485 network.

  The cable you use to connect the controller depends on the port you use on the 1761-NET-AIC converter.

If you connect to this port:	Use this cable:
port 1 DB-9 RS-232, DTE connection	1747-CP3 or 1761-CBL-AC00
port 2 mini-DIN 8 RS-232 connection	1761-CBL-AP00 or 1761-CBL-PM02

- RSLogix 5000 programming software to configure the serial port of the controller for DH-485 communications.

  Specify these characteristics on the Serial Port tab (default values are shown in bold):

Characteristic:	Description:
Baud Rate	Specifies the communication rate for the DH-485 port. All devices on the same DH-485 network must be configured for the same baud rate. Select 9600 or 19200 Kbps.
Node Address	Specifies the node address of the controller on the DH-485 network. Select a number 1-31 decimal, inclusive. To optimize network performance, assign node addresses in sequential order. Initiators, such as personal computers, should be assigned the lowest address numbers to minimize the time required to initialize the network.
Token Hold Factor	Number of transmissions (plus retries) that a node holding a token can send onto the data link each time that it receives the token. Enter a value between 1-4. The default is 1.
Maximum Node Address	Specifies the maximum node address of all the devices on the DH-485 network. Select a number 1-31 decimal, inclusive. To optimize network performance, make sure: • the maximum node address is the highest node number being used on the network • that all the devices on the same DH-485 network have the same selection for the maximum node address.

For more information... The *Data Highway/Data Highway Plus/Data Highway II/Data Highway-485 Cable Installation Manual*, 1770-6.2.2 describes how to plan and install a DH-485 network.

Third Party

The FlexLogix controller can operate on third-party networks. To operate on a third-party network, install the 1788-MODULE generic module communication card in the controller.

Use these software products when you use a FlexLogix controller on third-party network:

Software	Use	Required/optional
RSLogix 5000 programming software, Version 12 or later	Use this to configure the 1788-MODULE card as part of the FlexLogix system	Required
Third-party software	Software that configures the 1788-MODULE card on the third-party network	Required

Use RSLogix 5000 programming software to map the 1788-MODULE card as part of the FlexLogix system. In the Controller Organizer, add the card to the I/O Configuration folder.

Communication Format

The Communication Format field chooses a data type for information transmitted between the controller and a remote device connected to the 1788-MODULE communication card. This format creates an array in the controller of whatever data type you choose for the input and output data.

Connection Parameters

You must set connection parameters to define data identification and connection size. An Assembly Instance and Data Size must be assigned for input, output and configuration data.

Assembly Instance

The Assembly Instance is a number that identifies what data transferred between the owner-controller and I/O module looks like. You must create a map that defines your assembly instance entries.

Size

The size field determines how large the connections are between the owner-controller and the I/O module. Connections are sent in sizes matching the communications format data type selected. The default, DINT, results in 32-bit quantities.

Complete your system configuration and develop your program logic. Then download the project to the controller.

Notes:

Manage Controller Communications

Use This Chapter

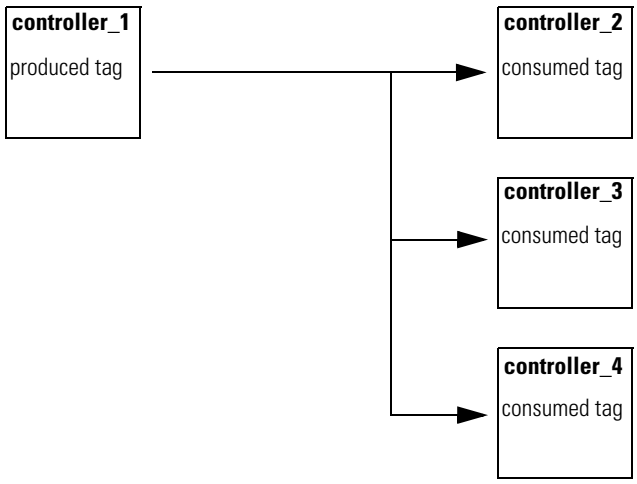
For this information	See
Produce and Consume (Interlock) Data	45
Send and Receive Messages	47
Connection Overview	49
Calculate Connection Use	50
Connections Example	52

Produce and Consume (Interlock) Data

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*,

The controller supports the ability to produce (broadcast) and consume (receive) system-shared tags over ControlNet or EtherNet/IP networks. Produced and consumed tags each require connections. Over ControlNet, produced and consumed tags are scheduled connections.



This type of tag	Description
produced	A produced tag allows other controllers to consume the tag, which means that a controller can receive the tag data from another controller. The producing controller uses one connection for the produced tag and one connection for each consumer. The controller's communication device uses one connection for each consumer. As you increase the number of controllers that can consume a produced tag, you also reduce the number of connections the controller and communication device have available for other operations, like communications and I/O.
consumed	Each consumed tag requires one connection for the controller that is consuming the tag. The controller's communication device uses one connection for each consumer.

For two controllers to share produced or consumed tags, both controllers must be attached to the same control network (such as a ControlNet or Ethernet/IP network). You cannot bridge produced and consumed tags over two networks.

The total number of tags that can be produced or consumed is limited by the number of available connections. If the controller uses all of its connections for I/O and communication devices, no connections are left for produced and consumed tags.

For more information...

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- produce a tag
- consume a tag
- produce a large array

The *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094 provides guidelines on how to:

- create produced and consumed tags
- specify an RPI
- manage connections

Send and Receive Messages

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*,

Messages transfer data to other devices, such as other controllers or operator interfaces. Messages use unscheduled connections to send or receive data. Connected messages can leave the connection open (cache) or close the connection when the message is done transmitting.

This message type:	With this communication method:	Is a connected message:	The message can be cached:
CIP data table read or write		3	3
PLC2, PLC3, PLC5, or SLC (all types)	CIP CIP with Source ID		
	DH+	3	3
CIP generic		your option (1)	3(2)
block-transfer read or write		3	3

(1) You can connect CIP generic messages. But for most applications we recommend you leave CIP generic messages unconnected.

(2) Consider caching only if the target module requires a connection.

Connected messages are unscheduled connections on both ControlNet and EtherNet/IP networks.

Each message uses one connection, regardless of how many devices are in the message path. You can programmatically change the target of a MSG instruction to optimize message transfer time.

Determine whether to cache message connections

When you configure a MSG instruction, you have the option of whether or not to cache the connection.

If the message executes	Then
repeatedly	Cache the connection. This keeps the connection open and optimizes execution time. Opening a connection each time the message executes increases execution time.
infrequently	Do not cache the connection. This closes the connection upon completion of the message, which frees up that connection for other uses.

The controller has the following limits on the number of connections that you can cache:

If you have this software and firmware revision:	Then you can cache:
11.x or earlier	• block transfer messages for up to 16 connections • other types of messages up to 16 connections
12.x or later	up to 32 total connections

For more information... The *Logix5000 Controllers General Instructions Reference Manual*, 1756-RM003 describes how to use the MSG instruction.

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- execute a MSG instruction
- get and set the number of unconnected buffers
- convert INT data to DINT data
- manage multiple MSG instructions
- send one MSG to multiple devices

Connection Overview

See:

- *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094

A Logix5000 system uses a connection to establish a communication link between two devices. Connections can be:

- controller to local I/O modules or local communication modules
- controller to remote I/O or remote communication modules
- controller to remote I/O (rack-optimized) modules
- produced and consumed tags
- messages
- controller access by RSLogix 5000 programming software
- controller access by RSLinx software for HMI or other applications

The limit of connections may ultimately reside in the communication module you use for the connection. If a message path routes through a communication module, the connection related to the message also counts towards the connection limit of that communication module.

This device	Supports this many connections
FlexLogix controller	100
1788-CNx communication card	32
1788-DNBO communication card	2
1788-ENBT communication card	32

Other controllers and communication modules support different maximum numbers of connections.

For more information... The *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094 describes how to optimize connection use.

Calculate Connection Use

To calculate the total number of local connections the controller uses:

Connection Type:	Device Quantity:	Connections per Device:	Total Connections:
rack-optimized connection for the local DIN rail and the extended-local DIN rail	2	1	2
I/O module (rack-optimized connection) on local rail		0	
I/O module (direct connection) on local rail		1	
I/O module (rack-optimized connection) on extended-local rail		0	
I/O module (direct connection) on extended-local rail		1	
1788-CNx ControlNet communication card		0	0
1788-DNBO DeviceNet communication card (direct connection)[1]		2	
1788-ENBT Ethernet/IP communication card		0	0
		total	

[1] FlexLogix controller connection to remote DeviceNet devices are accounted for in the 2 connections to the 1788-DNBO card.

Remote connections depend on the communication module. The number of connections the module itself supports determines how many connections the controller can access through that module. To calculate the total number of remote connections the controller uses:

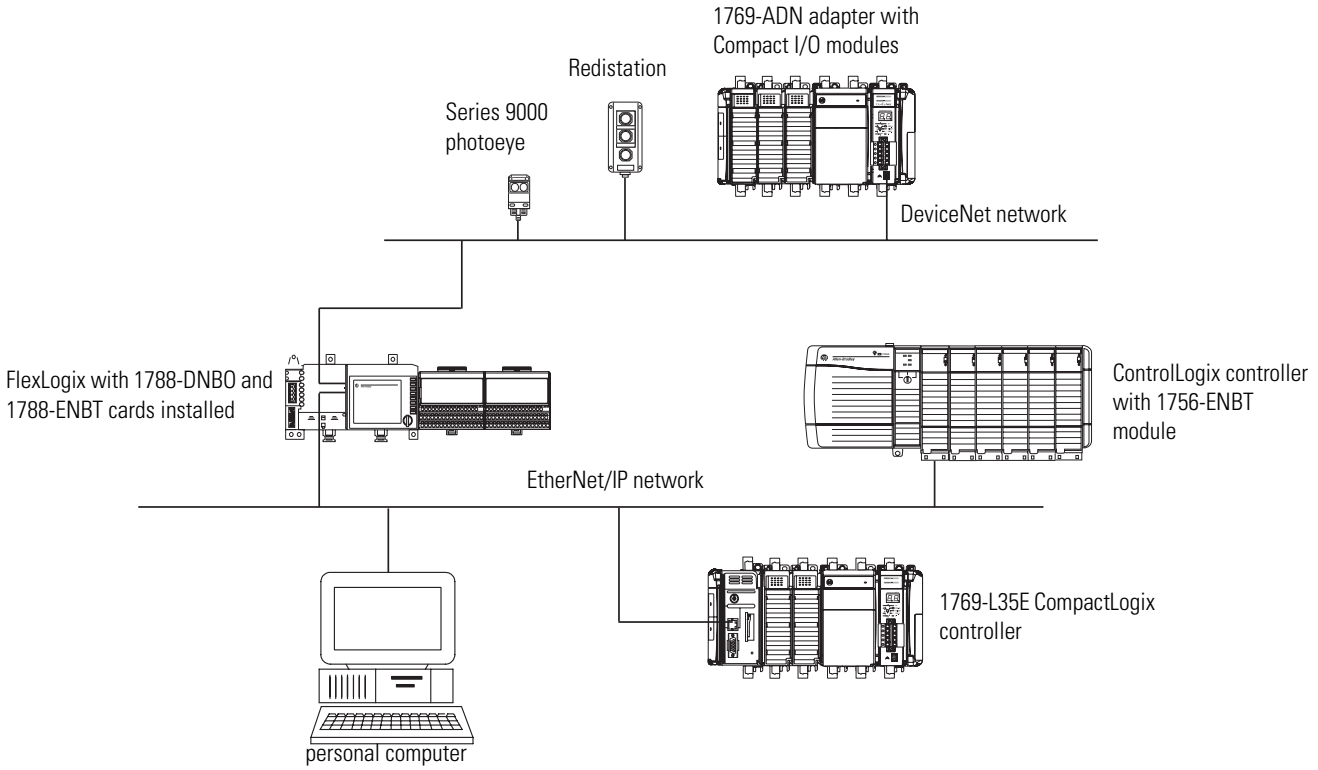
Remote Connection Type	Device Quantity	Connections per Device	Total Connections
remote ControlNet communication module			
I/O configured as direct connection (none)		0 **or**	
I/O configured as rack-optimized connection		1	
remote I/O module over ControlNet (direct connection)		1	
remote EtherNet/IP communication module			
I/O configured as direct connection (none)		0 **or**	
I/O configured as rack-optimized connection		1	
remote I/O module over EtherNet/IP (direct connection)		1	
remote device over DeviceNet			
(accounted for in rack-optimized connection for optional 1788-DNBO card)		0	
other remote communication adapter		1	
produced tag		1	
each consumer		1	
consumed tag		1	

Remote Connection Type	Device Quantity	Connections per Device	Total Connections
message (depending on type)		1	
block-transfer message		1	
		total	

Connections Example

In this example system the FlexLogix controller:

- controls local (in the same chassis) digital I/O modules
- controls remote I/O devices on DeviceNet
- sends and receives messages to/from a ControlLogix controller on EtherNet/IP
- produces one tag that the the CompactLogix controller consumes
- is programmed via RSLogix 5000 programming software



The FlexLogix controller in this example uses these connections:

Connection Type:	Device Quantity:	Connections per Device:	Total Connections:
controller to local I/O modules (rack-optimized)	2	1	2
controller to installed DeviceNet communication card	1	2	2
controller to installed EtherNet/IP communication card	1	0	0
controller to RSLogix 5000 programming software	1	1	1
message to ControlLogix controller	2	1	2
produced tag consumed by CompactLogix controller	2	1	2
		total	9

Place, Configure, and Monitor I/O

Use This Chapter

For this information:	See:
Select I/O Modules	53
Place Local I/O Modules	54
Configure I/O	55
Configure Distributed I/O on EtherNet/IP	59
Configure Distributed I/O on ControlNet	60
Configure Distributed I/O on DeviceNet	61
Address I/O Data	62
Determine When Data Is Updated	63
Reconfigure an I/O Module	66

Select I/O Modules

See:

- *FLEX I/O and FLEX EX Selection Guide,* 1794-SG002

When selecting 1794 FLEX I/O modules, select:

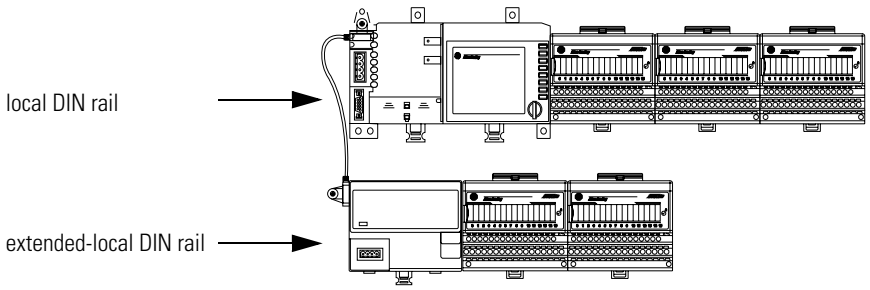
- Select a communication adapter - Choose the network for your operating system
- Select I/O modules based on field devices
- Select a terminal base - Choose an appropriate terminal base for your modules
- Select power supplies and make sure there is sufficient power for the communication adapter and modules

Place Local I/O Modules

See:

- *FLEX I/O Analog Modules User Manual*, 1794-6.5.2
- *FLEX I/O Digital Modules User*

The FlexLogix controller supports a local DIN rail of as many as 8 I/O modules and an extended-local DIN rail of as many as 8 I/O modules. The second DIN rail is optional.



Selecting a Power Supply

In a FlexLogix system, select an Allen-Bradley power supply. In applications that must be compliant with CSA requirements, use a Separated Extra-Low Voltage (SELV) power supply that is compliant with IEC 61010.1, Annex H.

When selecting power supplies:

- Provide power for the controller separately from the power for the FLEX I/O modules. To provide power for FLEX I/O modules, follow the guidelines in the documentation for those modules.
- When providing power for the 1794-FLA extended-local I/O adapter, treat the adapter as a communication adapter, not as an I/O module.

1794 FLEX power supplies

The following power supplies available for use with the FlexLogix system.

Catalog number	Nominal input voltage	Input voltage range	Maximum real input power	Maximum apparent input power	Maximum transformer load	Output current
1794-PS3	120/230V ac	85-265V ac	86W	205VA	250VA	3.0A @ 24V dc (horizontal mount) 2.8A @ 24Vdc (non-horizontal mount)
1794-PS13			36W	53VA	90VA	1.3A @ 24V dc

The FlexLogix controller also supports distributed (remote) I/O via these networks:

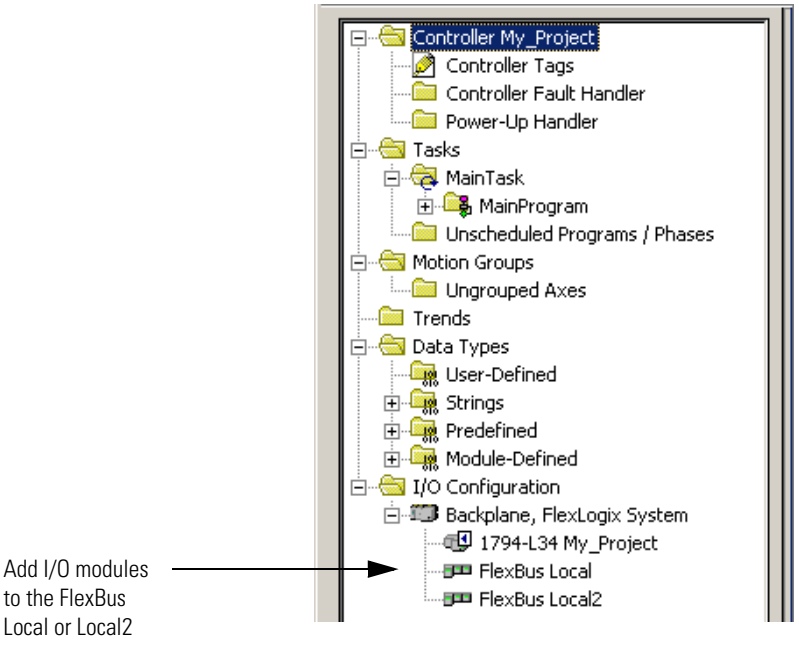
- EtherNet/IP
- ControlNet
- DeviceNet

Configure I/O

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*, 1756-RM094

To communicate with an I/O module in your system, you add the module to the I/O Configuration folder of the controller.



When you add a module, you also define a specific configuration for the module. While the configuration options vary from module to module, there are some common options that you typically configure:

Configuration Option:	Description:
requested packet interval (RPI)	The RPI specifies the period at which data updates over a connection. For example, an input module sends data to a controller at the RPI that you assign to the module. • Typically, you configure an RPI in milliseconds (ms). The range is 0.2 ms (200 microseconds) to 750 ms. • If a ControlNet network connects the devices, the RPI reserves a slot in the stream of data flowing across the ControlNet network. The timing of this slot may not coincide with the exact value of the RPI, but the control system guarantees that the data transfers at least as often as the RPI.
change of state (COS)	Digital I/O modules use change of state (COS) to determine when to send data to the controller. If a COS does not occur within the RPI timeframe, the module multicasts data at the rate specified by the RPI. Because the RPI and COS functions are asynchronous to the logic scan, it is possible for an input to change state during program scan execution. If this is a concern, buffer input data so your logic has a stable copy of data during its scan. Use the Synchronous Copy (CPS) instruction to copy the input data from your input tags to another structure and use the data from that structure.
communication format	Many I/O modules support different formats. The communication format that you choose also determines: • data structure of tags • connections • network usage • ownership • whether the module returns diagnostic information
electronic keying	When you configure a module, you specify the slot number for the module. However, it is possible to place a different module in that slot, either on purpose or accidently. Electronic keying lets you protect your system against the accidental placement of the wrong module in a slot. The keying option you choose determines how closely any module in a slot must match the configuration for that slot before the controller opens a connection to the module. There are different keying options depending on your application needs.

