Allen-Bradley

SoftLogix5800 System

1789-L10, -L30, -L60

User Manual

Rockwell Automation

Important User Information

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

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

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

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

ATTENTION

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

Attention statements help you to:

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

IMPORTANT

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

European Communities (EC) Directive Compliance

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

EMC Directive

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

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

This product is intended for use in an industrial environment.

Low Voltage Directive

This product is tested to meet Council Directive 73/23/EEC Low Voltage, by applying the safety requirements of EN 61131-2 Programmable Controllers, Part 2 - Equipment Requirements and Tests. For specific information required by EN 61131-2, see the appropriate sections in this publication, as well as the Allen-Bradley publication Industrial Automation Wiring and Grounding Guidelines For Noise Immunity, publication 1770-4.1.

This equipment is classified as open equipment and must be mounted in an enclosure during operation to provide safety protection.

Notes:

Introduction

This version of the SoftLogix5800 System User Manual corresponds to version 8 of the controller. Revision bars (shown in the left margin of this page) indicate changed information. Changes made to this manual include:

For this information:	See:
addition of 1789-L10 and 1789-L30 controllers	2-2
how to select the 1784-PCICS card through an online path of available devices	4-33
how to select the 1784-PCIDS card through an online path of available devices	5-17
information about remote programming over an Ethernet link	chapter 7
information about installing and configuring a 1789-SIM module in an RSLogix5000 project	chapter 8

Notes:

Purpose of this Manual

This manual guides the development of projects for SoftLogix5800 controllers. It provides procedures on how to establish communications and/or programming links over these networks:

- ControlNet
- DeviceNet
- Ethernet
- serial

This manual works together with the *Logix5000 Controllers Common Procedures Programming Manual*, publication 1756-PM001, which covers the following tasks:

- Manage project files
- Organize your logic
- Organize tags
- Program routines
- Test a project
- Handle faults

Who Should Use This Manual

This manual is intended for those individuals who program applications that use SoftLogix controllers, such as:

- software engineers
- control engineers
- application engineers
- instrumentation technicians

When to Use This Manual

Use this manual:

- when you are ready to integrate your application with the I/0 devices, controllers, and networks in your system.
- *after* you perform these actions:
  - develop the basic code for your application
  - perform isolated tests of your application

How to Use this Manual

This manual is divided into the basic tasks that you perform while programming a SoftLogix controller. Each chapter covers a main task, such as communicating over a specific network. For each main task, the chapter:

- lists what you need
- describes the steps to follow to accomplish that task
- provides details for each step, as necessary
- includes example system configurations

Chapter 1

Getting Started

Chapter 2

What Is SoftLogix?

Chapter 3

Controlling Motion Devices

Chapter 4

Communicating with Devices on a ControlNet Link

Chapter 5

Communicating with Devices on a DeviceNet Link

Chapter 6

Programming Over an Ethernet Link

Chapter 7

Communicating with Devices on a Serial Link

Chapter 8

Configuring and Using Simulated I/O

Appendix A

Windows NT/2000 Considerations

Appendix B

Monitoring Controller LEDs

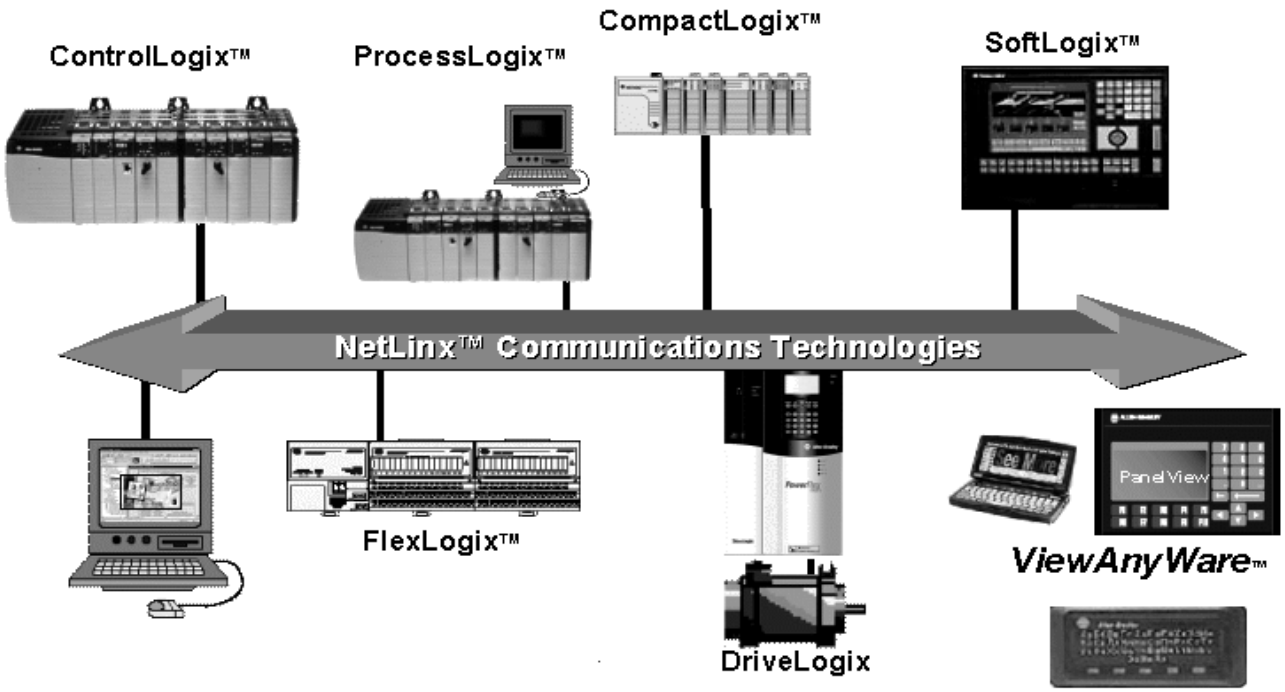
Getting Started

Introduction

This chapter introduces the SoftLogix5800 family of controllers and provides a quick overview on creating and downloading a project. This chapter introduces basic aspects of the SoftLogix controller.

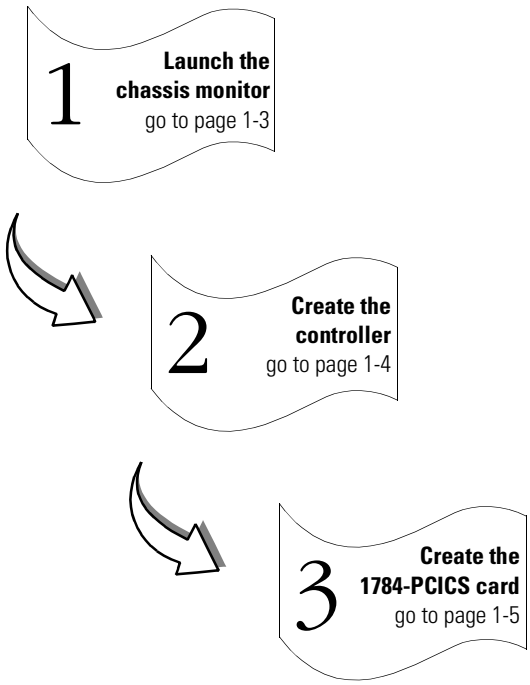
The SoftLogix controller is another platform in the Logix environment. The SoftLogix controller offers:

- high-speed logic execution
- seamless integration of control, motion, and information structures
- compatibility with a range of Allen-Bradley, Rockwell Software, and Microsoft-compatible products

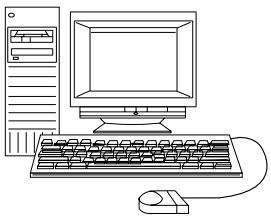


Creating and Configuring the Controller

The following diagram illustrates the steps you follow to create and configure a SoftLogix controller. Use the SoftLogix chassis monitor to perform these steps.



System setup for this quick start:



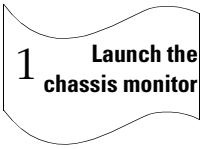
- SoftLogix5800 controller in slot 1 of the virtual chassis
- 1784-PCICS card in slot 2 of the virtual chassis

You need:

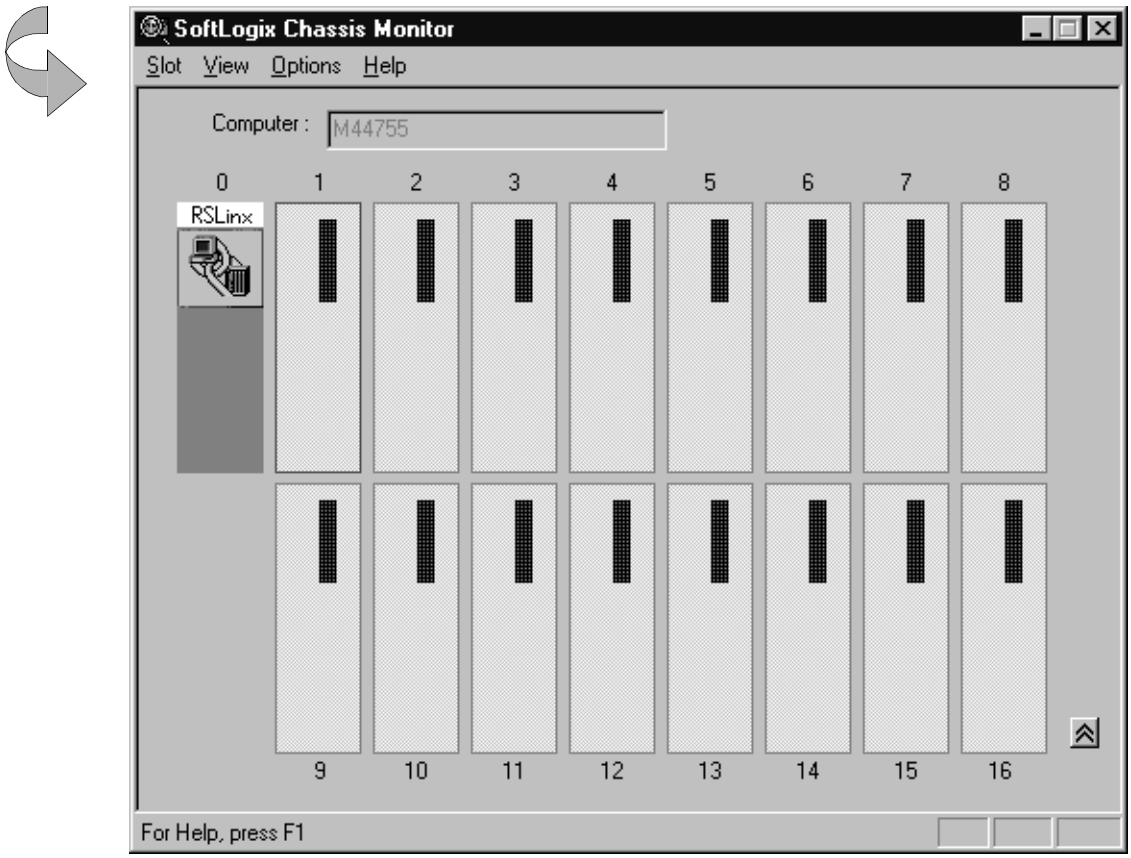
- SoftLogix chassis monitor
- RSLinx communication software

Launching the chassis monitor

1. **Double-click the SoftLogix chassis monitor icon on your computer desktop or in the system tray.**



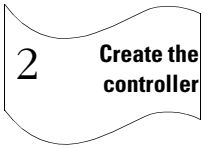
The default chassis monitor appears.



Once the SoftLogix chassis monitor is running and a card is installed in the virtual chassis, the controller automatically starts when you reboot your computer.

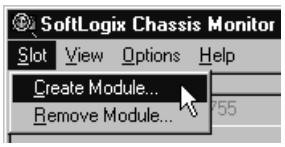
IMPORTANT The chassis monitor is your window to monitoring the SoftLogix system. If you close the chassis monitor, the controller and it's modules are still running. To stop a controller or associated module, remove it from the virtual chassis.

Creating the controller



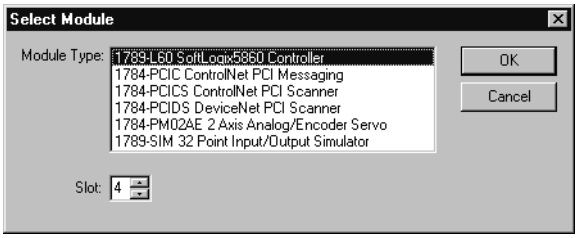
1. Create a new module.

Select Slot → Create Module
or right click the appropriate slot and select Create.



2. Select the controller.

Specify the chassis slot number.

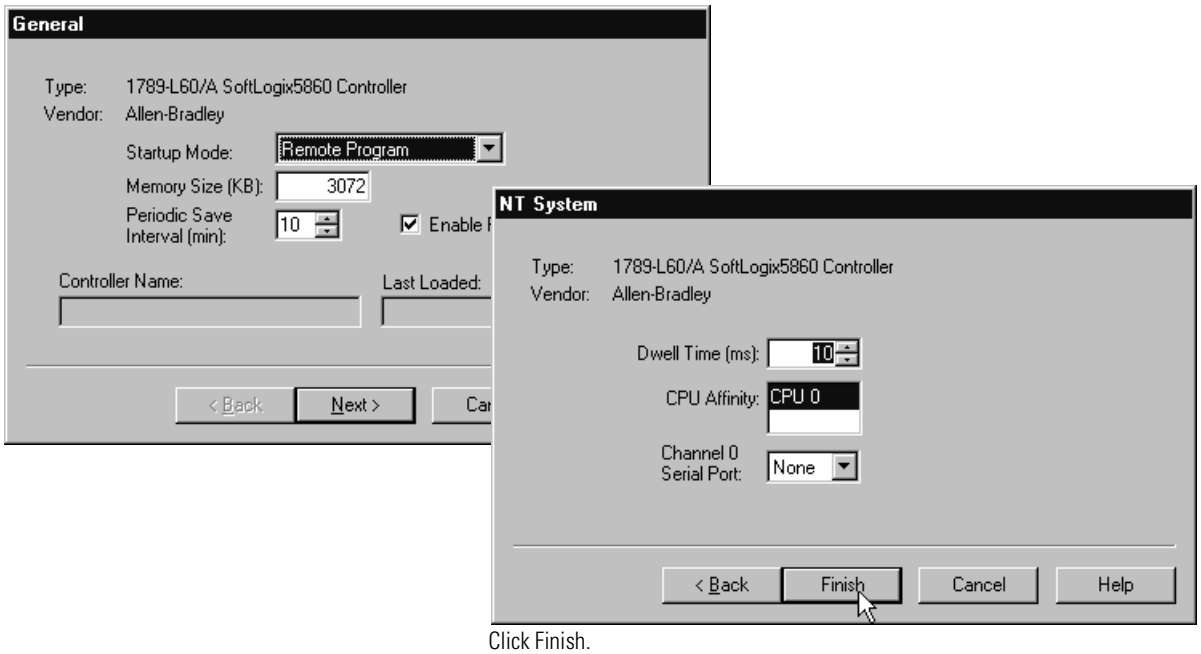


Click OK

3. Configure the controller.

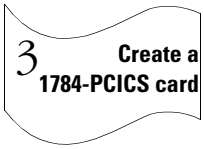
Specify configuration settings for the controller:

- startup mode
- memory size
- periodic save interval
- dwell time
- CPU affinity
- COM port to use for serial communications



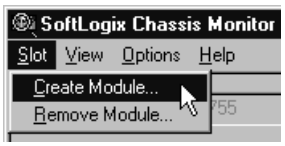
Click Finish.