I/O connections

A Logix5000 system uses connections to transmit I/O data. A connection can be:

Connection:	Description:
direct	A direct connection is a real-time, data transfer link between the controller and an I/O module. The controller maintains and monitors the connection between the controller and the I/O module. Any break in the connection, such as a module fault or the removal of a module while under power, causes the controller to set fault status bits in the data area associated with the module. Typically, analog I/O modules, diagnostic I/O modules, and specialty modules require direct connections.
rack-optimized	For digital I/O modules, you can select rack-optimized communication. A rack-optimized connection consolidates connection usage between the controller and all the digital I/O modules on a rack (or DIN rail). Rather than having individual, direct connections for each I/O module, there is one connection for the entire rack (or DIN rail).

Connections for local and extended-local I/O modules

The FlexLogix controller automatically assigns one rack-optimized connection for the local DIN rail and one rack-optimized connection for the extended-local DIN rail. You then configure each I/O module on a DIN rail to either use that rack-optimized connection or to use a direct connection. The rack-optimized connection for each DIN rail exists whether or not you configure the I/O modules to use that rack-optimized connection.

The rack-optimized connection lets you organize all the digital I/O modules on one DIN rail into one connection to the controller. Or you can choose to configure each I/O module to have a direct connection to the controller. Analog I/O modules must have a direct connection to the controller.

It is not as critical to manage the number of connections for local and extended-local I/O modules as it is for remote devices because the controller supports a direct connection for each possible local and extended-local I/O device.

Connections for remote devices

To optimize the number of available connections, place remote, digital I/O in the same location and use a rack-optimized connection to the remote adapter that connects the remote I/O to the FlexLogix system.

If you have remote analog I/O modules, or want a direct connection to specific remote I/O modules, you do not have to create the rack-optimized connection to the remote adapter. To use direct connections to remote I/O, select "none" for the communication format of the remote communication device.

IMPORTANT It is vital that you manage your connections to remote devices because, while the FlexLogix controller allows up to 100 total connections, the communications cards that connect to remote devices are limited to far fewer connections (i.e., 32 connections for ControlNet or EtherNet/IP).

For more information...

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- configure I/O
- address I/O data
- buffer I/O data

The *Logix5000 Controllers Design Guidelines Reference Manual*, 1756-RM094 provides guidelines on how to:

- buffer I/O
- specify an RPI rate
- select a communication format
- manage I/O connections

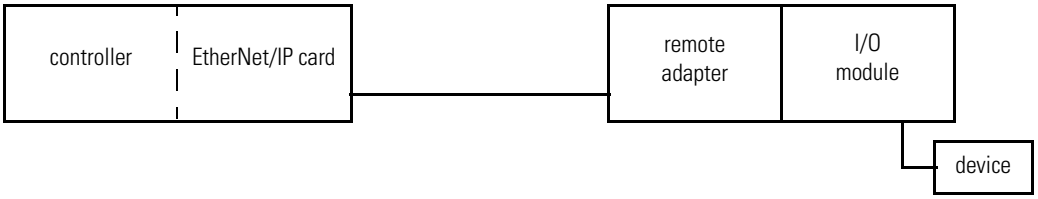
Configure Distributed I/O on EtherNet/IP

To communicate with distributed I/O modules over EtherNet/IP, you:

- install a 1788-ENBT communication card in your FlexLogix controller and add the card to the I/O configuration folder
- add an EtherNet/IP adapter, and I/O modules to the I/O Configuration folder of the controller.

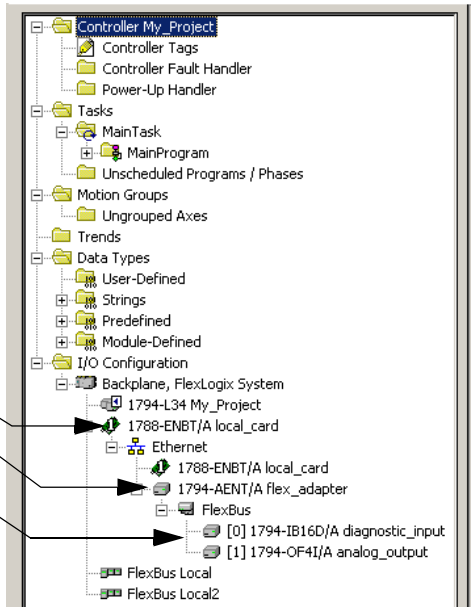
  Within the I/O Configuration folder, you organize the modules into a hierarchy (tree/branch, parent/child).

For a typical distributed I/O network…



…you build the I/O configuration in this order

1. Add the local communication card
2. Add the remote adapter for the distributed I/O chassis or DIN rail.



For more information... See *EtherNet/IP Communication Modules in Logix5000 Control Systems User Manual,* ENET-UM001.

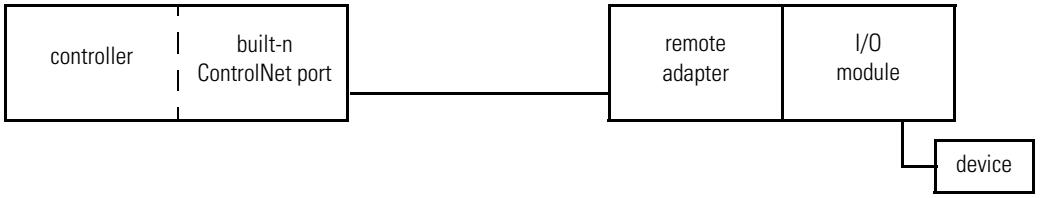
Configure Distributed I/O on ControlNet

To communicate with distributed I/O modules over ControlNet, you:

- install a 1788-CNx communication card in your FlexLogix controller and add the card to the I/O configuration folder
- add a ControlNet adapter, and I/O modules to the I/O Configuration folder of the controller.

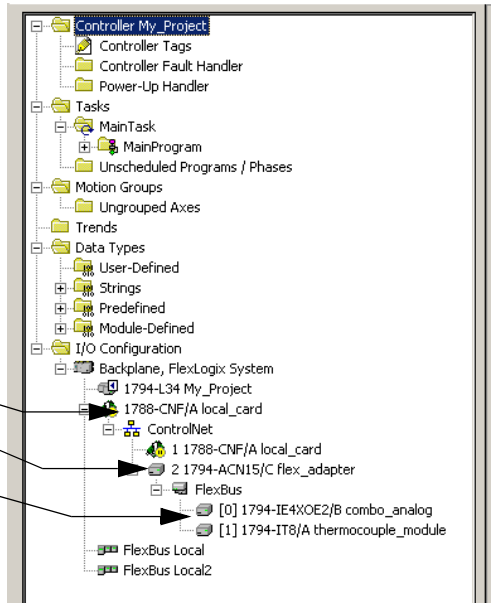
  Within the I/O Configuration folder, you organize the modules into a hierarchy (tree/branch, parent/child).

For a typical distributed I/O network…



…you build the I/O configuration in this order

1. Add the local communication card
2. Add the remote adapter for the distributed I/O chassis or DIN rail.



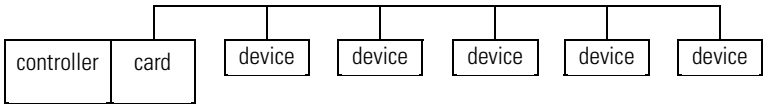
For more information... See *ControlNet Communication Modules in Logix5000 Control Systems User Manual,* CNET-UM001.

Configure Distributed I/O on DeviceNet

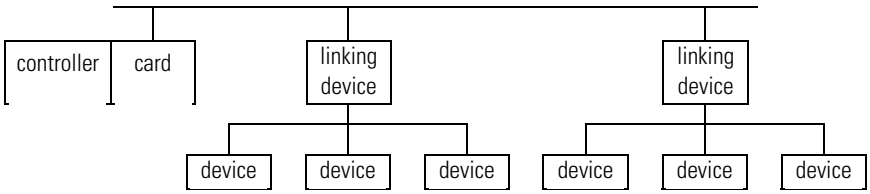
To communicate with the I/O modules over Device, you add the DeviceNet bridge to the I/O Configuration folder of the controller. You define a scanlist within the DeviceNet adapter to communicate data between devices and the controller.

For a typical distributed I/O network…

single network

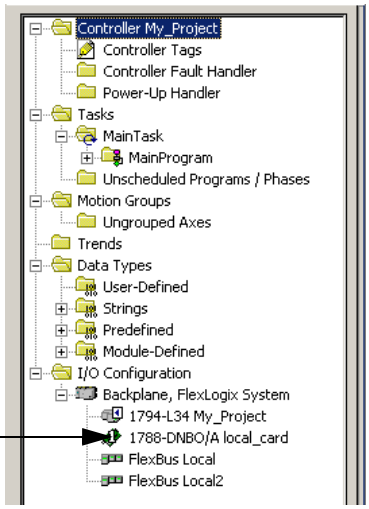


several smaller distributed networks (subnets)



…you build the I/O configuration in this order

Add the local scanner module.



For more information... See *DeviceNet Communication Modules in Logix5000 Control Systems User Manual,* DNET-UM004.

Address I/O Data

I/O information is presented as a set of tags.

- Each tag uses a structure of data. The structure depends on the specific features of the I/O module.
- The name of the tags is based on the location of the I/O module in the system.

An I/O address follows this format:

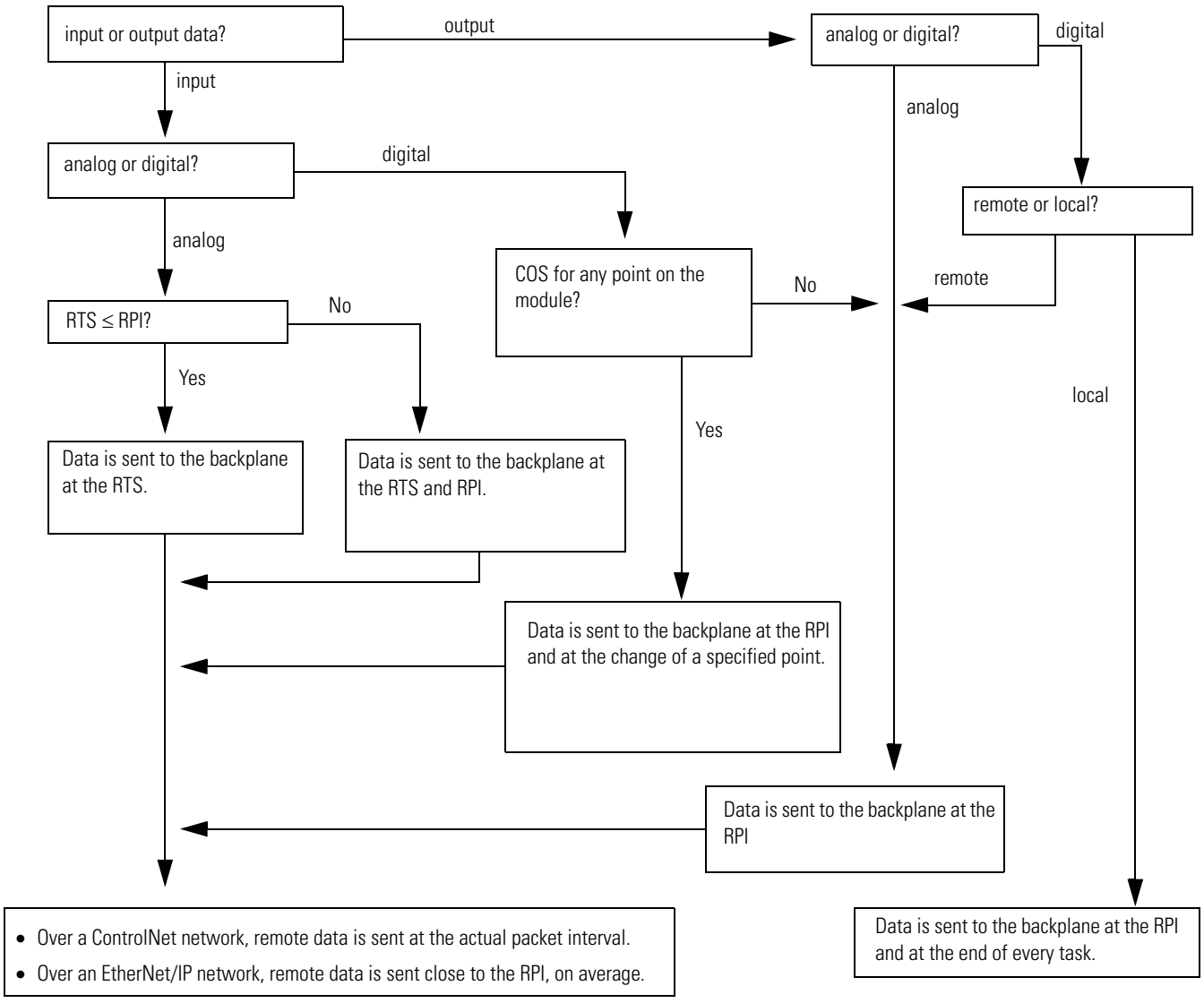
`Location` `:Slot` `:Type` `.Member` `.SubMember` `.Bit`

= Optional

Where:	Is:
`Location`	Network location LOCAL = same chassis or DIN rail as the controller *ADAPTER_NAME* = identifies remote communication adapter or bridge module
`Slot`	Slot number of I/O module in its chassis or DIN rail
`Type`	Type of data I = input O = output C = configuration S = status
`Member`	Specific data from the I/O module; depends on what type of data the module can store. • For a digital module, a Data member usually stores the input or output bit values. • For an analog module, a Channel member (CH#) usually stores the data for a channel.
`SubMember`	Specific data related to a Member.
`Bit`	Specific point on a digital I/O module; depends on the size of the I/O module (0-31 for a 32-point module)

Determine When Data Is Updated

FlexLogix controllers update date asynchronous with the execution of logic. Use the following flowchart to determine when a producer (controller, input module, or bridge module) will send data.



TIP If you need to ensure that the I/O values being used during logic execution are from one moment in time (such as at the beginning of a ladder program), use the Synchronous Copy instruction (CPS) to buffer I/O data.

For more information... See *Logix5000 Controllers Common Procedures Programming Manual*, publication number 1756-PM001 for examples of I/O buffering or to the *Logix5000 Controllers General Instruction Set Reference Manual,* publication number 1756-RM003 for information on the CPS instruction.

Monitor I/O Modules

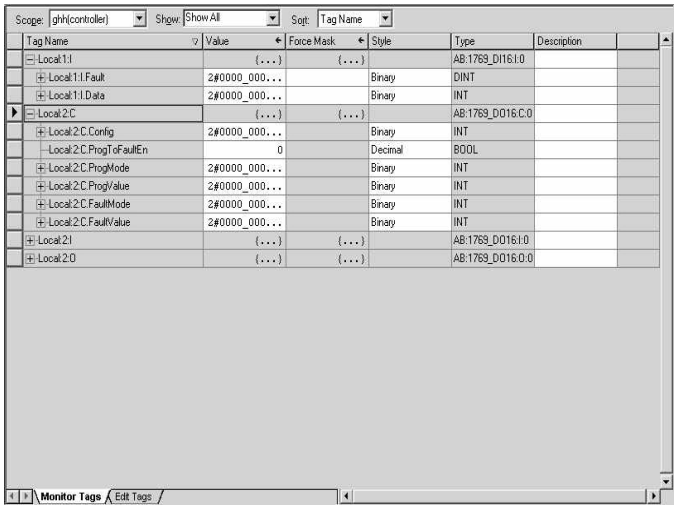
The FlexLogix controller offers different levels at which you can monitor I/O modules. You can:

- configure an I/O module so that the controller faults if that I/O module loses its connection with the controller
- use the programming software to display fault data (Refer to Displaying fault data on page 5-64)
- program logic to monitor fault data so you can take appropriate action (Refer to *Logix5000 Controllers Common Procedures Programming Manual*, publication number 1756-PM001, for examples.)

Displaying fault data

Fault data for certain types of module faults can be viewed through the programming software.

To view this data, select Controller Tags in the Controller Organizer. Right-click to select Monitor Tags.



The display for the fault data defaults to decimal. Change it to Hex to read the fault code.

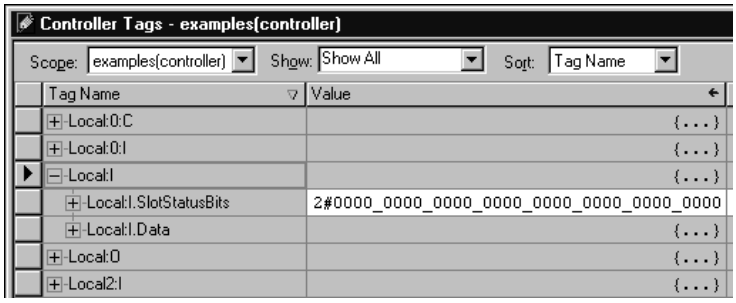
Monitor a rack-optimized connection

The controller views the DIN rail as another module in the system. Each DIN rail has its own data. To view this data through the programming software:

1. In the Controller Organizer, select Controller Tags. Right-click to display the Data Monitor.



2. Expand the data display as necessary.

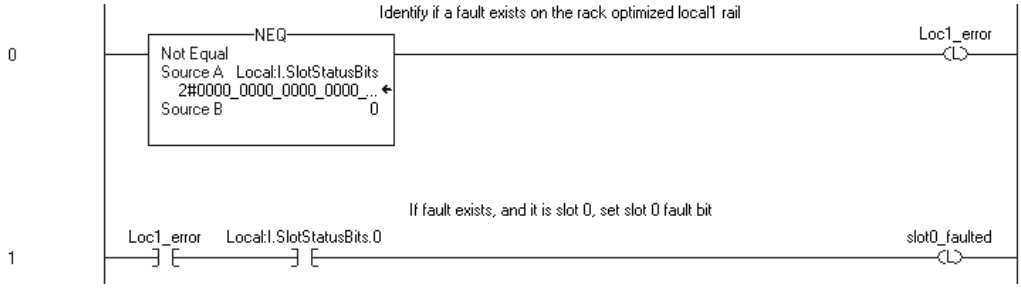


ATTENTION



If you have an extended-local DIN rail (LOCAL2) or a split rail, the modules after the 1794-CE1, -CE3 cable will fault if the cable is disconnected. In this case, all outputs are reset, regardless of the module configurations.

You can write logic to monitor the rack bits and take appropriate action if a fault occurs. For example, the following logic determines whether an error occurs on the Local rail. Then, the logic determines whether the error occurred at the module in slot 0. You can continue this logic to check each module on the rail.