Creating a 1784-PCICS card



1. Create a new module.

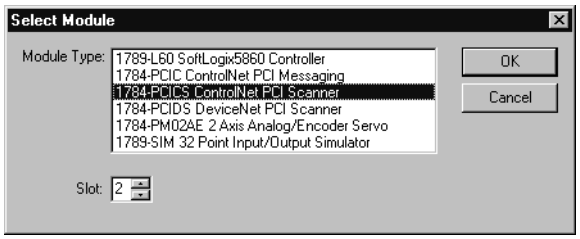
Select Slot → Create Module
or right click the appropriate slot and select Create.



2. Select the communication card.

Specify the chassis slot number.

Click OK

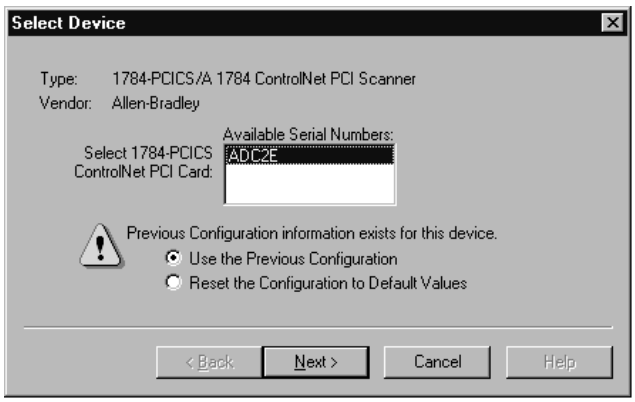


3. Select the communication card.

Select the serial number of the card.

If you previously configured the 1784-PCICS card that you selected by serial number, the chassis monitor remembers the configuration from the last time you used the card (whether in the same or different slot).

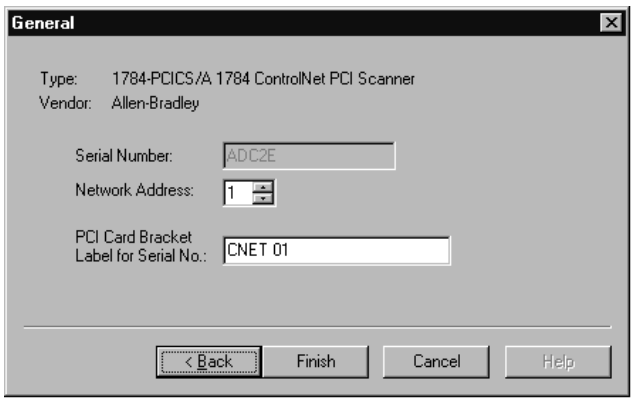
Click Next



4. Configure the communication card.

- specify the node address (MAC ID) on the ControlNet network
- enter the label name for the card (this is the name you wrote on the label of the card to help you identify the card from others in the same computer)

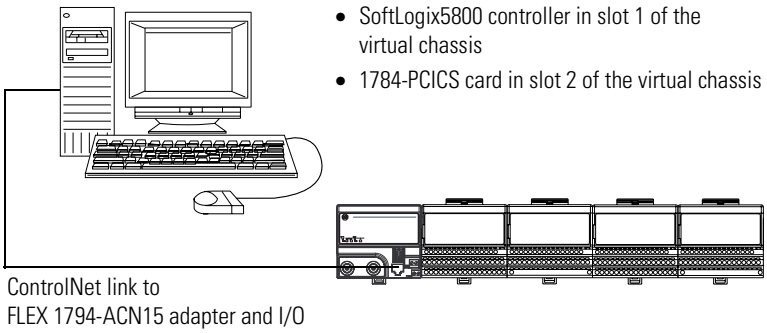
Click Finish



Creating and Downloading a Project

The following diagram illustrates the steps you follow to create and download a project. Use RSLogix 5000 programming software to perform these steps.

System setup for this quick start:



- SoftLogix5800 controller in slot 1 of the virtual chassis
- 1784-PCICS card in slot 2 of the virtual chassis

ControlNet link to
FLEX 1794-ACN15 adapter and I/O

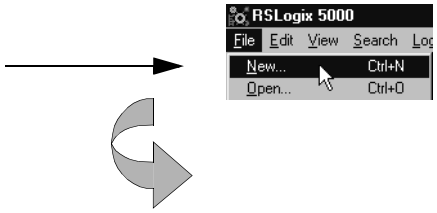
You need:

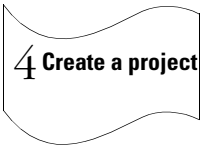
- SoftLogix chassis monitor
- RSLogix5000 programming software
- RSLinx communication software
- RSNetWorx for ControlNet software

If you don't have this hardware, you can still follow these steps. Substitute the I/O modules you have for the ones listed here and make the appropriate changes.

Creating a project

1. **Select File → New.**





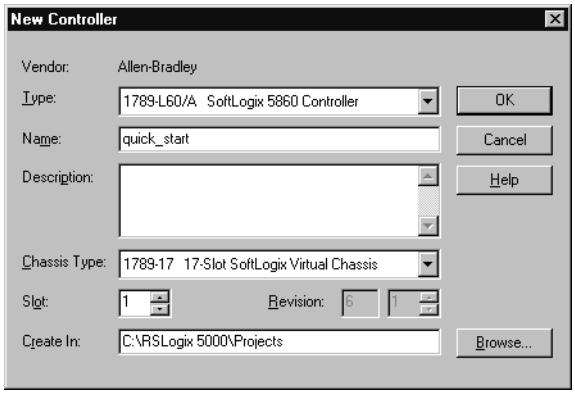
2. **Define the project.** The software uses the project name you enter with an .ACD extension to store your project.

Select a controller type.
Regardless of the SoftLogix product activation you have, select 1789-L60/A for the type.

Name the project.

Describe the project (optional).

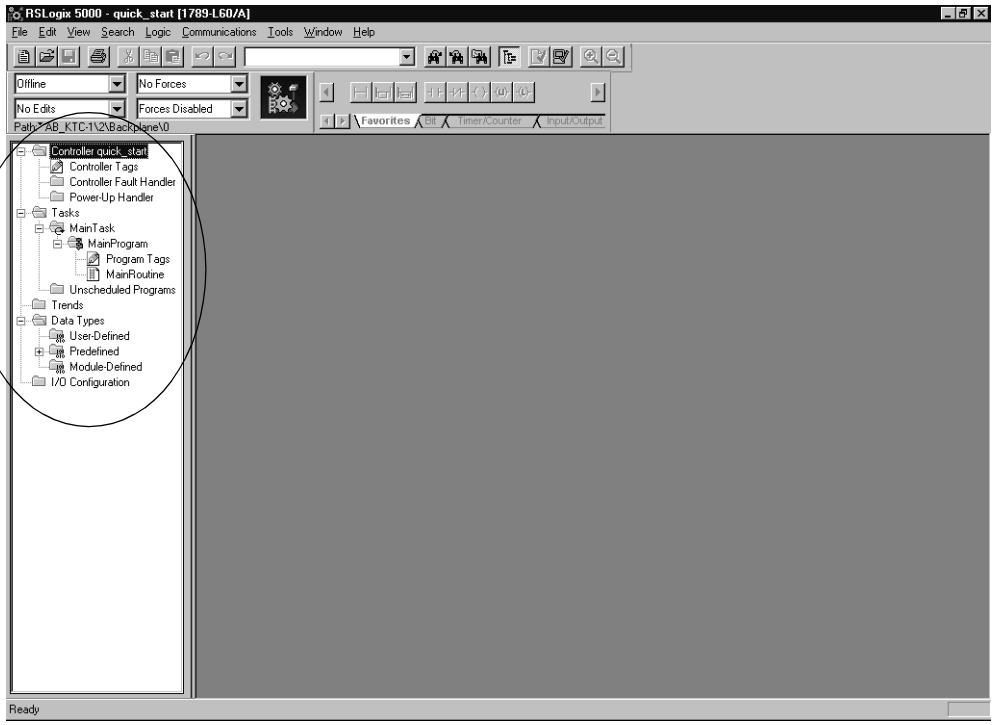
Select where to store the project (typically use the default directory).