Reconfigure an I/O Module

If an I/O module support reconfiguration, you can reconfigure the module via:

- Module Properties dialog in RSLogix 5000 software
- MSG instruction in program logic

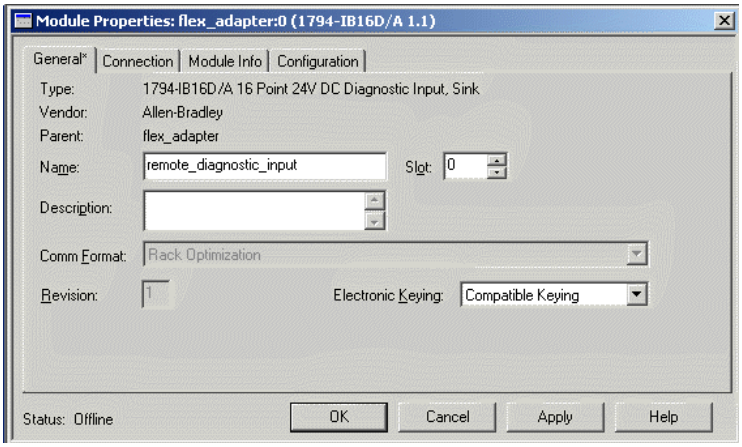
WARNING



Use care when changing the configuration of an I/O module. You could inadvertently cause the I/O module to operate incorrectly.

Reconfigure a module via RSLogix 5000 software

To change the configuration of an I/O module via RSLogix 5000 software, highlight the module in the I/O Configuration tree. Right-click and select Properties.



Reconfigure a module via a MSG instruction

To change the configuration of an I/O module programmatically, use a MSG instruction of type Module Reconfigure to send new configuration information to an I/O module. During the reconfiguration:

- Input modules continue to send input data to the controller.
- Output modules continue to controller their output devices.

A Module Reconfigure message requires the following configuration properties:

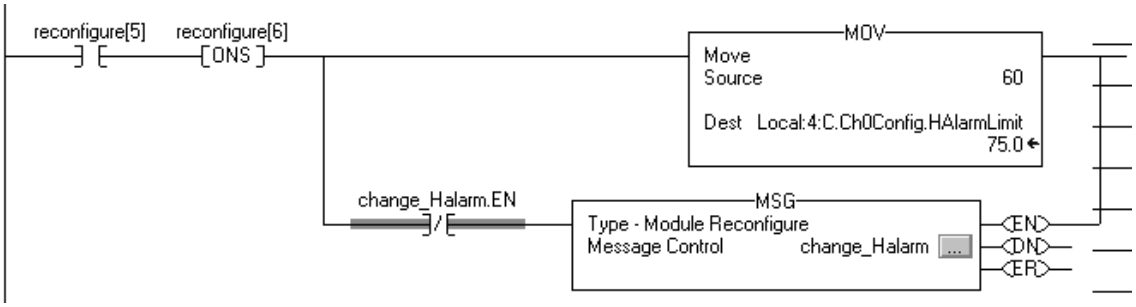
In this property:	Select:
Message Type	Module Reconfigure

To reconfigure an I/O module:

1. Set the required member of the configuration tag of the module to the new value.
2. Send a Module Reconfigure message to the module.

EXAMPLE Reconfigure an I/O module

When *reconfigure[5]* is on, the MOV instruction sets the high alarm to 60 for the local module in slot 4. The Module Reconfigure message then sends the new alarm value to the module. The ONS instruction prevents the rung from sending multiple messages to the module while the *reconfigure[5]* is on.



Notes:

Develop Applications

Use This Chapter

For this information:	See:
Manage Tasks	69
Develop Programs	70
Organize Tags	75
Select a Programming Language	76
Monitor Controller Status	79
Monitor Connections	80
Select a System Overhead Percentage	82
Use the Event Task	85

Manage Tasks

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*,

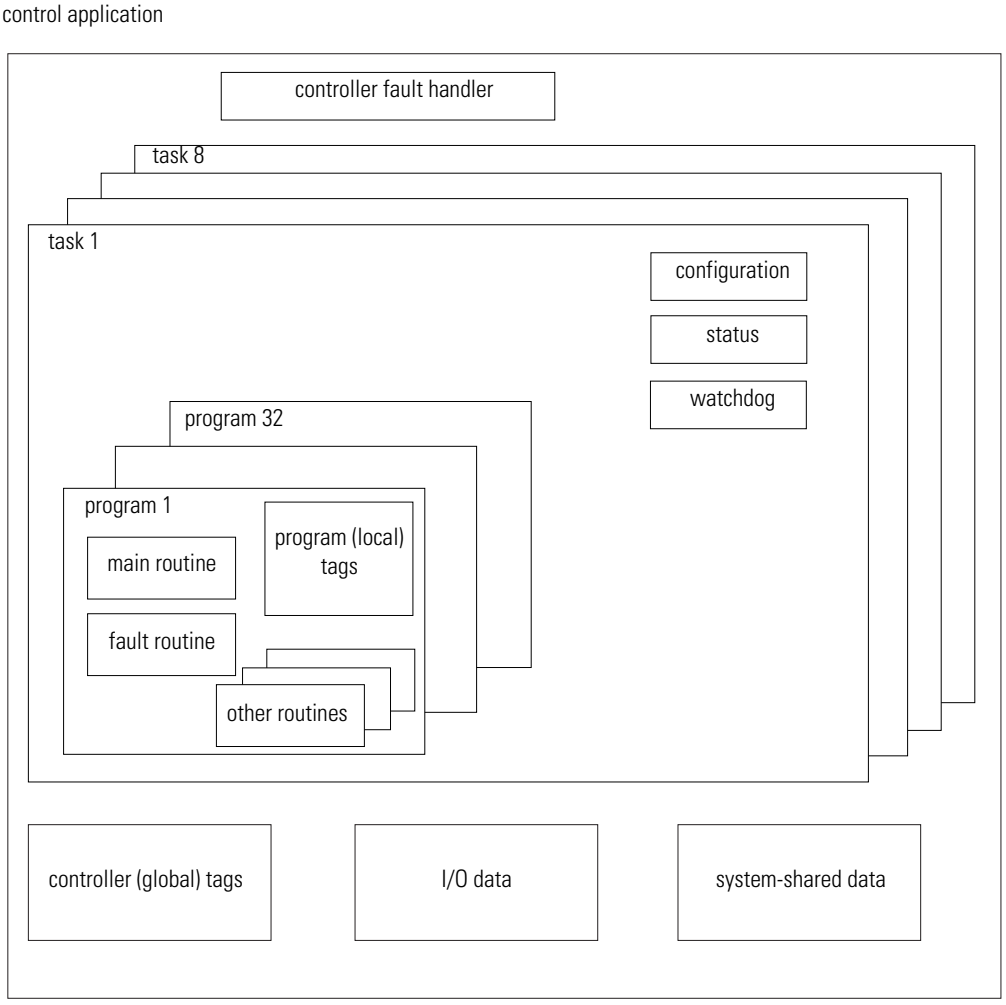
A Logix5000 controller lets you use multiple tasks to schedule and prioritize the execution of your programs based on specific criteria. This balances the processing time of the controller among the different operations in your application.

- The controller executes only one task at one time.
- A different task can interrupt a task that is executing and take control.
- In any given task, only one program executes at one time.

Develop Programs

The controller operating system is a preemptive multitasking system that is IEC 1131-3 compliant. This environment provides:

- tasks to configure controller execution
- programs to group data and logic
- routines to encapsulate executable code written in a single programming language



Defining tasks

A task provides scheduling and priority information for a set of one or more programs. You can configure tasks as continuous, periodic, or event. Only one task can be continuous.

A task can have as many as 100 separate programs, each with its own executable routines and program-scoped tags. Once a task is triggered (activated), all the programs assigned to the task execute in the order in which they are grouped. Programs can only appear once in the Controller Organizer and cannot be shared by multiple tasks.

Specifying task priorities

Each task in the controller has a priority level. The operating system uses the priority level to determine which task to execute when multiple tasks are triggered. You can configure periodic tasks to execute from the lowest priority of 15 up to the highest priority of 1. A higher priority task will interrupt any lower priority task. The continuous task has the lowest priority and is always interrupted by a periodic task.

The FlexLogix controller uses a dedicated periodic task at priority 6 to process I/O data. This periodic task executes at the RPI you configure for the FlexBus, which can be as fast as once every 2 ms. Its total execution time is as long as it takes to scan the configured I/O modules.

How you configure your tasks affects how the controller receives I/O data. Tasks at priorities 1 to 5 take precedence over the dedicated I/O task. Tasks in this priority range can impact I/O processing time.

For example, if you use the following configuration:

- I/O RPI = 2 ms
- a task of priority = 1 to 5 that requires 500 μs to execute and is scheduled to run every millisecond

this configuration leaves the dedicated I/O task 500 μs to complete its job of scanning the configured I/O.

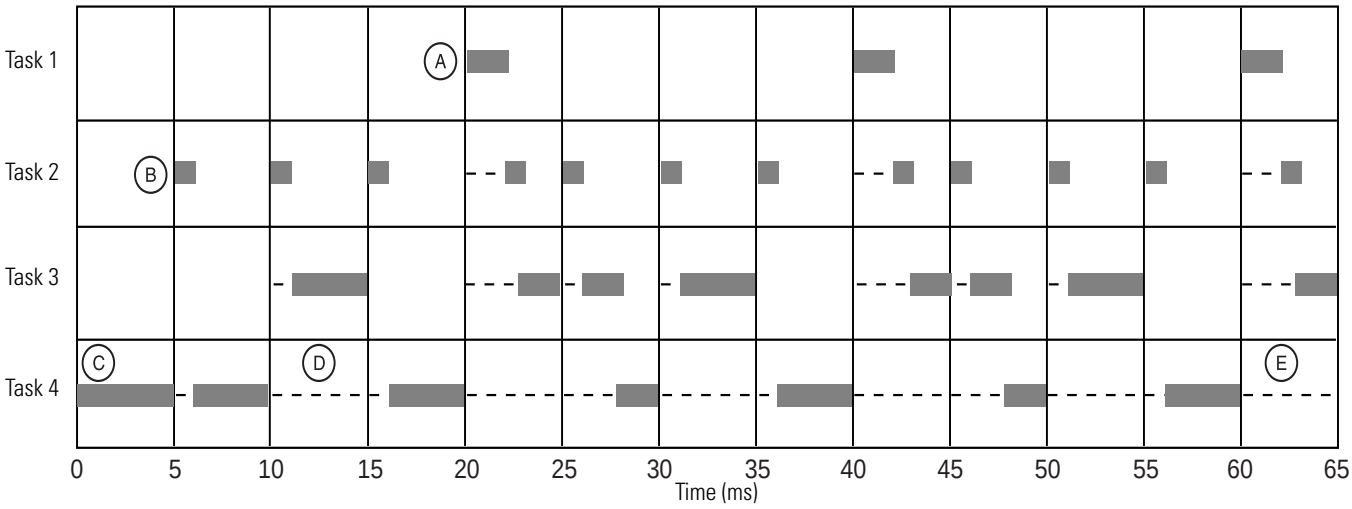
However, if you schedule two high priority tasks (1 to 5) to run every millisecond, and they both require 500 µs or more to execute, no CPU time would be left for the dedicated I/O task. Furthermore, if you have so much configured I/O that the execution time of the dedicated I/O task approaches 2 ms (or the combination of the high priority tasks and the dedicated I/O task approaches 2 ms) no CPU time is left for low priority tasks (7 to 15).

TIP

For example, if your program needs to react to inputs and control outputs at a deterministic rate, configure a periodic task with a priority higher than 5 (1 through 5). This keeps the dedicated I/O task from affecting the periodic rate of your program. However, if your program contains a lot of math and data manipulation, place this logic in a task with priority lower than 6 (7 through 15), such as the continuous task, so that the dedicated I/O task is not adversely affected by your program.

The following example shows the task execution order for an application with periodic tasks and a continuous task.

Task:	Priority Level:	Task Type:	Example Execution Time:	Worst Case Completion Time:
1	5	20 ms periodic task	2 ms	2 ms
2	6	dedicated I/O task 5 ms selected RPI	1 ms	3 ms
3	10	10 ms periodic task	4 ms	8 ms
4	none (lowest)	continuous task	25 ms	60 ms



Notes:

A. The highest priority task interrupts all lower priority tasks.

B. The dedicated I/O task can be interrupted by tasks with priority levels 1 to 7. The dedicated I/O task interrupts tasks with priority levels 7 to 15. This task runs at the selected RPI rate scheduled for the FlexLogix system (5 ms in this example).

C. The continuous task runs at the lowest priority and is interrupted by all other tasks.

D. A lower priority task can be interrupted multiple times by a higher priority task.

E. When the continuous task completes a full scan it restarts immediately, unless a higher priority task is running.

Defining programs

Each program contains program tags, a main executable routine, other routines, and an optional fault routine. Each task can schedule as many as 100 programs.

The scheduled programs within a task execute to completion from first to last. Programs that are not attached to any task show up as unscheduled programs. You must specify (schedule) a program within a task before the controller can scan the program.

Defining routines

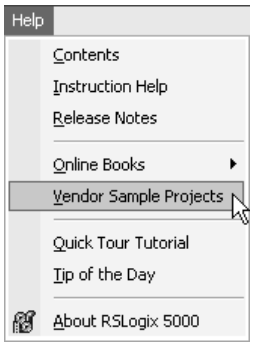
A routine is a set of logic instructions in a single programming language, such as ladder logic. Routines provide the executable code for the project in a controller. A routine is similar to a program file or subroutine in a PLC or SLC controller.

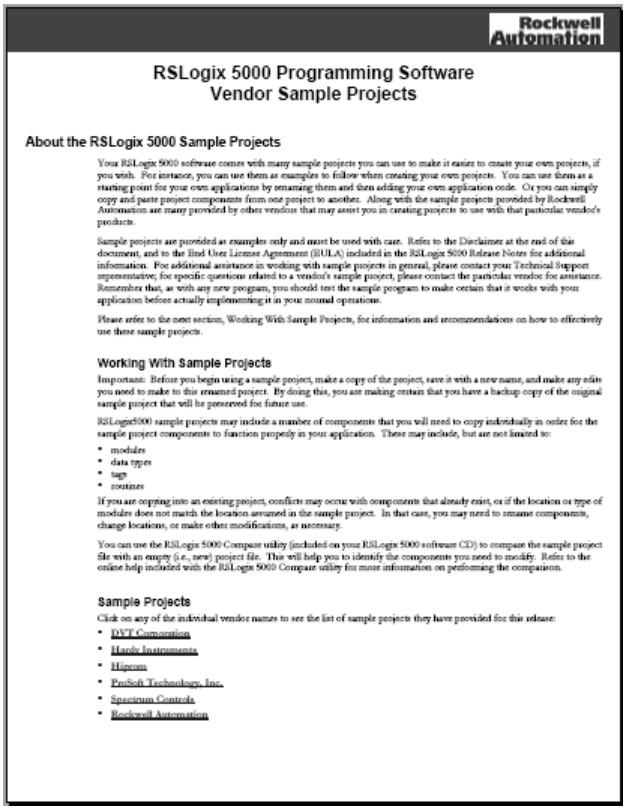
Each program has a main routine. This is the first routine to execute when the controller triggers the associated task and calls the associated program. Use logic, such as the Jump to Subroutine (JSR) instruction, to call other routines.

You can also specify an optional program fault routine. The controller executes this routine if it encounters an instruction-execution fault within any of the routines in the associated program.

Sample controller projects

RSLogix 5000 Enterprise programming software includes sample projects that you can copy and then modify to fit your application. From RSlogix 5000 software, select Help → Vendor Sample Projects to display a list of available, sample projects.





Scroll down to here and select the appropriate set of sample projects

For more information... The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- select which task to use
- configure tasks
- prioritize tasks
- inhibit tasks

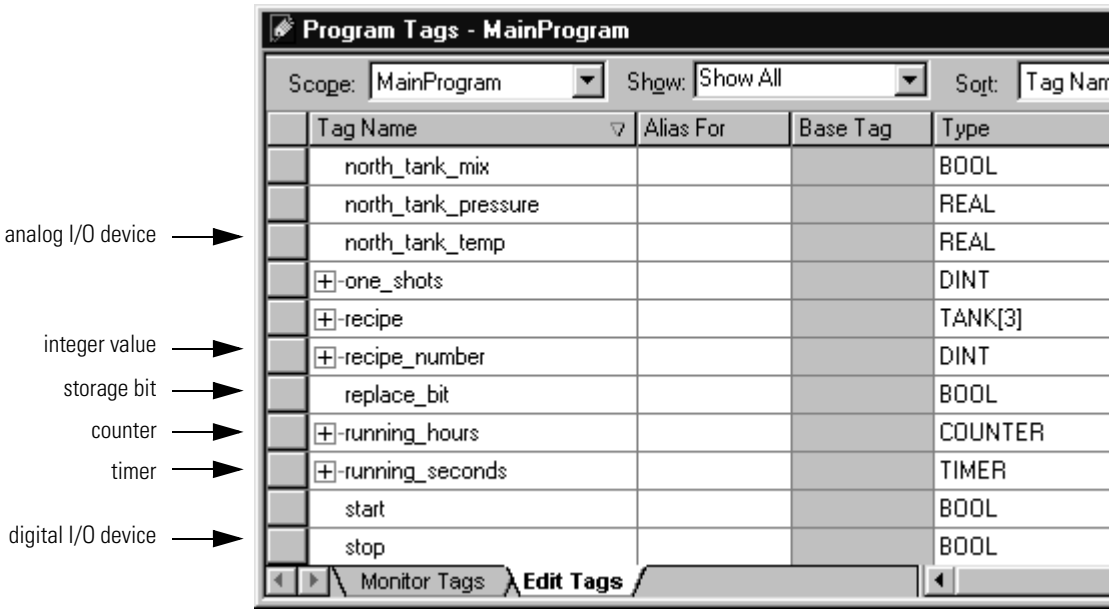
Organize Tags

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*,

With a Logix5000 controller, you use a tag (alphanumeric name) to address data (variables). In Logix5000 controllers, there is no fixed, numeric format. The tag name itself identifies the data. This lets you:

- organize your data to mirror your machinery
- document (through tag names) your application as you develop it



When you create a tag, you assign the following properties to the tag:

- tag type
- data type
- scope

For more information... The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- define tags
- create tags, arrays, and user-defined structures
- address tags
- create aliases to tags
- assign indirect addresses

Select a Programming Language

The FlexLogix controller supports these programming languages, both online and offline:

If you are programming:	Use this language:
continuous or parallel execution of multiple operations (not sequenced)	ladder diagram (LD)
boolean or bit-based operations	
complex logical operations	
message and communication processing	
machine interlocking	
operations that service or maintenance personnel may have to interpret in order to troubleshoot the machine or process	
continuous process and drive control	function block diagram (FBD)
loop control	
calculations in circuit flow	
high-level management of multiple operations	sequential function chart (SFC)
repetitive sequence of operations	
batch process	
motion control using structured text	
state machine operations	
complex mathematical operations	structured text (ST)
specialized array or table loop processing	
ASCII string handling or protocol processing	

Add-On Instructions

With version 16 of RSLogix 5000 programming software, you can design and configure sets of commonly used instructions to increase project consistency. Similar to the built-in instructions contained in Logix5000 controllers, these instructions you create are called Add-On Instructions. Add-On Instructions reuse common control algorithms. With them, you can:

- ease maintenance by animating logic for a single instance.
- protect intellectual property with locking instructions.
- reduce documentation development time.

You can use Add-On Instructions across multiple projects. You can define your instructions, obtain them from somebody else, or copy them from another project.

Once defined in a project, Add-On Instructions behave similarly to the built-in instructions in Logix5000 controllers. They appear on the instruction tool bar for easy access, as do internal RSLogix 5000 software instructions.

Save Time

With Add-On Instructions, you can combine your most commonly used logic into sets of reusable instructions. You save time when you create instructions for your projects and then share them with others. Add-On Instructions increase project consistency since commonly used algorithms all work in the same manner, regardless of who implements the project.

Use Standard Editors

You create Add-On Instructions by using one of three RSLogix 5000 software programming editors.

- Standard Ladder
- Function Block Diagram
- Structured Text

Once you have created instructions, you can use them in any RSLogix 5000 editor.

Export Add-On Instructions

You can export Add-On-Instructions to other projects as well as copy and paste them from one project to another. Give each instruction a

unique name so that you don't accidentally overwrite another instruction of the same name.

Use Context Views

Context views let you visualize an instruction's logic for a specific instant, simplifying online troubleshooting of your Add-On Instructions. Each instruction contains a revision, a change history, and an auto-generated help page.

Create Custom Help

When you create an instruction, you enter information for the description fields in software dialogs, information that becomes what is known as Custom Help. Custom Help makes it easier for users to get the help they need when implementing the instructions.

Apply Source Protection

As the creator of Add-On Instructions, you can limit users of your instruction(s) to read-only access, or you can bar access to the internal logic or local parameters used by the instruction(s). This source protection lets you prevent unwanted changes to your instruction(s) and protects your intellectual property.

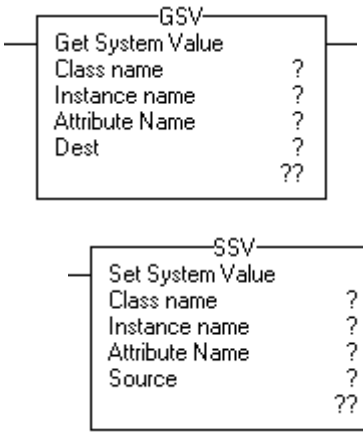
For more information...

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- design and program sequential function chart (SFC) logic
- program structured text (ST) logic
- program ladder diagram (LD) logic
- program function block diagram (FBD) logic
- force logic

The *Logix5000 Controllers Execution Time and Memory Use Reference Manual*, publication 1756-RM087 provides information on memory use and execution times for instructions.

Monitor Controller Status



The FlexLogix controller uses Get System Value (GSV) and Set System Value (SSV) instructions to get and set (change) controller data. The controller stores system data in objects. There is no status file, as in the PLC-5 processor.

The GSV instruction retrieves the specified information and places it in the destination. The SSV instruction sets the specified attribute with data from the source.

When you enter a GSV/SSV instruction, the programming software displays the valid object classes, object names, and attribute names for each instruction. For the GSV instruction, you can get values for all the available attributes. For the SSV instruction, the software displays only those attributes you are allowed to set.

In some cases, there will be more than one instance of the same type of object, so you might also have to specify the object name. For example, there can be several tasks in your application. Each task has its own TASK object that you access by the task name.

You can access these object classes:

- AXIS
- CONTROLLER
- CONTROLLERDEVICE
- CST
- DF1
- FAULTLOG
- MESSAGE
- MODULE
- MOTIONGROUP
- PROGRAM
- ROUTINE
- SERIALPORT
- TASK
- WALLCLOCKTIME

For more information...

The *Logix5000 Controllers General Instructions Reference Manual*, 1756-RM003 describes how to use the GSV and SSV instructions. These instructions support several different attributes of information.

The *Logix5000 Controllers Common Procedures Manual*, 1756-PM001 provides information on how to:

- handle major faults
- handle minor faults
- determine controller memory use

Monitor Connections

See:

- *Logix5000 Controllers Common Procedures Manual*, 1756-PM001
- *Logix5000 Controllers Design Considerations Reference Manual*,

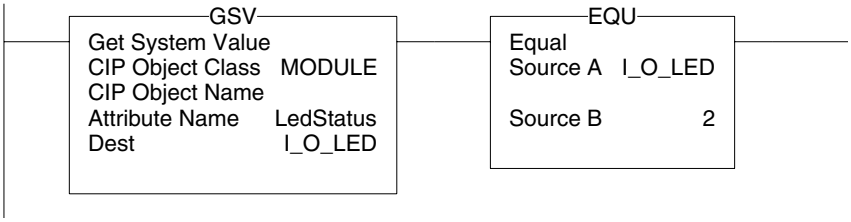
If communication with a device in the I/O configuration of the controller does not occur for 100 ms or 4 times the RPI (whichever is less), the communication times out and the controller produces the following warnings:

- The I/O LED on the front of the controller flashes green.
- A ⚠ shows over the I/O configuration folder and over the device (s) that has timed out.
- A module fault code is produced, which you can access through:
  - Module Properties dialog box for the module
  - GSV instruction

Determine if communication has timed out with any device

If communication times out with at least one device (module) in the I/O configuration of the controller, the I/O LED on the front of the controller flashes green.

- The GSV instruction gets the status of the I/O LED and stores it in the I_O_LED tag.
- If I_O_LED equal 2, the controller has lost communication with at least one device.



where:

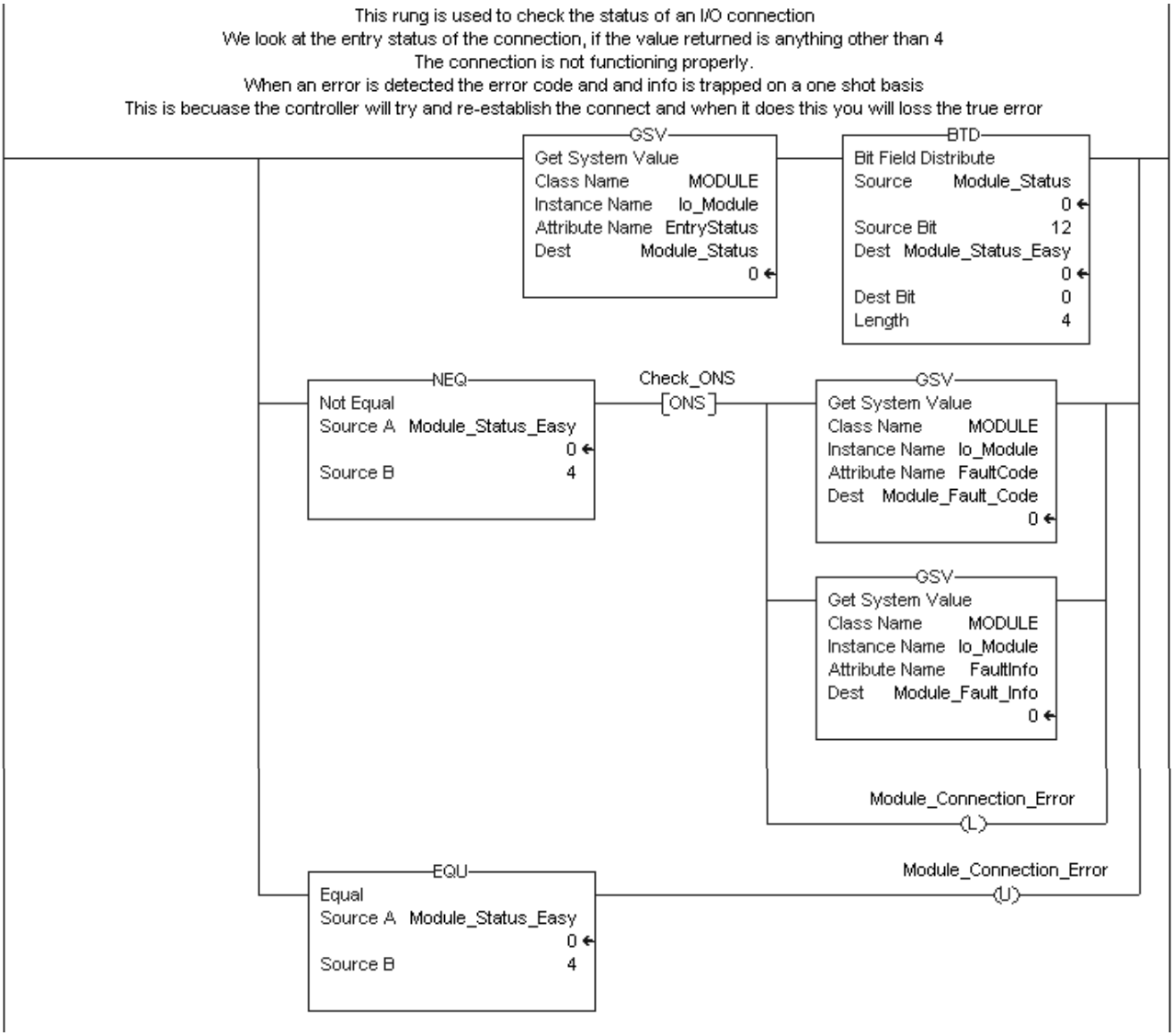
I_O_LED is a DINT tag that stores the status of the I/O LED on the front of the controller.

Determine if communication has timed out with a specific I/O module

If communication times out with a device (module) in the I/O configuration of the controller, the controller produces a fault code for the module.

- The GSV instruction gets the fault code for Io_Module and stores it in the Module_Status tag.

- If Module_Status is any value other than 4, the controller is not communicating with the module.



Interrupt the execution of logic and execute the fault handler

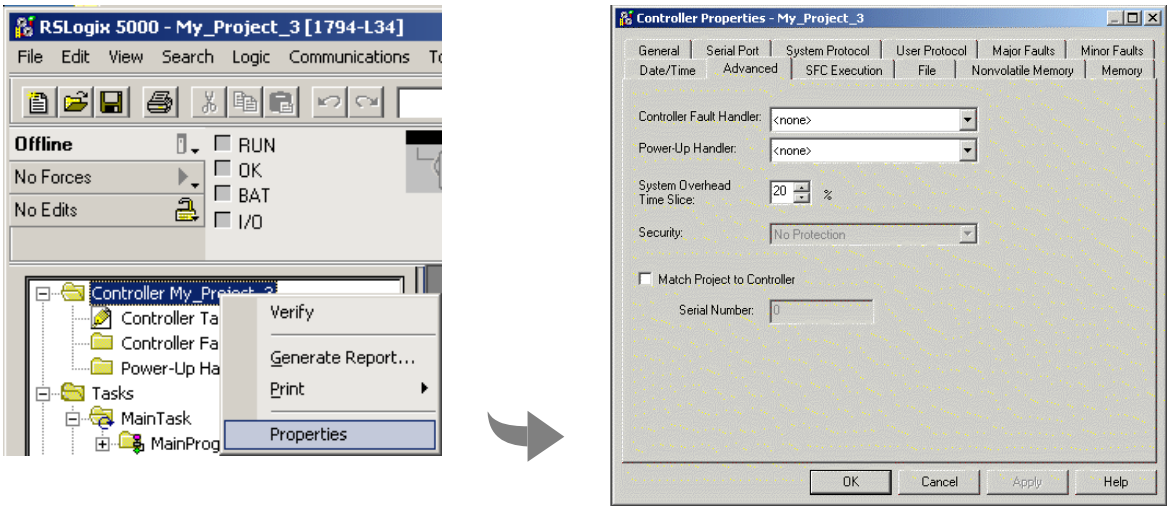
1. In the controller organizer, right-click the module and select *Properties*.
2. Click the Connection tab.
3. Select (check) the Major Fault If Connection Fails While in Run Mode check box.

4. Develop a routine for the Controller Fault Handler. See the *Logix5000 Controllers Common Procedures*, publication 1756-PM001.

Select a System Overhead Percentage

The Controller Properties dialog lets you specify a percentage for system overhead. This percentage specifies the percentage of controller time (excluding the time for periodic tasks) that is devoted to communication and background functions.

1. View properties for the controller and select the Advanced tab.



System overhead functions include:

- communicating with programming and HMI devices (such as RSLogix 5000 software)
- responding to messages
- sending messages
- re-establishing and monitoring I/O connections (such as RIUP conditions); this *does not* include normal I/O communications that occur during program execution
- bridging communications from the serial port of the controller to other communication devices

The controller performs system overhead functions for up to 1 ms at a time. If the controller completes the overhead functions in less than 1 ms, it resumes the continuous task.

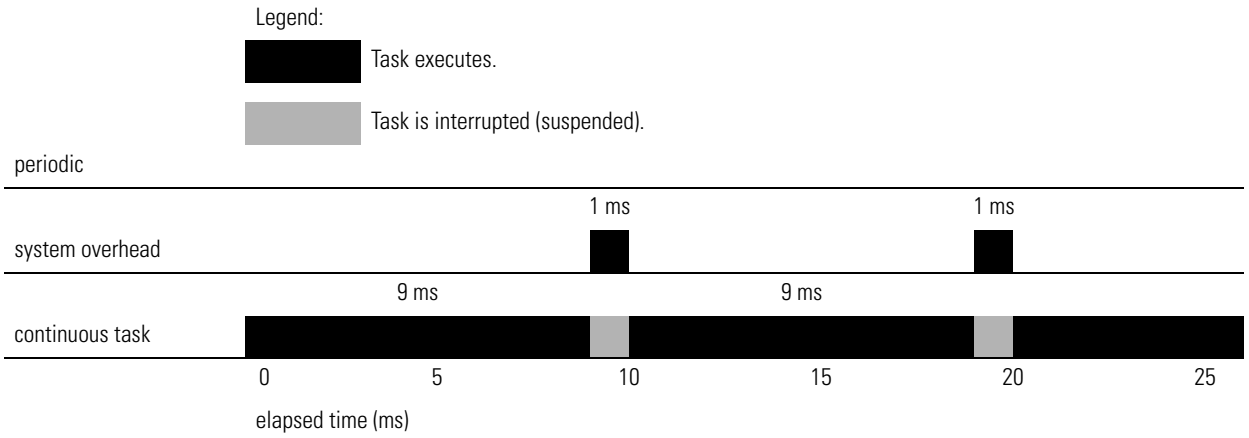
As the system overhead percentage increases, time allocated to executing the continuous task decreases. If there are no communications for the controller to manage, the controller uses the communications time to execute the continuous task. While increasing

the system overhead percentage does increase communications performance, it also increases the amount of time it takes to execute a continuous task - increasing overall scan time

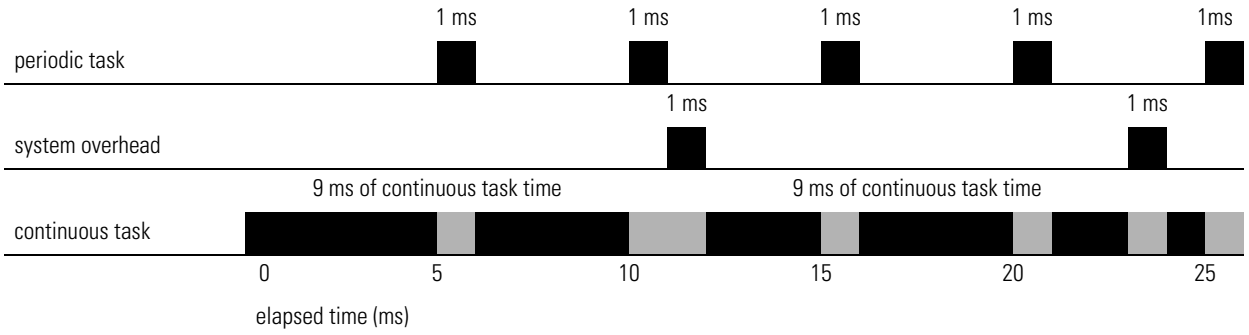
The table below shows the ratio between the continuous task and the system overhead functions:

At this time slice:	The continuous tasks runs for:	And then overhead occurs for up to:
10%	9 ms	1 ms
20%	4 ms	1 ms
33%	2 ms	1 ms
50%	1 ms	1 ms

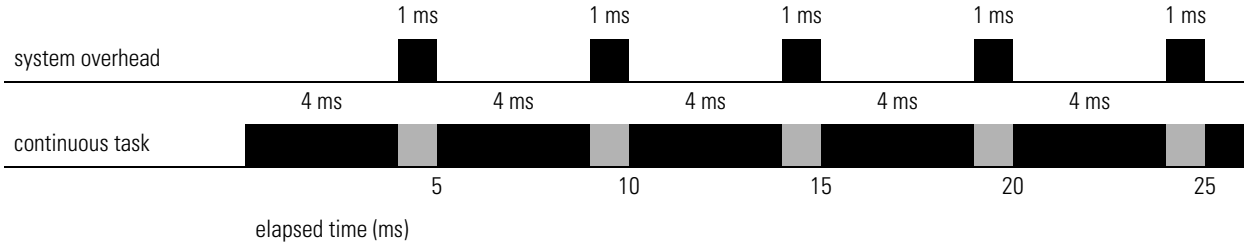
At a time slice of 10%, system overhead interrupts the continuous task every 9 ms (of continuous task time), as illustrated below.



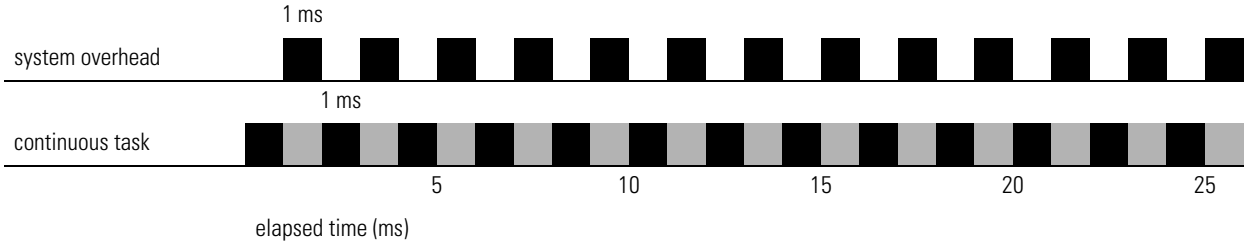
The interruption of a periodic task increases the elapsed time (clock time) between the execution of system overhead, as shown below.



If you use the default time slice of 20%, the system overhead interrupts the continuous task every 4 ms (of continuous task time).



If you increase the time slice to 50%, the system overhead interrupts the continuous task every 1 ms (of continuous task time).



If the controller only contains a periodic task(s), the system overhead timeslice value has no effect. System overhead runs whenever a periodic task is not running.



Use the Event Task

The event task is available with FlexLogix controllers using firmware version 12.x or greater. Previously, the only tasks available were the continuous task and periodic task. However, the event task offers FlexLogix controller users a task that executes a section of logic immediately when an event occurs.

An event task performs a function only when a specific event (trigger) occurs. Whenever the trigger for the event task occurs, the event task:

- interrupts any lower priority tasks
- executes one time
- returns control to where the previous task left off

For FlexLogix controller, the event task trigger can only be the EVENT instruction or a consume tag.

Prioritizing Periodic and Event Tasks

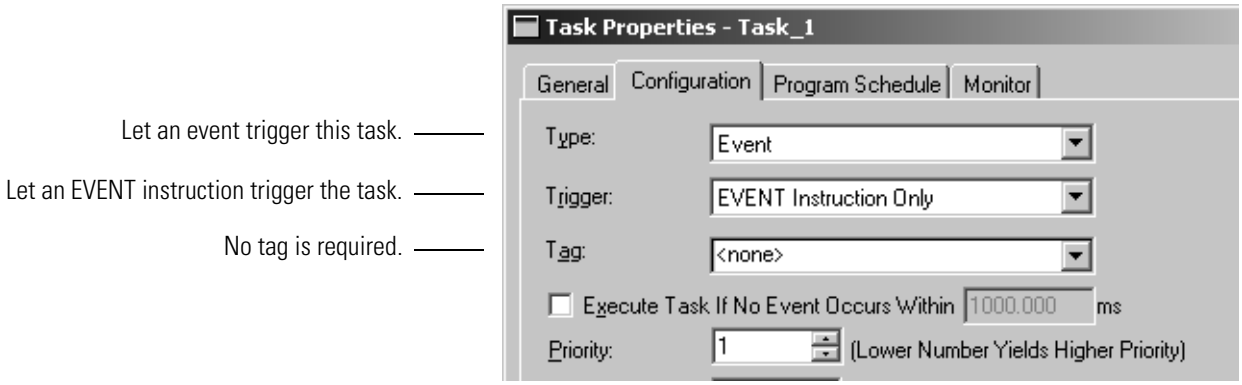
Although a FlexLogix project can contain up to 8 tasks, the controller executes only one task at a time. If a periodic or event task is triggered while another task is currently executing, the priority of each task tells the controller what to do.