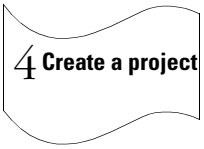
Click OK.

The software displays:

controller organizer



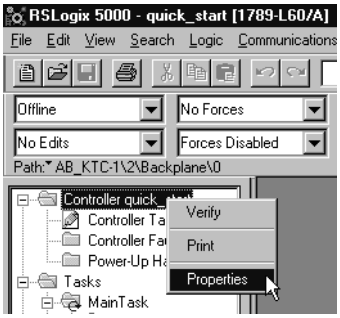
Changing project properties



1. View properties for Controller quick_start.

A. Place the cursor over the Controller quick_start folder.

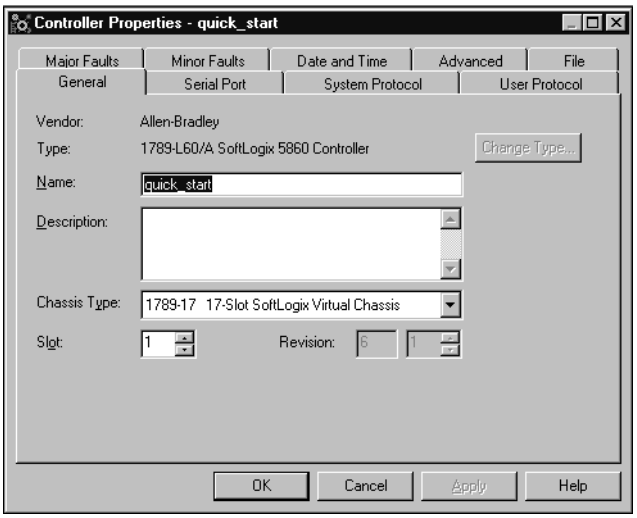
B. Click the right mouse button and select Properties.



2. View the General tab.

The screen defaults to the General tab.

Verify that the controller settings are correct. Make changes if necessary.

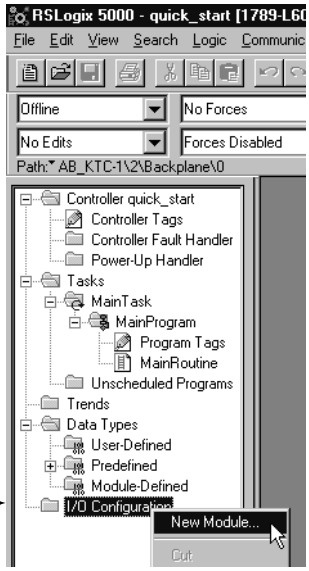


Click OK.

Adding a 1784-PCICS communication card to the project

5 **Configure I/O modules**

1. Create a new module.

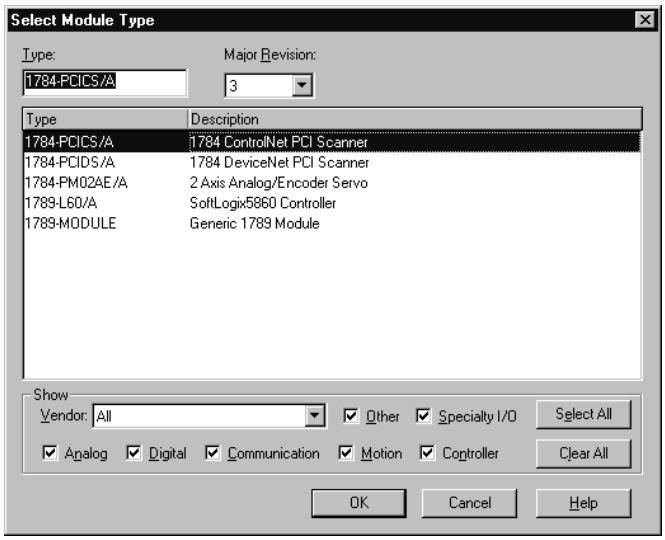


A. Place the cursor over the I/O Configuration folder.

B. Click the right mouse button and select New Module.

2. Select the 1784-PCICS card.

Select 1784-PCICS.



Click OK.

continued

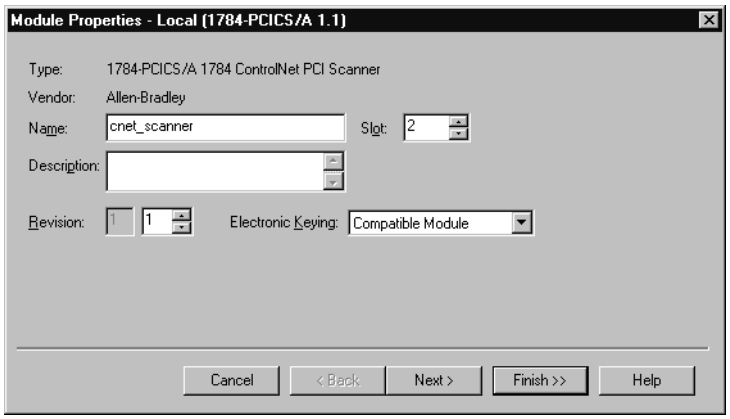
Adding a 1784-PCICS communication card to the project (*continued*)

3. Identify the communication card.

You should enter a name.
Verify the slot number.

Describe the device (optional).

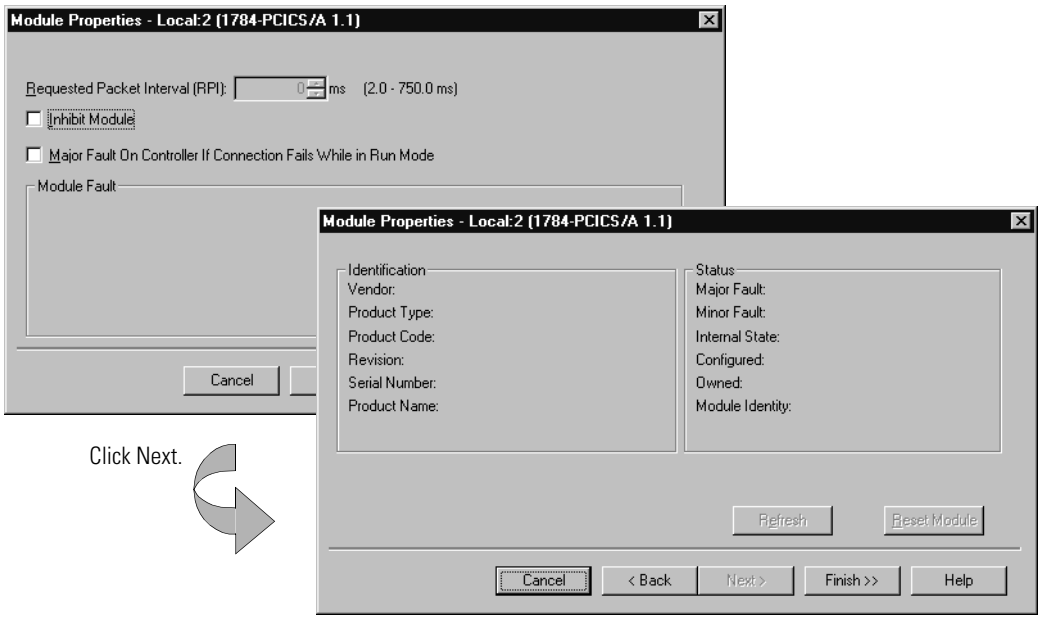
Specify electronic keying.