The FlexLogix controller has 15 priority levels for its tasks. To assign a priority to a task, use the guidelines described in the table..

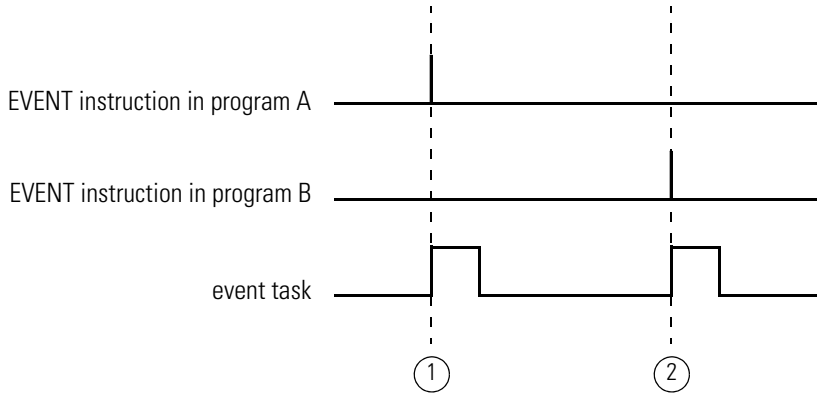
If you want:	Then	Notes:
this task to interrupt another task	Assign a priority number that is less than (higher priority) the priority number of the other task.	• A higher priority task interrupts all lower priority tasks. • A higher priority task can interrupt a lower priority task multiple times.
another task to interrupt this task	Assign a priority number that is greater than (lower priority) the priority number of the other task.	
this task to share controller time with another task	Assign the same priority number to both tasks.	The controller switches back and forth between each task and executes each one for 1ms.

Triggering the Event Task

To trigger an event task based on conditions in your logic, use the EVENT Instruction trigger.



The *EVENT Instruction Only* trigger requires that you use a Trigger Event Task (EVENT) instruction to trigger the task. You can use an EVENT instruction from multiple points in your project. Each time the instruction executes, it triggers the specified event task.



Description:	
1	Program A executes an EVENT instruction. The event task that is specified by the EVENT instruction executes one time.
2	Program B executes an EVENT instruction. The event task that is specified by the EVENT instruction executes one time.

Programmatically Determine if an EVENT Instruction Triggered a Task

To determine if an EVENT instruction triggered an event task, use a Get System Value (GSV) instruction to monitor the Status attribute of the task.

Status Attribute of the TASK Object

Attribute:	Data Type:	Instruction:	Description:	
Status	DINT	GSV SSV	Provides status information about the task. Once the controller sets a bit, you must manually clear the bit to determine if another fault of that type occurred.	
			To determine if:	**Examine this bit:**
			An EVENT instruction triggered the task (event task only).	*0*
			A timeout triggered the task (event task only).	1
			An overlap occurred for this task.	2

The controller does not clear the bits of the Status attribute once they are set.

- To use a bit for new status information, you must manually clear the bit.
- Use a Set System Value (SSV) instruction to set the attribute to a different value.

Checklist for an EVENT Instruction Task

For this:		Make sure you:
q	1. EVENT instruction	Use a Trigger Event Task (EVNT) instruction at each point in your logic that you want to trigger the event task.
q	2. Task priority	Configure the event task as the highest priority task. If a periodic task has a higher priority, the event task may have to wait until the periodic task is done.
q	3. Number of event tasks	Limit the number of event tasks. Each additional task reduces the processing time that is available for other tasks. This could cause an overlap.
q	4. Automatic Output Processing	For an event task, you can typically disable automatic output processing (default). This reduces the elapsed time of the task.

For more information on using the event task, see Logix5000 Controllers Common Procedures programming manual, publication 1756-PM001.

Notes:

Configure PhaseManager

Use This Chapter

See:
- PhaseManager User Manual, LOGIX-UM001

The PhaseManager option of RSLogix 5000 software gives you a state model for your equipment. This chapter summarizes:

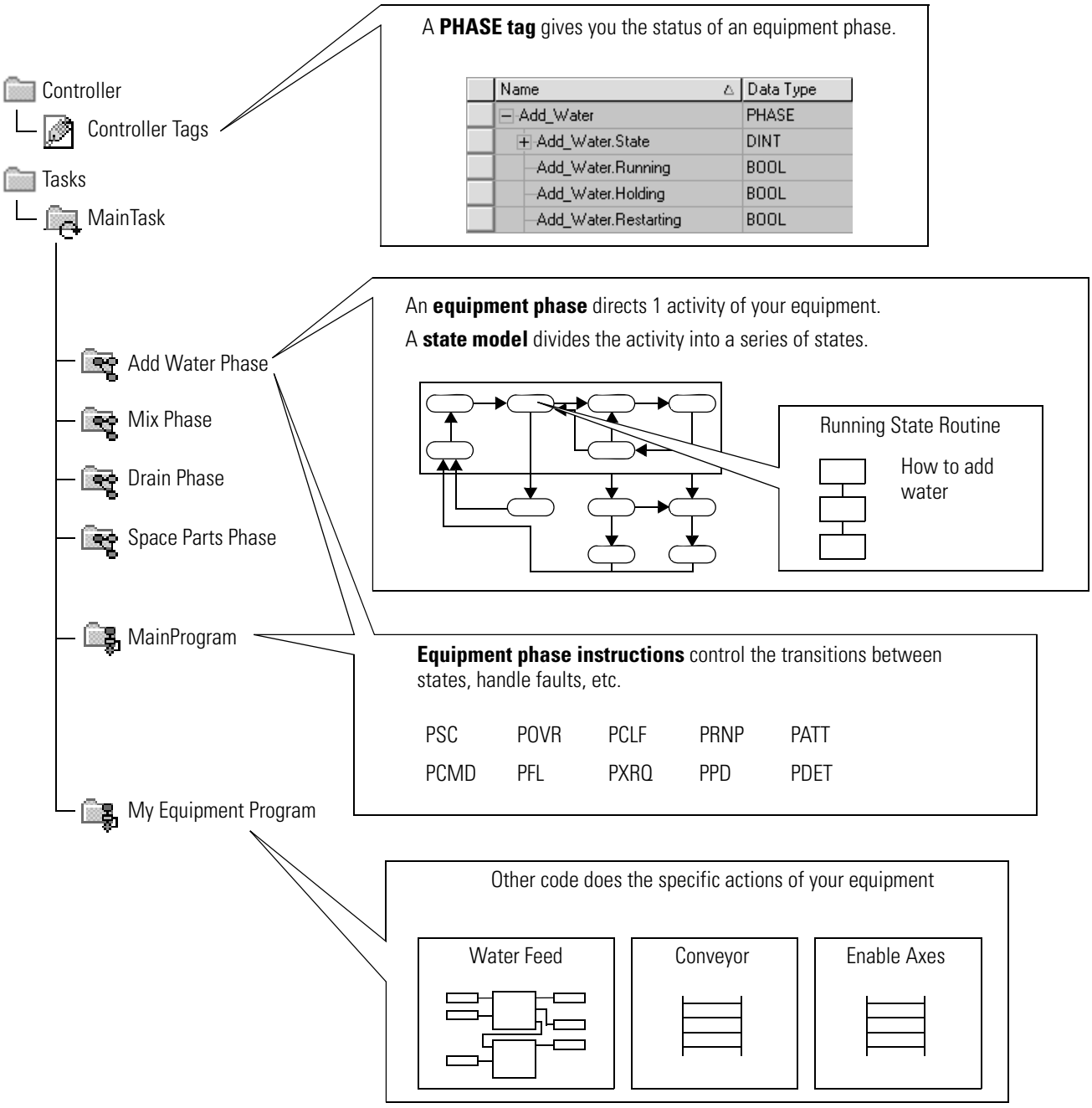
For this information:	See:
PhaseManager Overview	89
State Model Overview	91
Compare PhaseManager to Other State Models	94
Minimum System Requirements	94
Equipment Phase Instructions	95

PhaseManager Overview

PhaseManager lets you add equipment phases to your controller. An equipment phase helps you lay-out your code in sections that are easier to write, find, follow, and change.

Term	Description
equipment phase	An equipment phase is similar to a program: • You run the equipment phase in a task. • You give the equipment phase a set of routines and tags. An equipment phase is different from a program in these ways: • The equipment phase runs by a state model. • You use an equipment phase to do 1 activity of your equipment.
state model	A state model divides the operating cycle of your equipment into a series of states. Each state is an instant in the operation of the equipment. It's the actions or conditions of the equipment at a given time. The state model of an equipment phase is similar to the S88 and Pack*ML* state models.
state machine	An equipment phase includes an embedded state machine that: • calls the main routine (state routine) for an acting state • manages the transitions between states with minimal coding You code the transition conditions. When the conditions are true, the equipment phase transitions the equipment to the next required state. • makes sure that the equipment goes from state to state along an allowable path
PHASE tag	When you add an equipment phase, RSLogix 5000 software makes a tag for the equipment phase. The tag uses the PHASE data type.

Here's how the PhaseManager into RSLogix 5000 programming software:



State Model Overview

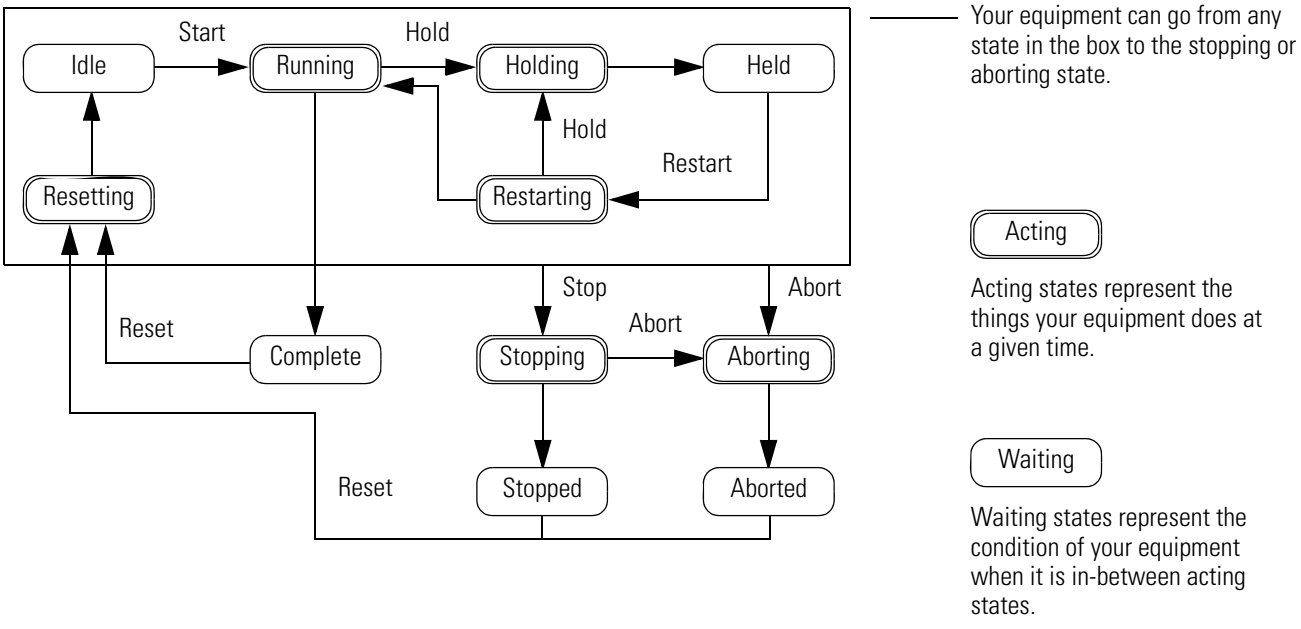
A state model divides the operating cycle of your equipment into a series of states. Each state is an instant in the operation of the equipment. It's the actions or conditions of the equipment at a given time.

In a state model, you define what your equipment does under different conditions, such as run, hold, stop, etc. You don't need to use all the states for your equipment. Use only the states that you want.

There are 2 types of states:

Type of state	Description
Acting	Does something or several things for a certain time or until certain conditions are met. An acting state runs one time or repeatedly.
Waiting	Shows that certain conditions are met and the equipment is waiting for the signal to go to the next state.

PhaseManager uses the following states:



With a state model, you define the behavior of your equipment and put it into a brief functional specification. In this way you show what happens and when it happens.

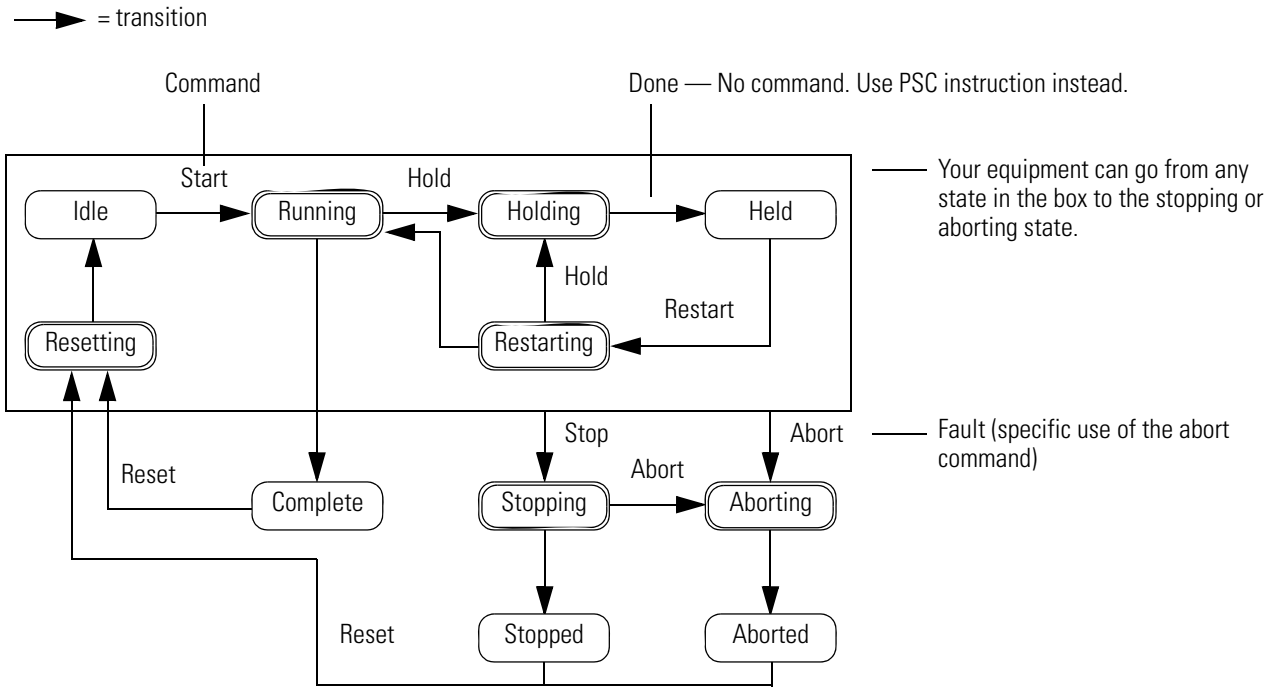
For this State:	Ask:
Stopped	What happens when you turn on power?
Resetting	How does the equipment get ready to run?
Idle	How do you tell that the equipment is ready to run?
Running	What does the equipment do to make product?
Holding	How does the equipment ***temporarily*** stop making product without making scrap?
Held	How do you tell if the equipment is safely holding?
Restarting	How does the equipment resume production after holding?
Complete	How do you tell when the equipment is done with what it had to do?
Stopping	What happens during an normal shutdown?
Aborting	How does the equipment shutdown if a fault or failure happens?
Aborted	How do you tell if the equipment is safely shutdown?

How equipment changes states

The arrows in the state model show to which states your equipment can go from the state it is in now.

- Each arrow is called a transition.
- A state model lets the equipment make only certain transitions. This gives the equipment the same behavior as any other equipment that uses the same model.

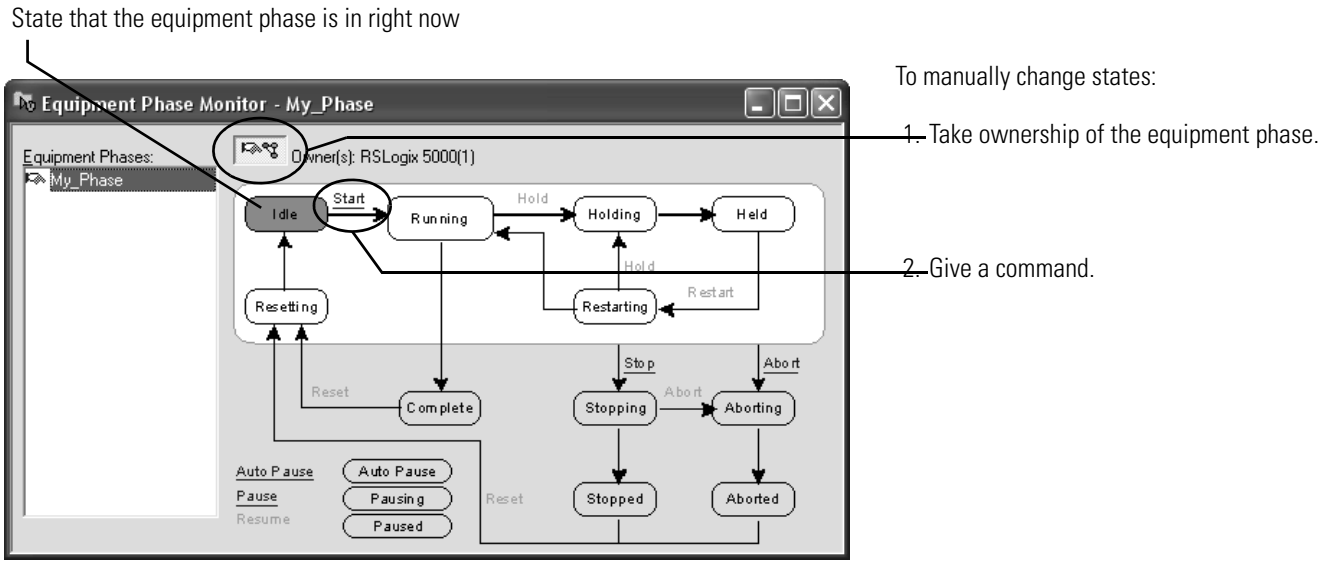
PhaseManager uses the following transitions:



Type of transition	Description
Command	A command tells the equipment to start doing something or do something different. For example the operator pushes the start button to start production and the stop button to shutdown. PhaseManager uses these commands: reset stop restart start hold abort
Done	Equipment goes to a waiting state when it's done with what it's doing. You don't give the equipment a command. Instead, you set up your code to signal when the equipment is done. The waiting state shows that the equipment is done.
Fault	A fault tells you that something out of the ordinary has happened. You set up your code to look for faults and take action if it finds any. Suppose you want your equipment to shut down as fast as possible if a certain fault happens. In that case, set up your code look for that fault and give the abort command if it finds it.

Manually change states

RSLogix 5000 software has a window that lets you monitor and command an equipment phase.



Compare PhaseManager to Other State Models

This table compares PhaseManager's state model to other common state models:

S88	Pack*ML*	PhaseManager
Idle	Starting ⇒ Ready	Resetting ⇒ Idle
Running ⇒ Complete	Producing	Running ⇒ Complete
Pausing ⇒ Paused	Standby	subroutines, breakpoints, or both.
Holding ⇒ Held	Holding ⇒ Held	Holding ⇒ Held
Restarting	none	Restarting
Stopping ⇒ Stopped	Stopping ⇒ Stopped	Stopping ⇒ Stopped
Aborting ⇒ Aborted	Aborting ⇒ Aborted	Aborting ⇒ Aborted

Minimum System Requirements

To develop PhaseManager programs, you need:

- FlexLogix controller with firmware revision 15.0 or later
- communication path to the controller
- RSLogix 5000 software version 15.0 or later

To enable PhaseManager support, you need the full or professional editions of RSLogix 5000 software or the optional PhaseManager add-on (9324-RLDPMENE) to your RSLogix 5000 software package.

Equipment Phase Instructions

The controller supports several instructions to support equipment phases. The instructions are available in ladder diagram (LD) and structured text (ST).

If you want to:	Use this instruction:
signal a phase that the state routine is complete so go to the next state	PSC
change the state or substate of a phase	PCMD
signal a failure for a phase	PFL
clear the failure code of a phase	PCLF
initiate communication with RSBizWare Batch software	PXRQ
clear the NewInputParameters bit of a phase	PRNP
set up breakpoints within the logic of a phase	PPD
take ownership of a phase to either: • prevent another program or RSBizWare Batch software from commanding a phase • make sure another program or RSBizWare Batch software does not already own a phase	PATT
relinquish ownership of a phase	PDET
override a command	POVR

For more information… The PhaseManager User Manual, LOGIX-UM001 provides information on how to design, configure, and program, and phase manager application.

Notes:

Maintain the Battery

Using this Appendix

For information about:	See page
Storing Replacement Batteries	97
Estimating Battery Life	98
Replacing a Battery	99

Storing Replacement Batteries

Because a battery may leak potentially dangerous chemicals if stored improperly, store batteries as follows:

ATTENTION



Store batteries in a cool, dry environment. We recommend 25° C with 40% to 60% relative humidity. You may store batteries for up to 30 days between -45° to 85° C, such as during transportation. To avoid possible leakage, *do not* store batteries above 60° C for more than 30 days.

Estimating Battery Life

When the battery is about 95 percent discharged, the controller provides the following warnings:

- On the front of the controller, the BATTERY LED turns on (solid red).
- A minor fault occurs (type 10, code 10).

To prevent the battery from leaking potentially dangerous chemicals, replace the battery at least as often as:

ATTENTION



To prevent possible battery leakage, even if the BATTERY LED is off, replace the battery according to this schedule:

If the temperature 1 in. below the controller is:	Replace the battery within:
0° to 35° C	No required replacement
36° to 40° C	3 years
41° to 45° C	2 years
46° to 50° C	16 months
51° to 55° C	11 months
56° to 60° C	8 months

To estimate how long the battery will support the memory of the controller:

1. Determine the temperature (° C) 1 in. below the FlexLogix controller.
2. Determine the percentage of time that the controller is powered off per week.

EXAMPLE

If a controller is off:

- 8 hr/day during a 5-day work week
- all day Saturday and Sunday

Then the controller is off 52% of the time:

1. total hours per week = 7 x 24 = 168 hours
2. total off hours per week = (5 days x 8 hrs/day) + Saturday + Sunday = 88 hours
3. percentage off time = 88/168 = 52%

Use the off-time percentage you calculated with the following table to determine battery life:

Catalog number:	Temperature:	Worst-case battery life estimate: Power off 100%:	Power off 50%:	Battery duration after the LED turns on:[1]
1794-L34	60° C	1.8 years	3.6 years	3 days
	25° C	6.7 months	1.1 year	3 days

(1) The battery indicators (BATTERY) warns you when the battery is low. These durations are the amounts of time the battery will retain controller memory from the time the controller is powered down after the LED first turns on.

IMPORTANT If the BATTERY LED turns on when you apply power to the controller, the battery life may be less than the table above indicates. Some of the warning time may have been used while the controller was off and unable to turn on the BATTERY LED.

Replacing a Battery

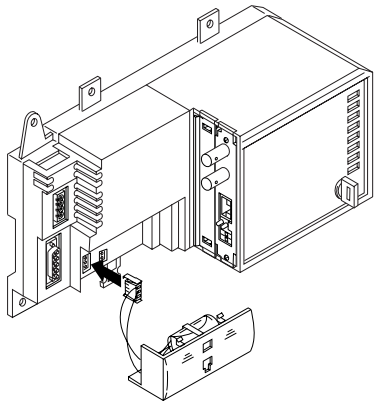
Because the controller uses a lithium battery, you must follow specific precautions when handling or disposing a battery.

ATTENTION  The controller uses a lithium battery, which contains potentially dangerous chemicals. Before handling or disposing a battery, review *Guidelines for Handling Lithium Batteries*, publication AG-5.4.

1. Turn off power to the FlexLogix controller.
2. Does the existing battery show signs of leakage or damage?

If:	Then:
Yes	Before handling the battery, review *Guidelines for Handling Lithium Batteries*, publication AG-5.4.
No	Go to the next step.

3. Remove the old battery.



4. Install a new 1756-BA1 battery.

ATTENTION

Only install a 1756-BA1 battery. If you install a different battery, you may damage the controller.

5. Attach the battery label:
   a. Write on the battery label the date you install the battery.
   b. Attach the label to the inside of the battery compartment.

6. On the front of the controller, is the BATTERY LED off?

If:	Then:
Yes	Go to the next step.
No	A. Check that the battery is correctly connected to the controller. B. If the BATTERY LED remains on, install another 1756-BA1 battery. C. If the BATTERY LED remains on after you complete Step B., contact your Rockwell Automation representative or local distributor.

7. Dispose the old battery according to state and local regulations.

ATTENTION

Do not incinerate or dispose lithium batteries in general trash collection. They may explode or rupture violently. Follow state and local regulations for disposal of these materials. You are legally responsible for hazards created while your battery is being disposed.

FlexLogix System Status Indicators

Controller LEDs

The table below describes the controller LEDs present on all FlexLogix controllers.

If this indicator:	is in this condition:	It means:
RUN	off	The controller is in Program or Test mode.
	steady green	The controller is in Run mode.
FORCE	off	No tags contain I/O force values. I/O forces are inactive (disabled).
	steady amber	I/O forces are active (enabled). I/O force values may or may not exist.
	flashing amber	One or more input or output addresses have been forced to an On or Off state, but the forces have not been enabled.
BAT	off	The battery supports memory.
	steady red	Either the battery is: • not installed. • 95% discharged and should be replaced.
I/O	off	Either: • There are *no* devices in the I/O configuration of the controller. • The controller does *not* contain a project (controller memory is empty).
	steady green	The controller is communicating with all the devices in its I/O configuration.
	flashing green	One or more devices in the I/O configuration of the controller are *not* responding.
	flashing red	The controller is not communicating to any devices. The controller is faulted.
OK	off	No power is applied.
	flashing red	One of the following: • The controller requires a firmware update. • A major recoverable fault occurred on the controller. To clear the fault: 1. Turn the controller keyswitch from PROG to RUN to PROG. 2. Go online with RSLogix 5000.
	steady red	The controller detected a non-recoverable major fault, so it cleared the project from memory. To recover: 1. Cycle power to the chassis. 2. Download the project. 3. Change to Run mode. If the OK LED remains steady red, contact your Rockwell Automation representative or local distributor.
	steady green	Controller is OK.

Notes:

FlexLogix Back-Up on DeviceNet

Using This Appendix

For information about:	See page
How the Back-up Works	104
Power-Up and System Start-up	106
Developing the FlexLogix Back-Up Application	108
Using Indicators to Check Status	115
Development and Debugging Tips	115

This chapter offers a solution to back-up your FlexLogix controller on DeviceNet. FlexLogix Back-Up on DeviceNet is a simple, low-cost, back-up system most effective when used in smaller applications that require fast switchovers from a primary to a secondary controller.

This back-up solution will:

- minimize downtime in case of controller failure when the same program is used in both programs.
- mitigate the risk of changes adversely affecting the application (use old, proven program in one controller and new, untested program in other controller). If the new untested program causes a problem, a forced switchover can be made to the older proven program without downloading the program again.

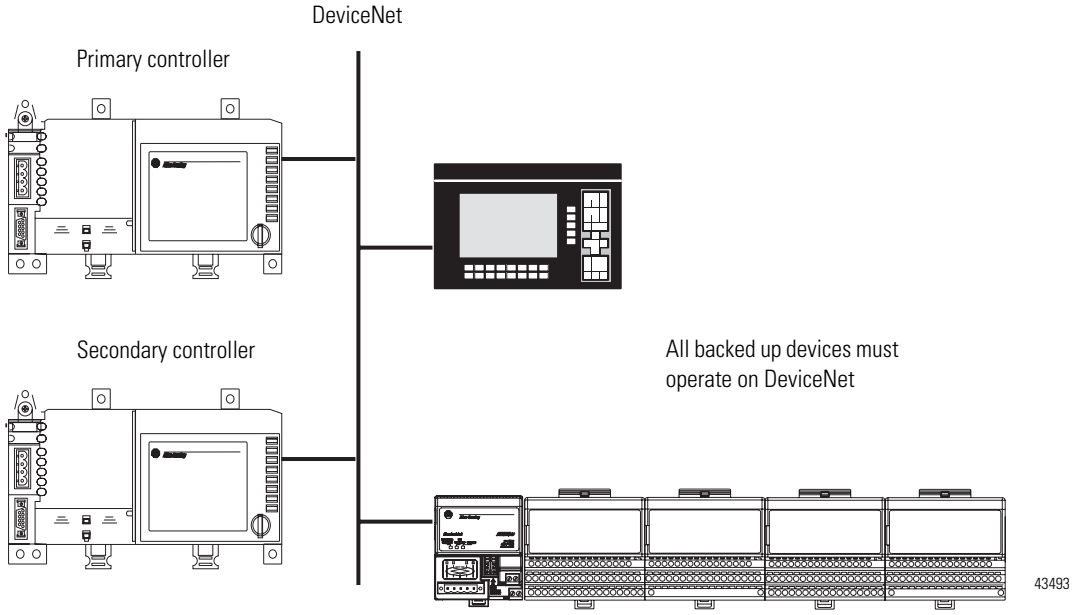
The FlexLogix Back-Up on DeviceNet solution takes advantage of *Shared DeviceNet Mastership of Slave I/O Devices* technology. Typically, only a single DeviceNet master exists for any particular slave. With Shared DeviceNet Mastership, two masters can exist. Heartbeat communications between primary and secondary controllers determines which scanner is the master and which scanner remains in stand-by mode.

How the Back-up Works

Figure shows an example back-up system. In the back-up system, the following occurs:

- Both controllers/scanners simultaneously receive all inputs.
- Both controllers execute in parallel but are NOT synchronized.
- Only the primary controllers sends output data to the I/O devices. A virtual switch in the 1788-DNBO cards is used to switch outputs between primary and secondary controllers.
- After failure or forced switchover, outputs are automatically switched by the 1788-DNBO card from the primary controller to secondary. When the switch occurs, the secondary controller becomes the primary controller.

  The switchover occurs so quickly that the I/O devices do not timeout; these devices are unaware that redundant controllers/scanners exist and are unaware of the switchover.



Requirements of the Back-Up

The FlexLogix Back-Up on DeviceNet solution requires that you use the following:

- RSLogix 5000, version 10 or higher
- 2 FlexLogix controllers, firmware revision 10.x or higher
- 2 1788-DNBO communication cards, firmware revision 2.x or higher

IMPORTANT Many applications use multiple communications cards in a FlexLogix controller to communicate with several networks. This solution requires the software and FlexLogix controllers use version 10.x or higher.

However, if you are using the 1788-ENBT card in your application, remember that you must use software and FlexLogix controllers of version 11.x or higher.

Additional requirements are as follows:

- When setting up the DeviceNet network, you must set the primary and secondary 1788-DNBO cards to the same node address and reserve the next node address.

  We recommend you set the primary and secondary 1788-DNBO node addresses to 0 and reserve node 1. However, you can use any successive node numbers (e.g. 30 and 31).

- All I/O and operator interfaces that required back-up must be on DeviceNet.

- The scanlists in the two DeviceNet scanner must be identical.

Power-Up and System Start-up

To configure a FlexLogix Back-up system on DeviceNet, you can take the following steps. Some of these steps are described in greater detail in the rest of the appendix.

1. Install all I/O and operator interfaces that you need to back-up on DeviceNet.

   We recommend that you reserve node addresses 0 and 1 for the two FlexLogix controllers used in the back-up. If you do not use 0 and 1, make sure you reserve two consecutive numbers for the controllers when you install I/O and other devices on DeviceNet.

2. Connect a FlexLogix controller with a 1788-DNBO scanner to the DeviceNet network.

3. Set the controller node address to 0 (or the lower of the 2 node addresses reserved for the FlexLogix controllers).

4. Power-up the controller and the network.

5. Use RSNetWorx for DeviceNet to download the network's scanlist to the 1788-DNBO card.

   You can use either a scanlist from a new configuration or previously-used configuration. If the scanlist is a new configuration, we recommend you save it to a new project for later use.

6. Use RSLogix5000 software to download the appropriate user program to the FlexLogix controller.

   The program should contain the explicit message(s) that enable the back-up feature for this controller and scanner. The messages are described in the Developing the FlexLogix Back-Up Application section beginning on page 108.

7. Put controller into RUN mode.

8. Either disable power to the controller or disconnect the scanner from DeviceNet. This controller will be the secondary controller.

9. Connect the other FlexLogix controller with a 1788-DNBO scanner on the network.

10. Set the node address to 0.

11. Power-up the controller and scanner.

12. Use RSNetWorx for DeviceNet to download the same scanlist used in step 5.

    It may be necessary to browse the network again before downloading the scanlist. This second browsing of the network allows RSNetWorx for DeviceNet to establish communication to the new scanner at the same node number as the previous scanner.

13. Use RSLogix5000 to download the user program to the second FlexLogix controller as performed in step 6.

    Typically, the same user program is downloaded to the second FlexLogix controller as the first. However, unlike the scanlists, the user programs in the controllers do not have to be identical.

14. Put the controller into RUN mode.

    This controller is now ready to go and is the primary controller.

15. Reapply power to the secondary controller and/or reconnect the secondary scanner to the DeviceNet subnet.

This completes the back-up process. For more detailed information on some of the steps listed previously, see the next section.

Developing the FlexLogix Back-Up Application

The FlexLogix back-up is enabled from an RSLogix 5000 user program with a few simple ladder rungs (or equivalent). The following rungs are used in the FlexLogix back-up:

- Back-up Heartbeat Configuration Rungs - required
- Reading Back-up State Rung - optional
- Reading Back-up Status - optional

Back-up Heartbeat Configuration Rungs

The first, and most critical, step is to set the back-up "heartbeat" constant in the DeviceNet scanner. The heartbeat constant enables the back-up feature and determines the switchover time (2 x heartbeat).

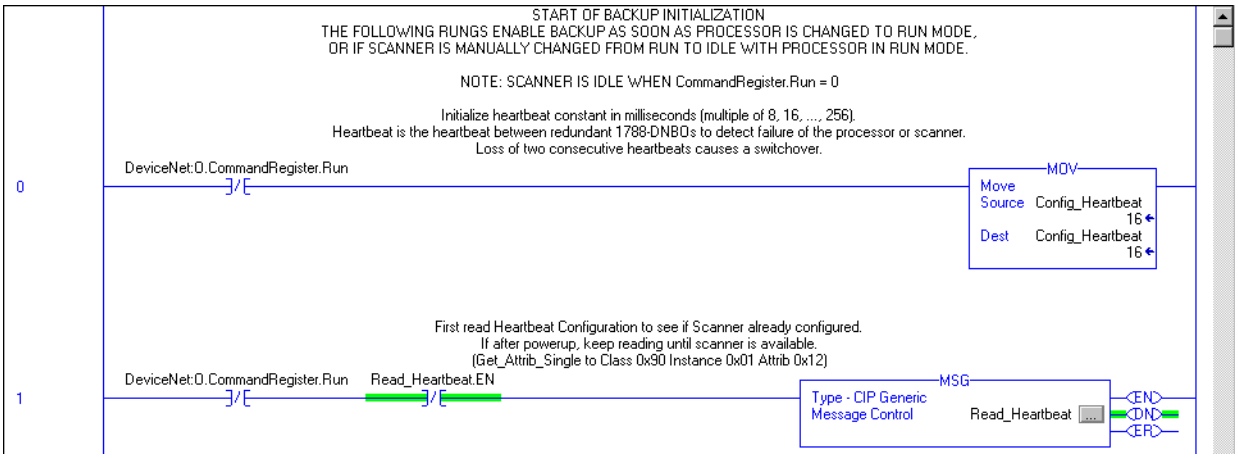
By default, the heartbeat is zero; this default value disables the back-up mode. Your user program must set the heartbeat to a non-zero value to enable back-up.

The heartbeat occurs in multiples of 8ms (i.e. 8, 16, 24, etc.). We recommend a value of 16-48ms for most applications. The recommended heartbeat times result in switchover times of 32-96ms. However, these times do not include controller scan delays.

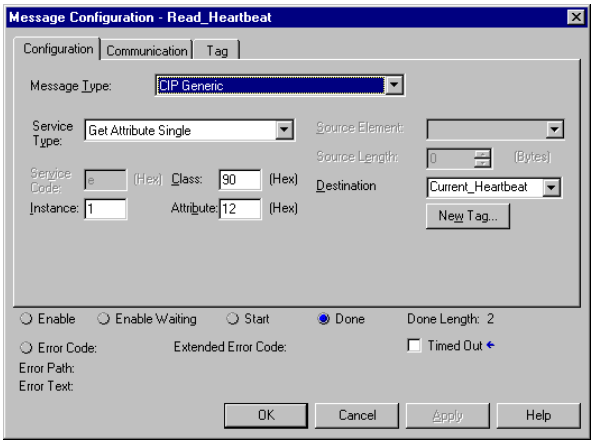
IMPORTANT If multiples of 8 are not used for the requested heartbeat, then the DeviceNet scanner uses the next higher supported heartbeat value that can be read from the scanner. For example, if you set the heartbeat to 10, the scanner uses a 16ms heartbeat.

Setting the Heartbeat Constant

You can set the heartbeat constant with five rungs of ladder logic. Figure shows rungs 0 & 1 and the message set-up used in rung 1. The message in rung 1 uses the INT data type.