Click Next.

4. Use the Create wizard to configure the communication card.

Use default values for this example. If you do not want to go through each screen in the Create wizard, click Finish to create the card using default values.

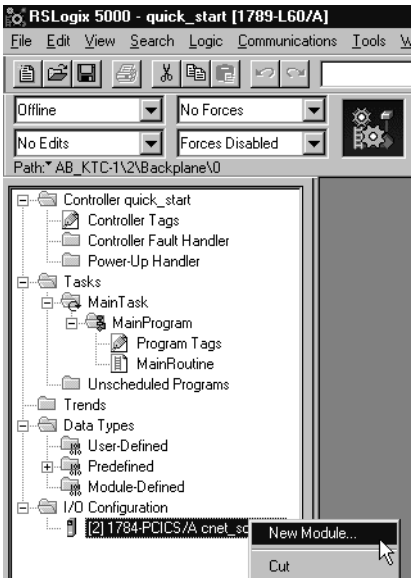


Click Next.

Click Finish.

Adding an I/O adapter to the project

1. Create a new module.



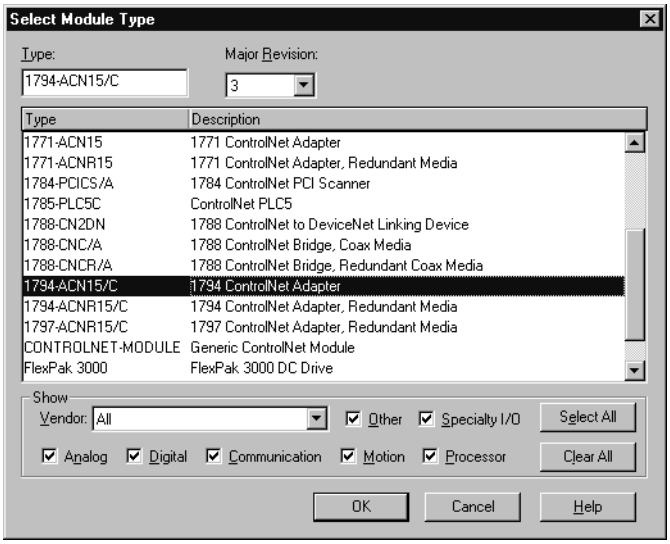
5 **Configure I/O modules**

A. Place the cursor over the local 1784-PCICS card.

B. Click the right mouse button and select New Module.

2. Select an I/O adapter.

Select the 1794-ACN15 FLEX adapter.



Click OK.

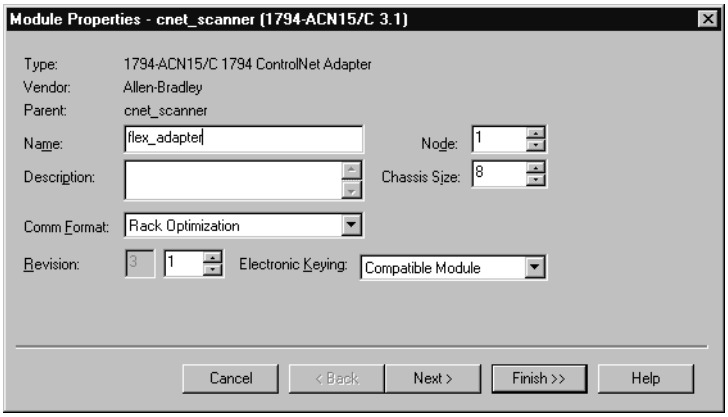
continued

Adding an I/O adapter to the project (*continued*)

3. Identify the I/O adapter.

5 **Configure I/O modules**

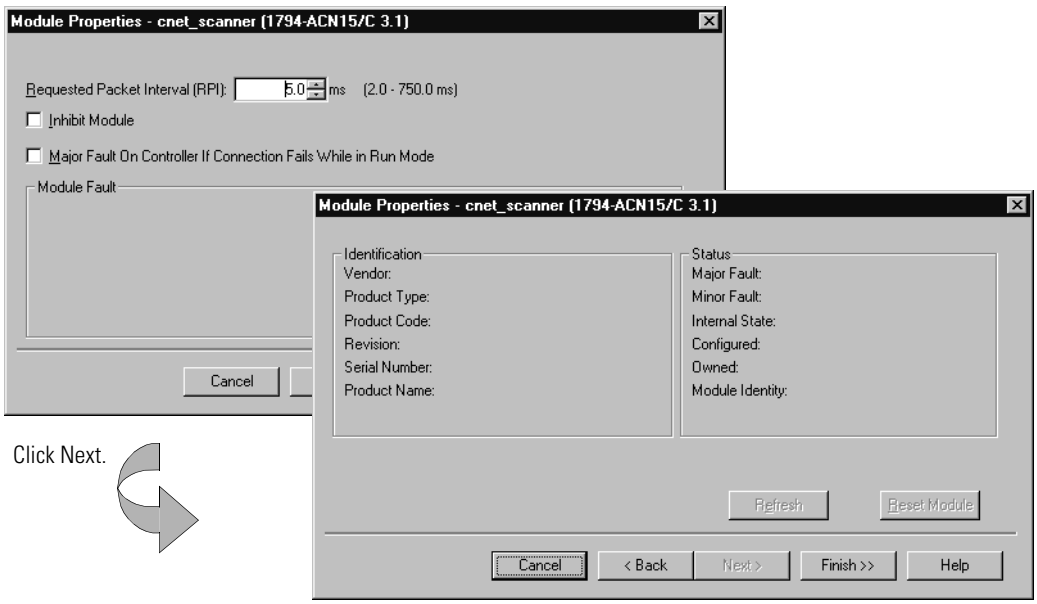
You should enter a name.
Verify node number and chassis size.
Describe the device (optional).
Select the communication format.
Specify electronic keying.



Click Next.

4. Use the Create wizard to configure the output module.

Use default values for this example.
If you do not want to go through each screen in the Create wizard, click Finish to create the module using default values.



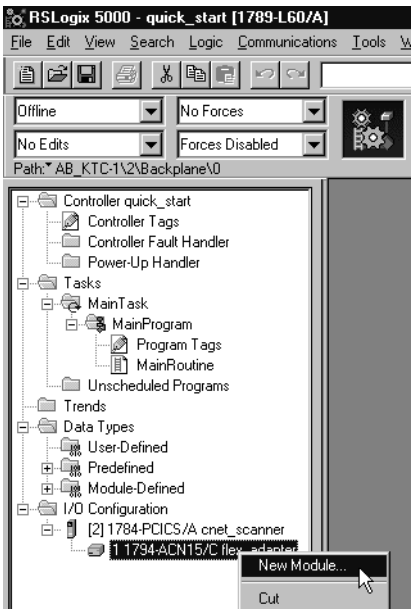
Click Next.

Click Finish.

Adding an I/O module to the project

5 **Configure I/O modules**

1. **Create a new module.**

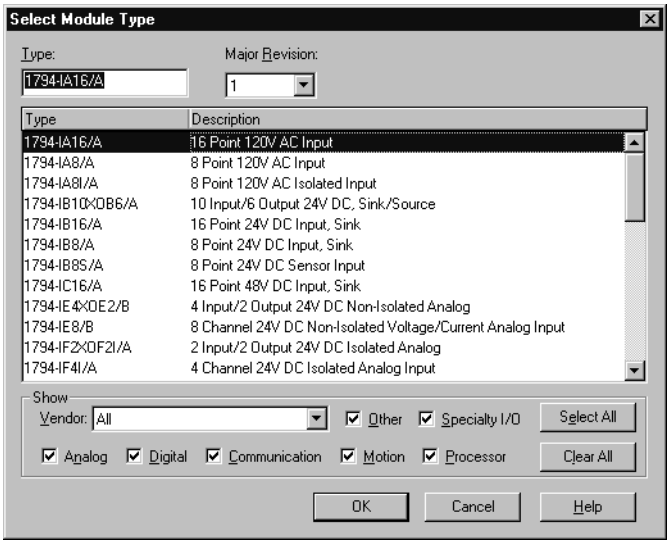


A. Place the cursor over the remote 1794-ACN15 adapter.

B. Click the right mouse button and select New Module.

2. **Select an I/O module to add.**

Select a catalog number.
For this quick start example, select 1794-IA16.

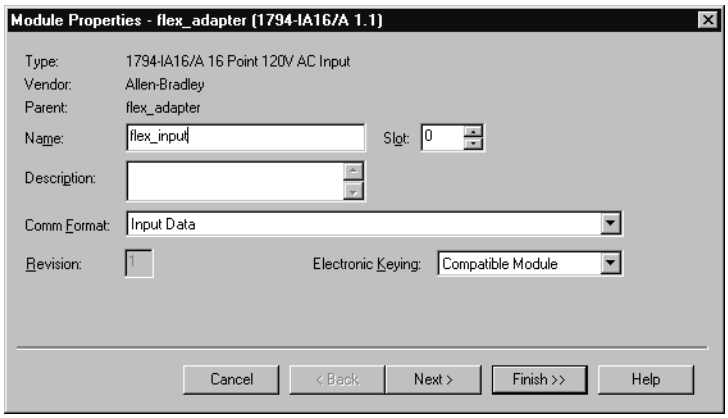


Click OK.

continued

Adding an I/O module to the project (*continued*)

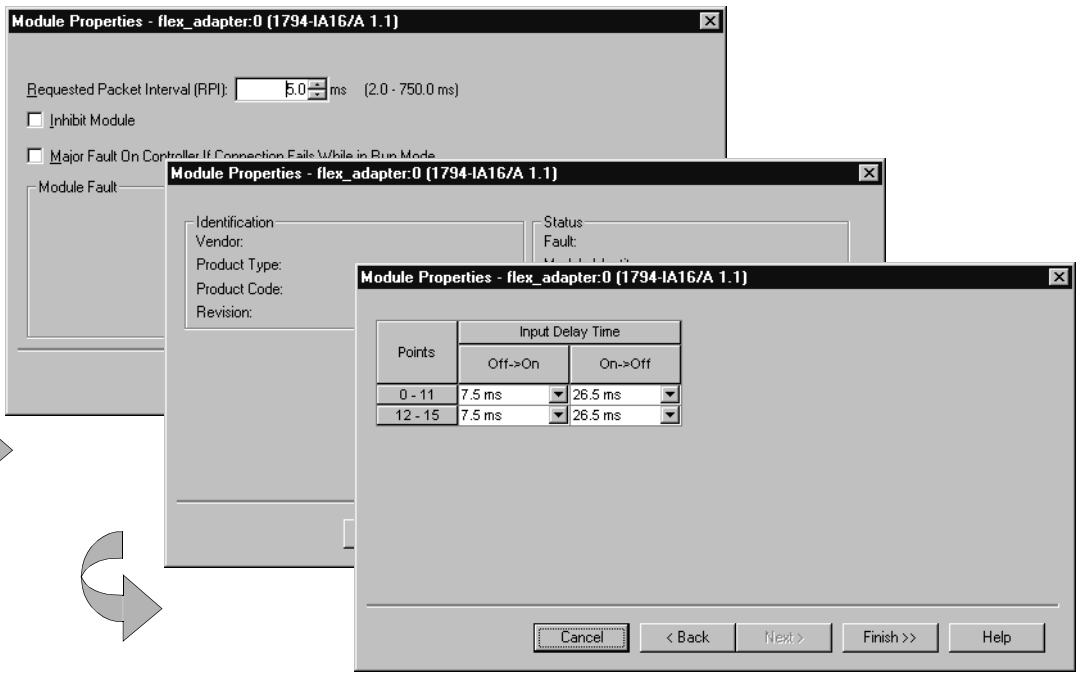
3. Identify the I/O module. These screens are specific to the 1794-IA16 input module.



You should enter a name.
Verify the slot number.
Describe the device (optional).
Select the communication format.
Specify electronic keying.

Click Next.

4. Use the Create wizard to configure the input module. Use default values for this example.
If you do not want to go through each screen in the Create wizard, click Finish to create the module using default values.

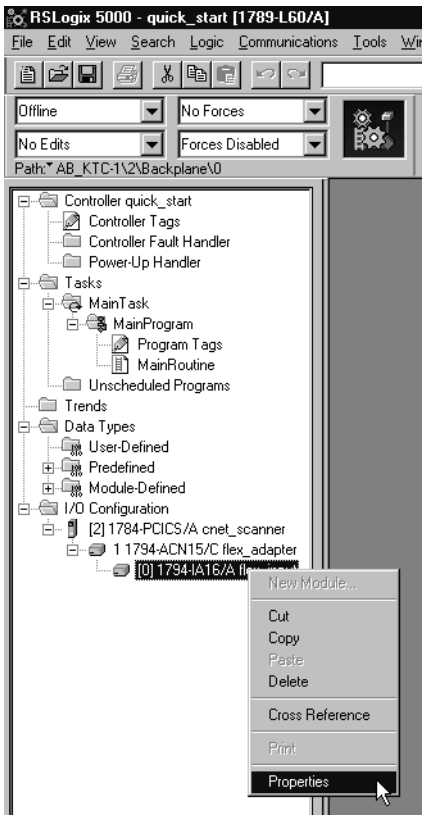


Click Next.

Click Finish.

Changing module properties

1. View properties for the module.



5 **Configure I/O modules**

A. Select a catalog number.
For this quick start example, select 1794-IA16.

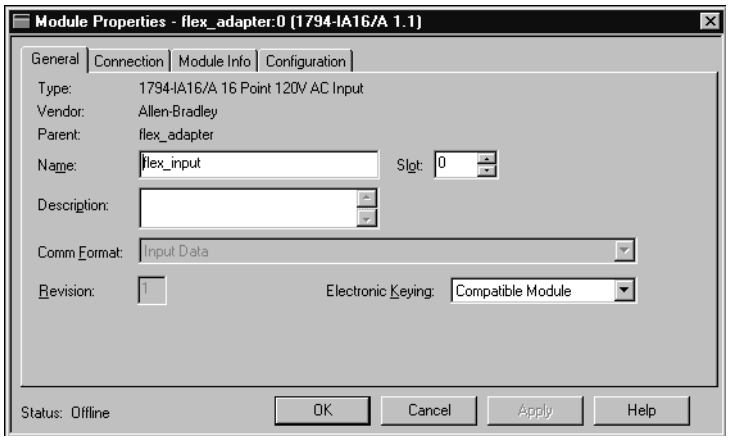
B. Click the right mouse button and select Properties.

2. View the General tab.

The screen defaults to the General tab.

Verify that the module settings are correct. Make changes if necessary.

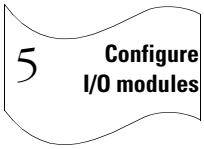
Click OK.



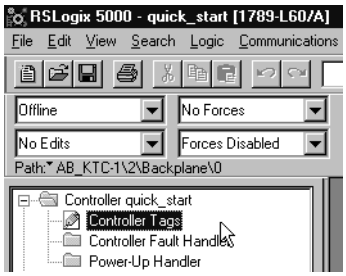
The tabs that appear depend on the type of module.