Rung 1 message configuration and communication tabs



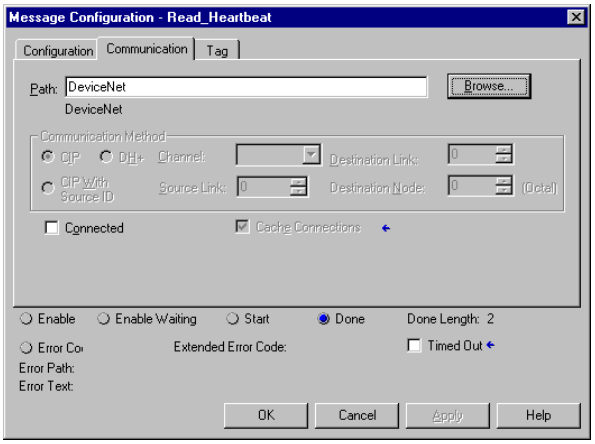
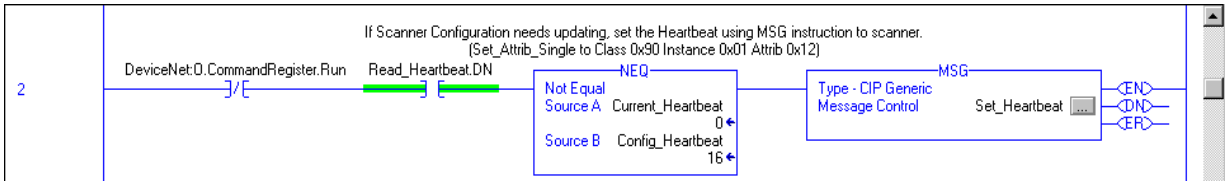
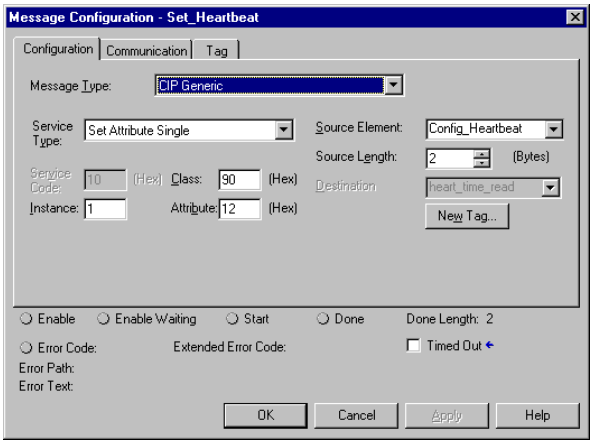


Figure shows rung 2 and the message set-up used on it. The message in rung 2 uses the INT data type.



Rung 2 message configuration and communication tabs



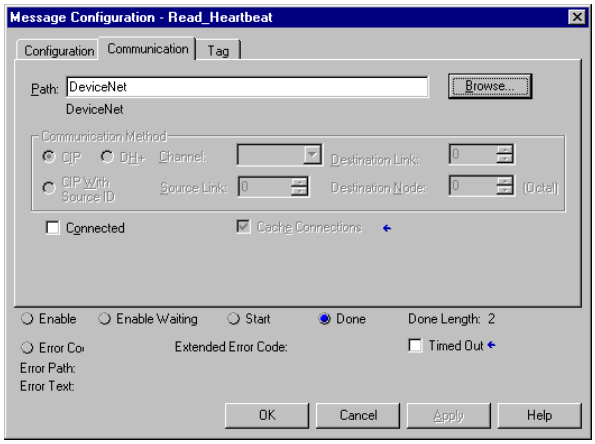
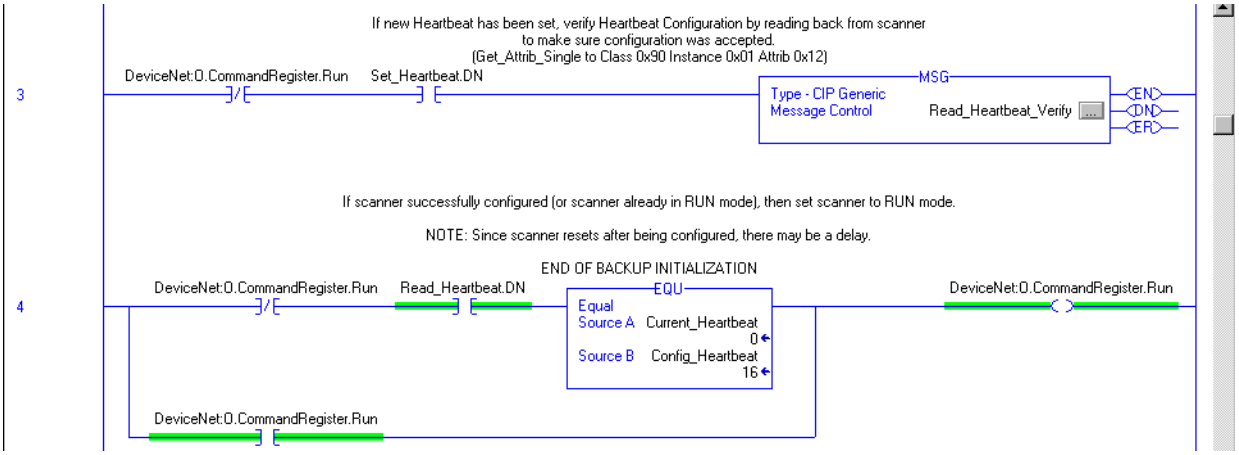
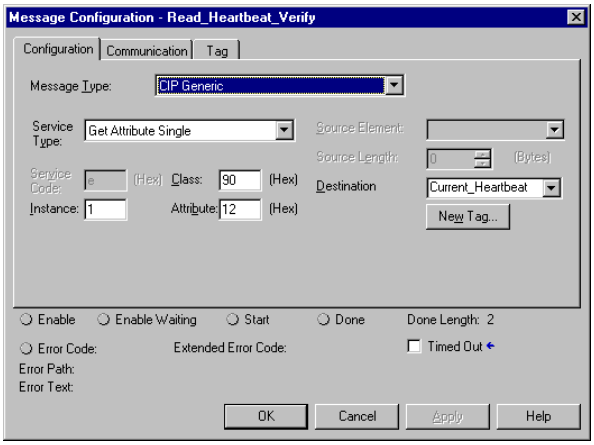
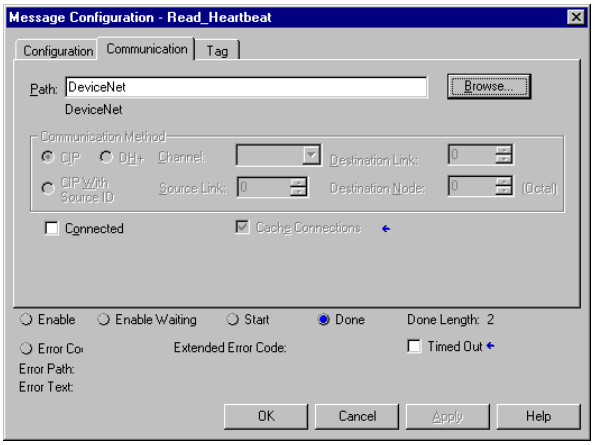


Figure shows rungs 3 & 4 and the message set-up used on it. The message in rung 3 uses the INT data type.



Rung 3 message configuration and communication tabs



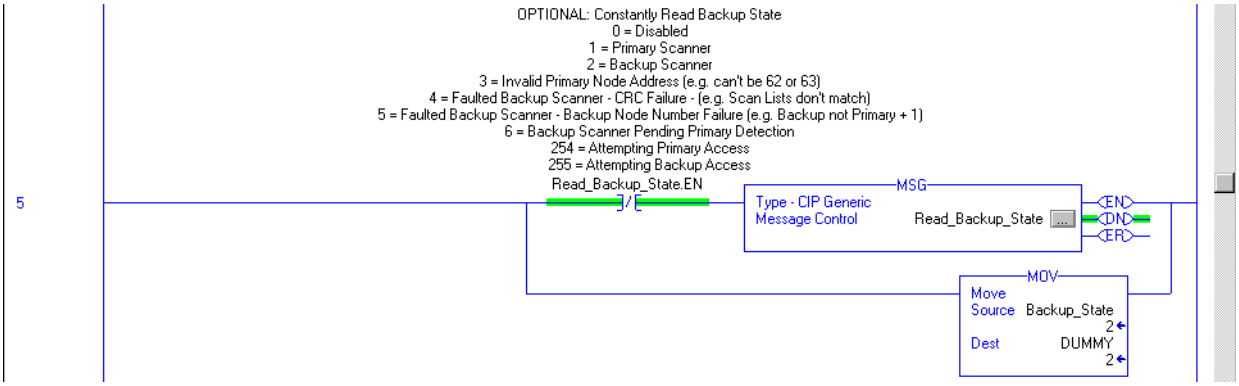


This completes the required portion of ladder logic to enable the FlexLogix back-up on DeviceNet. The following sections describe how to use additional ladder logic to read back-up state and status. However, these sections are not required to complete the back-up solution.

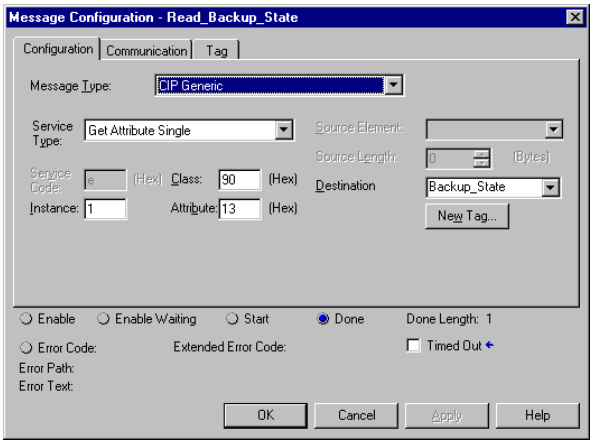
Reading Back-up State Rung

You can read the back-up state of the DeviceNet scanner with a single rung of ladder logic. The back-up state is useful for debug or more sophisticated back-up schemes. The message in this rung uses the SINT data type.

Figure shows the rung you can use to read the back-up state.



Rung 5 message configuration and communication tabs



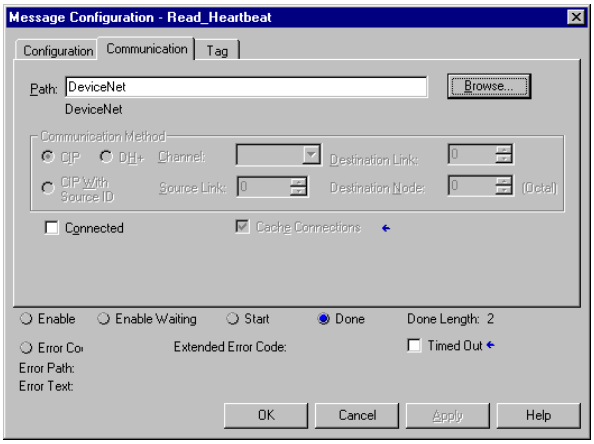


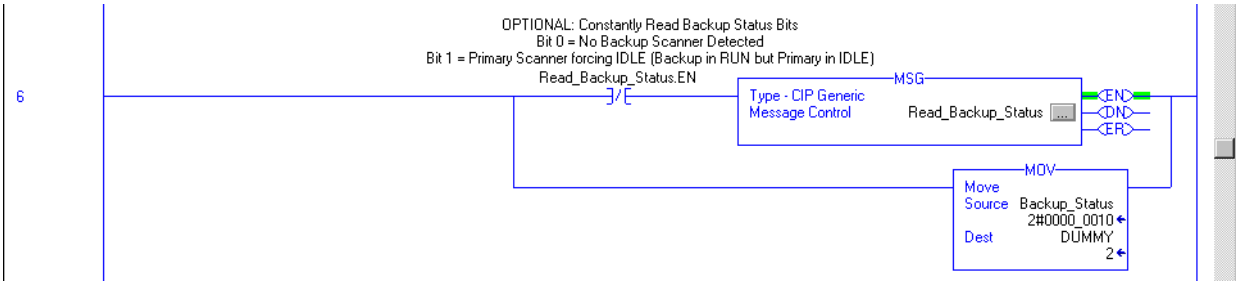
Table describes the possible values this message may return when reading the back-up state of the DeviceNet scanner.

If the message reads this value:	the back-up state of the DeviceNet scanner is:
0	Disabled
1	Primary scanner
2	Back-up scanner
3	Invalid primary node address (e.g. the node address cannot be 62 or 63)
4	Faulted back-up scanner - CRC failure (e.g. the scanlists in the scanners do not match)
5	Faulted back-up scanner - back-up node number failure (e.g. the back-up scanner is not using a node number = the primary node number + 1)
6	Back-up scanner pending primary detection
254	Attempting primary access
255	Attempting back-up access

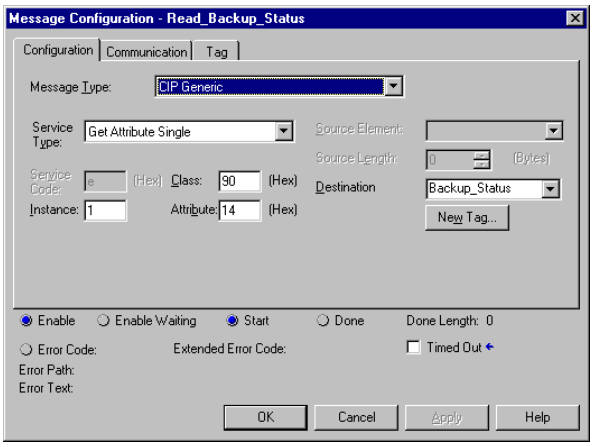
Reading Back-up Status

You can read the back-up status of the DeviceNet scanner with a single rung of ladder logic. The back-up state is useful for debugging or more sophisticated back-up schemes. The message in this rung uses the SINT data type.

Figure shows the rung you can use to read the back-up state.



Rung 6 message configuration and communication tabs



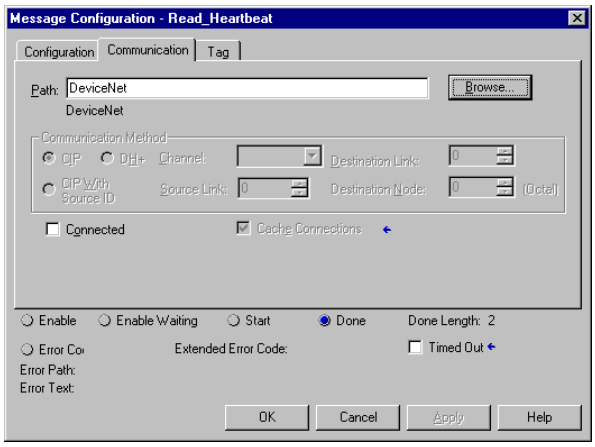


Table describes the possible values this message may return when reading the back-up status of the DeviceNet scanner.

If the message reads this value:	the back-up state of the DeviceNet scanner is:
0	No back-up scanner detected
1	Primary scanner forcing IDLE (back-up in RUN but primary in IDLE)

Using Indicators to Check Status

The 1788-DNBO card's status indicators provide useful information (e.g. determining which controller is primary) about back-up scanner status. Table lists the indicators to monitor when checking back-up status.

If this indicator	exhibits this behavior	this condition exists:
Module status (MS)	Flashing red	A secondary controller was not found (or other minor fault detected)
Back-up status (BS)[1]	Solid green	This scanner is the primary controller.
	Flashing green	This scanner is a qualified secondary controller.
	Off	This scanner is not configured for back-up mode.

[1] The BS status indicator may not be labelled on current 1788-DNBO communication cards.

Development and Debugging Tips

When you implement the FlexLogix Back-Up on DeviceNet solution, we recommend you consider the following development and debugging tips:

- Develop and debug the entire application with only the primary controller and scanner present. When the application is totally verified, then download the program and exact same scanlist to the secondary controller, without the primary controller present. Verify that the secondary is also functioning properly, and then both primary and secondary can be added to the network at the same time.
- No configuration parameters are entered from RSNetworx for DeviceNet or RSLogix5000 to enable Back-up. All configuration occurs in the user program. Almost your entire application (e.g. except for a few ladder rungs) can be developed without knowledge that the application will have a back-up controller and scanner.
- Local I/O still works when this solution is used but the Local I/O is not backed up.
- Switchover time depends on the user configurable heartbeat. After two heartbeats are lost between primary and secondary the switchover occurs. This time can be as little as 50ms with a heartbeat of 16ms.

- The I/O during switchover is NOT bumpless. Since the programs and I/O updates are not synchronized, it is possible for the secondary controller to be either slightly faster or slower than the primary.

  For example, if output changes during a switchover, the fact that the primary and secondary controllers are unsynchronized can cause the output to momentarily switch between an older and newer value. If you configure the switchover time slower than the program scan and I/O update, the secondary lags behind the primary and eliminates this.

- State variables, such as counters or timers are NOT synchronized. The user program must synchronize the primary and secondary controllers, typically over an EtherNet/IP or ControlNet link between controllers. If the outputs are dependent on a state variable, the lack of synchronization can also cause a bumpy switchover.

- As with all back-up and redundancy systems, the I/O must change at a slower rate than the switchover time. If the inputs change faster than the switchover, the change of state is lost.

- Either the user program or user action determine the primary controller. In its simplest mode, the first scanner to power-up or become available on DeviceNet first is the primary.

- Unlike some back-up systems (i.e. PLC5), the primary controller still maintain control of the I/O and switchover does NOT occur if the primary controller is set to Program/Idle mode. The secondary 1788-DNBO scanner also indicates that it is in Idle Mode.

- By default, a switchover will NOT occur if the default fault routine or user fault routine is executed in the primary controller. However, the user fault routine can force a switchover if so desired.

- If an operator interface is on DeviceNet, then it can work without knowledge which controller is primary or secondary.

- Online edits are not automatically performed on both Primary and Secondary since no synchronization exists between Primary and Secondary. Once an online edit occurs on the Primary, then the Primary and Secondary will have different programs.

- FlexLogix Back-up on DeviceNet is not Hot Back-up. Hot Back-up implies complete synchronization of program, program variables, and I/O. Also, I/O switchover is completely bumpless is Hot Back-up.

Instruction Locator

Where to Find an Instruction

This locator table lists the available instructions, which publications describe the instructions, and which programming languages are available for the instructions.