Viewing I/O tags

1. View the tags for the controller.



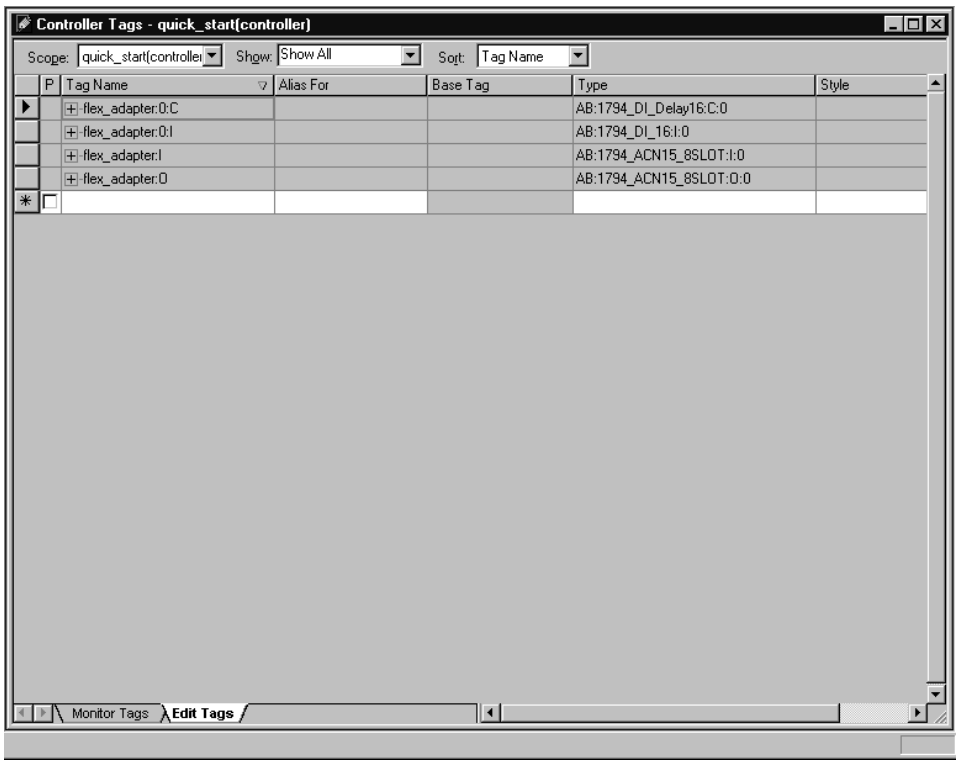
Place the cursor on the Controller Tags folder and double-click.



The software displays the module-defined tags for the I/O modules you created.

The 1794-IA16 input module is in slot 0 of the remote rail.

The remote 1794-ACN15 FLEX adapter.

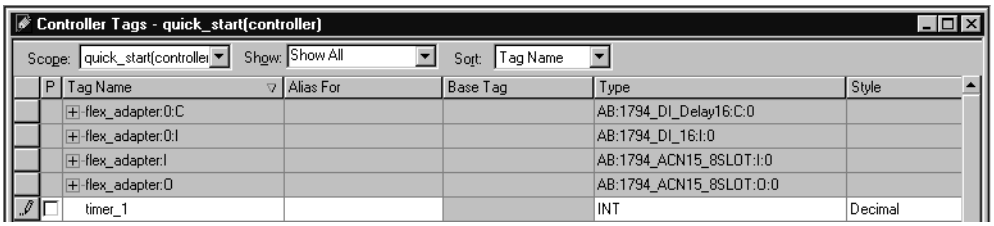


Click the Edit Tags tab.

Creating other tags

6 **Create tags**

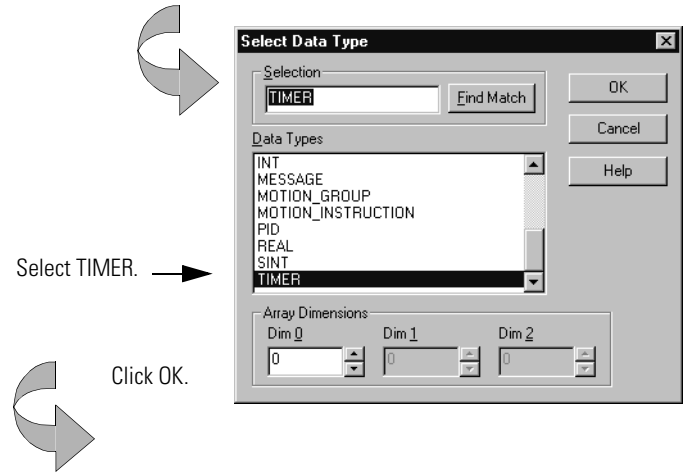
1. Create a tag.



Enter the name of the new tag.

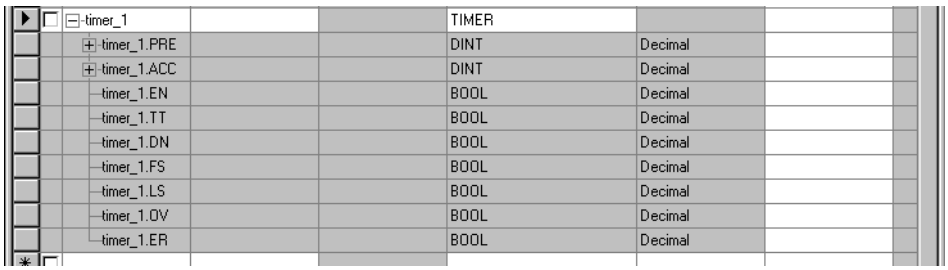
Tab to this column and select the data type.

2. Select the data type.



The software displays the tag.

Click + to display the members of the TIMER structure.



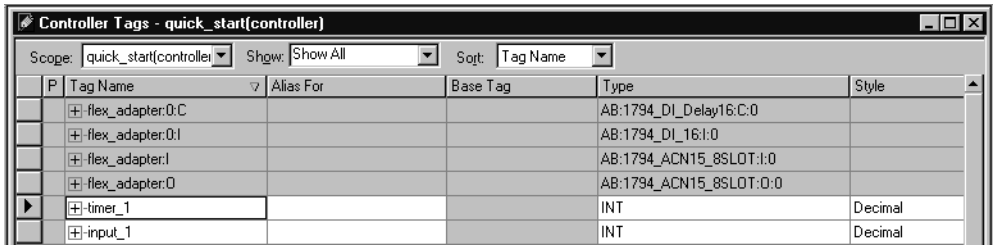
You might have to resize the column to see the tag extensions.

continued

Documenting I/O with alias tags

6 **Create tags**

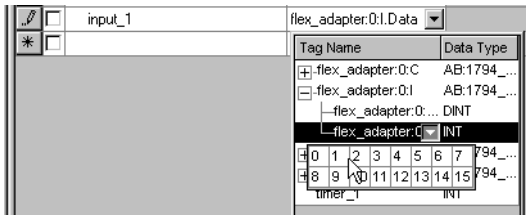
1. **Create an alias tag *input_1* for flex_adapter:0:I.Data.1.**



Enter the name of the tag.

Tab here or click in the box.
Click here to select tag to reference.

2. **Select an input data word.**



Click here to display a grid of bits.

Entering logic

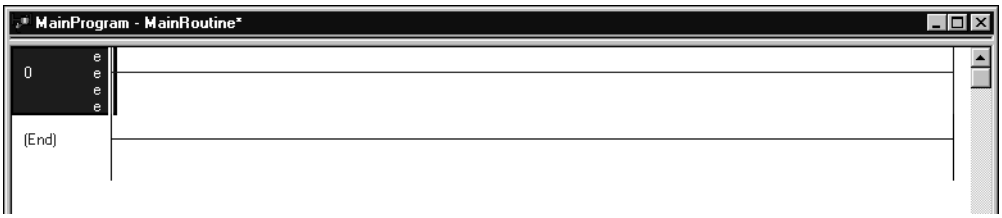
7 **Enter logic**

1. Use default task, program, and routine.

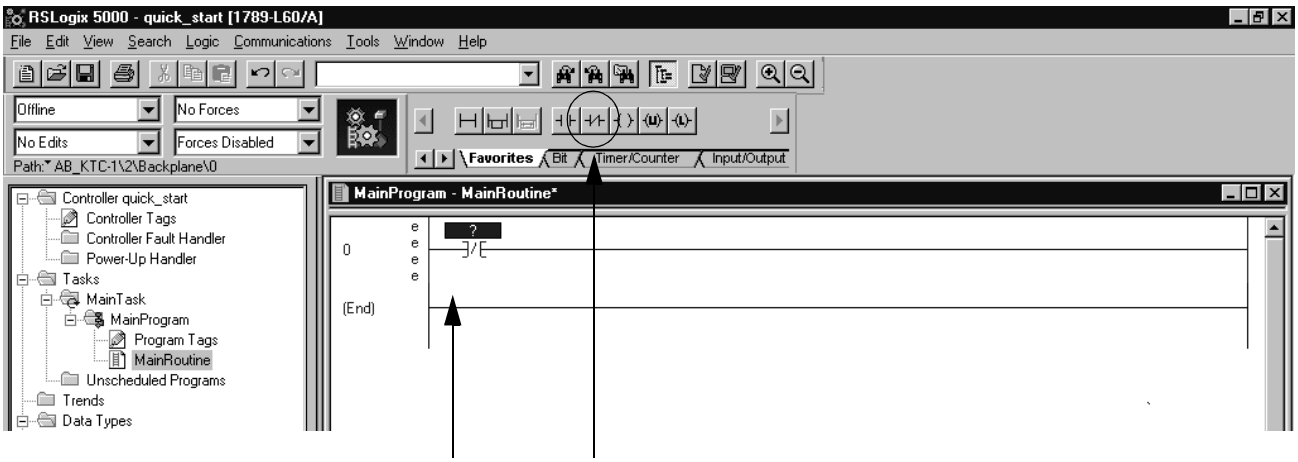
When you created the project, the software automatically created a MainTask, MainProgram, and MainRoutine. Use these defaults for this example.

Double-click MainRoutine. →

The software displays an empty routine.



2. Enter an XIO instruction.



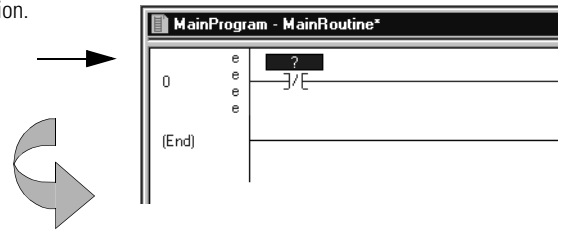
Drag and drop the XIO instruction on an empty rung.

Entering logic (*continued*)

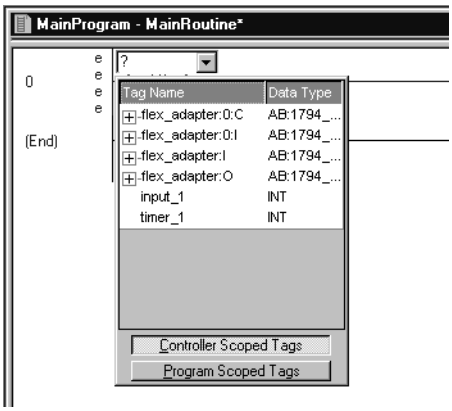
7 **Enter logic**

3. Assign a tag to the XIO instruction.

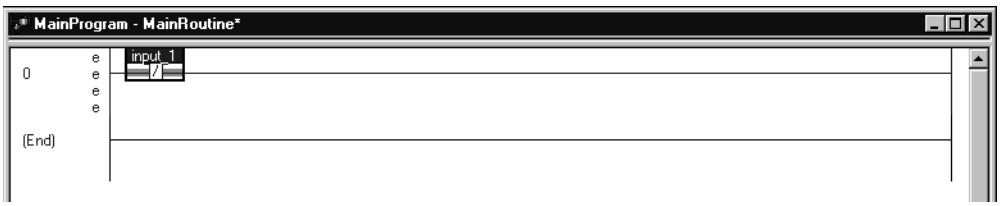
Double-click the tag area of the instruction.



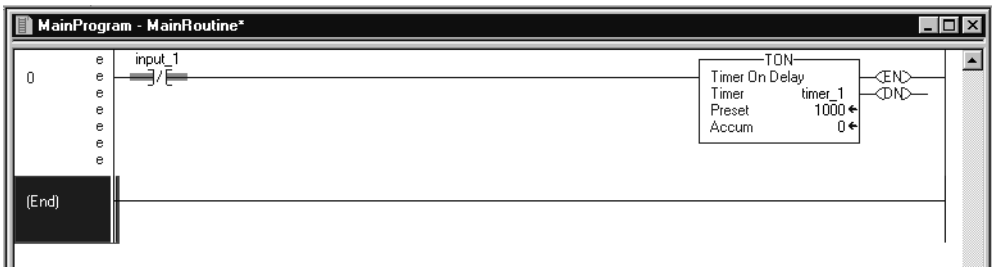
Use the drop-down menu to select *input_1*.



The software displays an incomplete rung.



4. Enter this logic.



5. To save the project, from the File menu, select Save.

Downloading a project

8 **Download a project**

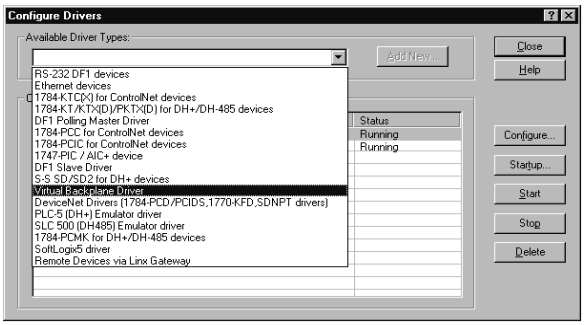
1. Use RSLinx software to install the virtual backplane driver:

A. In RSLinx software, select Communication → Configure Driver.

B. From the Available Driver Types list, select "Virtual Backplane Driver".

There are no configuration settings to select. The chassis monitor uses this driver for the controller to communicate with RSLinx software.

C. Click Add New.



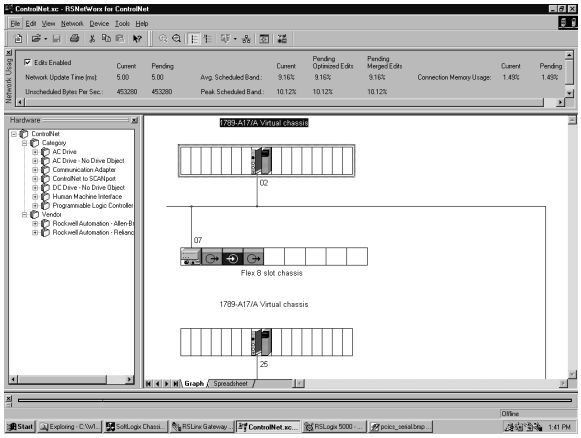
2. Make sure the 1784-PCICS communication card is connected to the ControlNet link.

A. In RSNetWorx software, go online through the 1784-PCICS card in the virtual chassis.

B. Enable edits and survey the network.

C. Specify the network update time (NUT).

D. Save and re-write the schedule for all connections.