If the locator lists:	The instruction is documented in:
general	Logix5000 Controllers General Instructions Set Reference Manual, 1756-RM003
process control	Logix5000 Controllers Process Control and Drives Instructions Set Reference Manual, 1756-RM006
motion	Logix5000 Controllers Motion Instructions Set Reference Manual, 1756-RM007
phase	PhaseManager User Manual, LOGIX-UM001

Instruction:	Location:	Languages:
ABL ASCII Test For Buffer Line	general	relay ladder structured text
ABS Absolute Value	general	relay ladder structured text function block
ACB ASCII Chars in Buffer	general	relay ladder structured text
ACL ASCII Clear Buffer	general	relay ladder structured text
ACOS Arc Cosine	general	structured text
ACS Arc Cosine	general	relay ladder function block
ADD Add	general	relay ladder structured text function block
AFI Always False Instruction	general	relay ladder
AHL ASCII Handshake Lines	general	relay ladder structured text
ALM Alarm	process control	structured text function block
ALMA Analog Alarm	general	relay ladder structured text function block
ALMD Digital Alarm	general	relay ladder structured text function block

Instruction:	Location:	Languages:
AND Bitwise AND	general	relay ladder structured text function block
ARD ASCII Read	general	relay ladder structured text
ARL ASCII Read Line	general	relay ladder structured text
ASIN Arc Sine	general	structured text
ASN Arc Sine	general	relay ladder function block
ATAN Arc Tangent	general	structured text
ATN Arc Tangent	general	relay ladder function block
AVE File Average	general	relay ladder
AWA ASCII Write Append	general	relay ladder structured text
AWT ASCII Write	general	relay ladder structured text
BAND Boolean AND	general	structured text function block
BNOT Boolean NOT	general	structured text function block
BOR Boolean OR	general	structured text function block

Instruction:	Location:	Languages:
BRK Break	general	relay ladder
BSL Bit Shift Left	general	relay ladder
BSR Bit Shift Right	general	relay ladder
BTD Bit Field Distribute	general	relay ladder
BTDT Bit Field Distribute with Target	general	structured text function block
BTR Message	general	relay ladder structured text
BTW Message	general	relay ladder structured text
BXOR Boolean Exclusive OR	general	structured text function block
CLR Clear	general	relay ladder structured text
CMP Compare	general	relay ladder
CONCAT String Concatenate	general	relay ladder structured text
COP Copy File	general	relay ladder structured text
COS Cosine	general	relay ladder structured text function block
CPS Synchronous Copy File	general	relay ladder structured text
CPT Compute	general	relay ladder
CTD Count Down	general	relay ladder
CTU Count Up	general	relay ladder
CTUD Count Up/Down	general	structured text function block
D2SD Discrete 2-State Device	process control	structured text function block
D3SD Discrete 3-State Device	process control	structured text function block
DDT Diagnostic Detect	general	relay ladder
DEDT Deadtime	process control	structured text function block
DEG Degrees	general	relay ladder structured text function block
DELETE String Delete	general	relay ladder structured text
DERV Derivative	process control	structured text function block

Instruction:	Location:	Languages:
DFF D Flip-Flop	process control	structured text function block
DIV Divide	general	relay ladder structured text function block
DTOS DINT to String	general	relay ladder structured text
DTR Data Transitional	general	relay ladder
EOT End of Transition	general	relay ladder structured text
EQU Equal to	general	relay ladder structured text function block
ESEL Enhanced Select	process control	structured text function block
EVENT Trigger Event Task	general	relay ladder structured text
FAL File Arithmetic and Logic	general	relay ladder
FBC File Bit Comparison	general	relay ladder
FFL FIFO Load	general	relay ladder
FFU FIFO Unload	general	relay ladder
FGEN Function Generator	process control	structured text function block
FIND Find String	general	relay ladder structured text
FLL File Fill	general	relay ladder
FOR For	general	relay ladder
FRD Convert to Integer	general	relay ladder function block
FSC File Search and Compare	general	relay ladder
GEQ Greater than or Equal to	general	relay ladder structured text function block
GRT Greater Than	general	relay ladder structured text function block
GSV Get System Value	general	relay ladder structured text
HLL High/Low Limit	process control	structured text function block
HPF High Pass Filter	process control	structured text function block
ICON Input Wire Connector	general	function block

Instruction:	Location:	Languages:
INSERT Insert String	general	relay ladder structured text
INTG Integrator	process control	structured text function block
IOT Immediate Output	general	relay ladder structured text
IREF Input Reference	general	function block
JKFF JK Flip-Flop	process control	structured text function block
JMP Jump to Label	general	relay ladder
JSR Jump to Subroutine	general	relay ladder structured text function block
JXR Jump to External Routine	general	relay ladder
LBL Label	general	relay ladder
LDL2 Second-Order Lead Lag	process control	structured text function block
LDLG Lead-Lag	process control	structured text function block
LEQ Less Than or Equal to	general	relay ladder structured text function block
LES Less Than	general	relay ladder structured text function block
LFL LIFO Load	general	relay ladder
LFU LIFO Unload	general	relay ladder
LIM Limit	general	relay ladder function block
LN Natural Log	general	relay ladder structured text function block
LOG Log Base 10	general	relay ladder structured text function block
LOWER Lower Case	general	relay ladder structured text
LPF Low Pass Filter	process control	structured text function block
MAAT Motion Apply Axis Tuning	motion	relay ladder structured text
MAFR Motion Axis Fault Reset	motion	relay ladder structured text
MAG Motion Axis Gear	motion	relay ladder structured text
MAH Motion Axis Home	motion	relay ladder structured text
MAHD Motion Apply Hookup Diagnostics	motion	relay ladder structured text
MAJ Motion Axis Jog	motion	relay ladder structured text
MAM Motion Axis Move	motion	relay ladder structured text
MAOC Motion Arm Output Cam	motion	relay ladder structured text
MAPC Motion Axis Position Cam	motion	relay ladder structured text
MAR Motion Arm Registration	motion	relay ladder structured text
MAS Motion Axis Stop	motion	relay ladder structured text
MASD Motion Axis Shutdown	motion	relay ladder structured text
MASR Motion Axis Shutdown Reset	motion	relay ladder structured text
MATC Motion Axis Time Cam	motion	relay ladder structured text
MAVE Moving Average	process control	structured text function block
MAW Motion Arm Watch	motion	relay ladder structured text
MAXC Maximum Capture	process control	structured text function block
MCCD Motion Coordinated Change Dynamics	motion	relay ladder structured text
MCCM Motion Coordinated Circular Move	motion	relay ladder structured text
MCCP Motion Calculate Cam Profile	motion	relay ladder structured text
MCD Motion Change Dynamics	motion	relay ladder structured text
MCLM Motion Coordinated Linear Move	motion	relay ladder structured text
MCR Master Control Reset	general	relay ladder
MCS Motion Coordinated Stop	motion	relay ladder structured text
MCSD Motion Coordinated Shutdown	motion	relay ladder structured text
MCSR Motion Coordinated Shutdown Reset	motion	relay ladder structured text
MCT Motion Coordinated Transform	motion	relay ladder structured text
MCTP Motion Calculate Transform Position	motion	relay ladder structured text

Instruction:	Location:	Languages:
MDF Motion Direct Drive Off	motion	relay ladder structured text
MDO Motion Direct Drive On	motion	relay ladder structured text
MDOC Motion Disarm Output Cam	motion	relay ladder structured text
MDR Motion Disarm Registration	motion	relay ladder structured text
MDW Motion Disarm Watch	motion	relay ladder structured text
MEQ Mask Equal to	general	relay ladder structured text function block
MGS Motion Group Stop	motion	relay ladder structured text
MGSD Motion Group Shutdown	motion	relay ladder structured text
MGSP Motion Group Strobe Position	motion	relay ladder structured text
MGSR Motion Group Shutdown Reset	motion	relay ladder structured text
MID Middle String	general	relay ladder structured text
MINC Minimum Capture	process control	structured text function block
MOD Modulo	general	relay ladder structured text function block
MOV Move	general	relay ladder
MRAT Motion Run Axis Tuning	motion	relay ladder structured text
MRHD Motion Run Hookup Diagnostics	motion	relay ladder structured text
MRP Motion Redefine Position	motion	relay ladder structured text
MSF Motion Servo Off	motion	relay ladder structured text
MSG Message	general	relay ladder structured text
MSO Motion Servo On	motion	relay ladder structured text
MSTD Moving Standard Deviation	process control	structured text function block
MUL Multiply	general	relay ladder structured text function block
MUX Multiplexer	process control	function block
MVM Masked Move	general	relay ladder
MVMT Masked Move with Target	general	structured text function block
NEG Negate	general	relay ladder structured text function block
NEQ Not Equal to	general	relay ladder structured text function block
NOP No Operation	general	relay ladder
NOT Bitwise NOT	general	relay ladder structured text function block
NTCH Notch Filter	process control	structured text function block
OCON Output Wire Connector	general	function block
ONS One Shot	general	relay ladder
OR Bitwise OR	general	relay ladder structured text function block
OREF Output Reference	general	function block
OSF One Shot Falling	general	relay ladder
OSFI One Shot Falling with Input	general	structured text function block
OSR One Shot Rising	general	relay ladder
OSRI One Shot Rising with Input	general	structured text function block
OTE Output Energize	general	relay ladder
OTL Output Latch	general	relay ladder
OTU Output Unlatch	general	relay ladder
PATT Attach to Equipment Phase	phase	relay ladder structured text
PCLF Equipment Phase Clear Failure	phase	relay ladder structured text
PCMD Equipment Phase Command	phase	relay ladder structured text
PDET Detach from Equipment Phase	phase	relay ladder structured text
PFL Equipment Phase Failure	phase	relay ladder structured text
PI Proportional + Integral	process control	structured text function block
PID Proportional Integral Derivative	general	relay ladder structured text

Instruction:	Location:	Languages:
PIDE Enhanced PID	process control	structured text function block
PMUL Pulse Multiplier	process control	structured text function block
PPD Equipment Phase Paused	phase	relay ladder structured text
POSP Position Proportional	process control	structured text function block
PRNP Equipment Phase New Parameters	phase	relay ladder structured text
PSC Phase State Complete	phase	relay ladder structured text
PXRQ Equipment Phase External Request	phase	relay ladder structured text
RAD Radians	general	relay ladder structured text function block
RES Reset	general	relay ladder
RESD Reset Dominant	process control	structured text function block
RET Return	general	relay ladder structured text function block
RLIM Rate Limiter	process control	structured text function block
RMPS Ramp/Soak	process control	structured text function block
RTO Retentive Timer On	general	relay ladder
RTOR Retentive Timer On with Reset	general	structured text function block
RTOS REAL to String	general	relay ladder structured text
SBR Subroutine	general	relay ladder structured text function block
SCL Scale	process control	structured text function block
SCRV S-Curve	process control	structured text function block
SEL Select	process control	function block
SETD Set Dominant	process control	structured text function block
SFP SFC Pause	general	relay ladder structured text
SFR SFC Reset	general	relay ladder structured text

Instruction:	Location:	Languages:
SIN Sine	general	relay ladder structured text function block
SIZE Size In Elements	general	relay ladder structured text
SNEG Selected Negate	process control	structured text function block
SOC Second-Order Controller	process control	structured text function block
SQI Sequencer Input	general	relay ladder
SQL Sequencer Load	general	relay ladder
SQO Sequencer Output	general	relay ladder
SQR Square Root	general	relay ladder function block
SQRT Square Root	general	structured text
SRT File Sort	general	relay ladder structured text
SRTP Split Range Time Proportional	process control	structured text function block
SSUM Selected Summer	process control	structured text function block
SSV Set System Value	general	relay ladder structured text
STD File Standard Deviation	general	relay ladder
STOD String To DINT	general	relay ladder structured text
STOR String To REAL	general	relay ladder structured text
SUB Subtract	general	relay ladder structured text function block
SWPB Swap Byte	general	relay ladder structured text
TAN Tangent	general	relay ladder structured text function block
TND Temporary End	general	relay ladder
TOD Convert to BCD	general	relay ladder function block
TOF Timer Off Delay	general	relay ladder
TOFR Timer Off Delay with Reset	general	structured text function block
TON Timer On Delay	general	relay ladder

Instruction:	Location:	Languages:
TONR Timer On Delay with Reset	general	structured text function block
TOT Totalizer	process control	structured text function block
TRN Truncate	general	relay ladder function block
TRUNC Truncate	general	structured text
UID User Interrupt Disable	general	relay ladder structured text
UIE User Interrupt Enable	general	relay ladder structured text
UPDN Up/Down Accumulator	process control	structured text function block
UPPER Upper Case	general	relay ladder structured text
XIC Examine If Closed	general	relay ladder
XIO Examine If Open	general	relay ladder
XOR Bitwise Exclusive OR	general	relay ladder structured text function block
XPY X to the Power of Y	general	relay ladder structured text function block

A

add-on instructions 77
address data 62
AOI 77
architecture 11
ASCII characters 37-38

B

back-up
- FlexLogix back-up on DeviceNet 103-116

battery
- catalog number 97
- how to replace 99
- life 98
- storage 97
- when to replace 98

BOOTP 22

C

cables
- DH-485 link cable length 40

cache message 47
calculate connection use 50
change of state 56
command
- give 92

communication
- ControlNet 25-28
- determine timeout with any device 80
- determine timeout with I/O module 80
- DeviceNet 28-30
- DH-485 39, 39-41
- EtherNet/IP 22-24
- format 56
- serial 31-38

configuration folder 55
configure
- ControlNet I/O module 60
- DeviceNet I/O module 61
- EtherNet/IP I/O module 59
- I/O module 55-61
- serial driver 18

connect
- ControlNet 25-28
- DeviceNet 28-30
- DH-485 39, 39-41
- EtherNet/IP 22-24
- serial 15-20, 31-38
- third party network 42

connection
- calculate use 50
- consume data 45
- ControlNet 26
- determine timeout with any device 80
- determine timeout with I/O module 80
- EtherNet/IP 24
- example 52
- for more information 49
- I/O module 57
- limits 24, 27, 49
- message 47-48
- monitor 80-82
- monitor rack-optmized 65
- overview 45
- produce data 45
- summary 49

consume data
- connection use 45
- for more information 46
- overview 21

control distributed I/O
- overview 21

controller
- consume data 21
- control distributed I/O 21
- design 12
- fault handler 81
- install 13
- message 21
- monitor status 79
- path 20
- produce data 21
- serial connection 15-20
- status 79

ControlNet
- connection use 26
- distributed I/O 60
- example configuration 26
- for more information 28
- module capability 25
- overview 25-28
- scheduled 27
- software use 25
- unscheduled 27

D

design 12
develop application
- define tasks 71-73
- fault handler 81
- for more information 74
- monitor connection 80-82
- monitor status 79
- overview 69
- programming language 76
- programs 70
- sample controller projects 74
- tag 75
- task 69
- using event task 85-87

DeviceNet
- distributed I/O 61
- example configuration 29
- FlexLogix back-up on the network 103-116
- for more information 30
- module capability 29
- overview 28-30
- software use 29

DF1 device 33
DH-485
- cables 40
- controller communication 39-41
- controller configuration 41
- overview 39

DHCP 22
direct connection 57
distributed I/O
- ControlNet 60
- DeviceNet 61
- EtherNet/IP 59
- overview 21

E

electronic keying 56
equipment phase
- compared to PackML 94
- compared to S88 94
- instructions 89
- monitor 94
- overview 89

equipment phase instructions
- overview 89

EtherNet/IP
- connection use 24
- distributed I/O 59
- example configuration 23
- for more information 24
- module capability 23
- overview 22-24
- software use 22

event tasks 85-87
example system 11

F

fault data 64
fault handler 81
FBD 76
function block diagram 76

G

GSV instruction 79

H

hardware installation 13

I

I/O
- address data 62
- communication format 56
- configuration folder 55
- configure 53
- connection use 57
- COS 56
- determine update 63
- direct connection 57
- distributed via ControlNet 60
- distributed via DeviceNet 61
- distributed via EtherNet/IP 59
- electronic keying 56
- for more information 58
- local and extended-local DIN rails 54
- module capability 53
- monitor 53, 64-65
- monitor connection 80
- place 53
- rack-optimized 57
- reconfigure module 66-67
- RPI 56
- select 1794 FLEX I/O 53
- select a 1794 FLEX power supply 54

I/O module
- fault data 64

install
- hardware 13

instruction locator 117
isolator
connected to the serial port 15-17

L

ladder diagram 76
language 76
locator 117
Logix5000 controller environment 11
low battery 98

M

message
cache 47
connection use 47-48
for more information 48
overview 21
reconfigure I/O module 67
Modbus support 38
monitor
rack-optimized connection 65

N

network
overview 21

P

phase
See equipment phase
priority 71
produce data
connection use 45
for more information 46
overview 21
program
defining 73
programming language 76
project
program 73
routine 73
task 71

R

rack-optimized connection 57
reconfigure I/O module 66-67
related documentation 9-10
relay ladder 76
replace the battery
how 99
when 98
requested packet interval 56
routine
defining 73
RPI 56
RS-232 DF1 Device driver 18
RSLinx 25
RSLogix 5000 22, 25, 29
controller path 20
serial driver 18-19
RSNetWorx for ControlNet 25
RSNetWorx for DeviceNet 29
RSNetWorx for EtherNet/IP 22

S

scheduled 27
sequential function chart 76
serial
cable 33
communicate with ASCII device
ASCII device 35
communicate with DF1 device 33
connect an isolator 15-17
controller communication 31-38
controller connection 15-20
DH-485 configuration 39, 41
driver 18
for more information 34, 38
Modbus support 38
modes 31
RSLogix 5000 18-20
select controller path 20
SFC 76
SSV instruction 79
ST 76
start 11
state model
See states
states
compared to PackML 94
compared to S88 94
manually step through 94
overview 91
transition 92
status 79
status indicators 101
store batteries 97
structured text 76
system layout 11

T

tag
consumed 46
for more information 75
organize 75
produced 46
task 69
defining 71
priority 71
troubleshooting 101

U

unscheduled 27
update 63

W

where to start 11

How Are We Doing?

Your comments on our technical publications will help us serve you better in the future. Thank you for taking the time to provide us feedback.

You can complete this form and mail (or fax) it back to us or email us at RADocumentComments@ra.rockwell.com

Pub. Title/Type FlexLogix Controller System User Manual

Cat. No. 1794-L34 Pub. No. 1794-UM001G-EN-P Pub. Date January 2007 Part No. 953014-91

Please complete the sections below. Where applicable, rank the feature (1=needs improvement, 2=satisfactory, and 3=outstanding).

Overall Usefulness 1 2 3	How can we make this publication more useful for you?	
Completeness (all necessary information is provided) 1 2 3	Can we add more information to help you?	
	procedure/step	illustration feature
	example	guideline other
	explanation	definition
Technical Accuracy (all provided information is correct) 1 2 3	Can we be more accurate?	
	text	illustration
Clarity (all provided information is easy to understand) 1 2 3	How can we make things clearer?	
Other Comments	You can add additional comments on the back of this form.	

Your Name _____

Your Title/Function _____

Location/Phone _____

Would you like us to contact you regarding your comments?

___No, there is no need to contact me

___Yes, please call me

___Yes, please email me at _____

___Yes, please contact me via _____

Return this form to: Rockwell Automation Technical Communications, 1 Allen-Bradley Dr., Mayfield Hts., OH 44124-9705

Fax: 440-646-3525 Email: RADocumentComments@ra.rockwell.com

PLEASE FASTEN HERE (DO NOT STAPLE)

Other Comments

PLEASE FOLD HERE

PLEASE REMOVE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST-CLASS MAIL PERMIT NO. 18235 CLEVELAND OH

POSTAGE WILL BE PAID BY THE ADDRESSEE

Allen-Bradley
RELIANCE ELECTRIC
DODGE
ROCKWELL SOFTWARE
Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Rockwell Automation Support

Rockwell Automation provides technical information on the Web to assist you in using its products. At http://support.rockwellautomation.com, you can find technical manuals, a knowledge base of FAQs, technical and application notes, sample code and links to software service packs, and a MySupport feature that you can customize to make the best use of these tools.

For an additional level of technical phone support for installation, configuration, and troubleshooting, we offer TechConnect Support programs. For more information, contact your local distributor or Rockwell Automation representative, or visit http://support.rockwellautomation.com.

Installation Assistance

If you experience a problem with a hardware module within the first 24 hours of installation, please review the information that's contained in this manual. You can also contact a special Customer Support number for initial help in getting your module up and running.

United States	1.440.646.3223 Monday – Friday, 8am – 5pm EST
Outside United States	Please contact your local Rockwell Automation representative for any technical support issues.

New Product Satisfaction Return

Rockwell tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning, it may need to be returned.

United States	Contact your distributor. You must provide a Customer Support case number (see phone number above to obtain one) to your distributor in order to complete the return process.
Outside United States	Please contact your local Rockwell Automation representative for return procedure.

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, Vorstlaan/Boulevard du Souverain 36, 1170 Brussels, 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 1794-UM001G-EN-P - January 2007 PN 953014-91
Supersedes Publication 1794-UM001F-EN-P - May 2005
Copyright © 2007 Rockwell Automation, Inc. All rights reserved. Printed in the U.S.A.

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP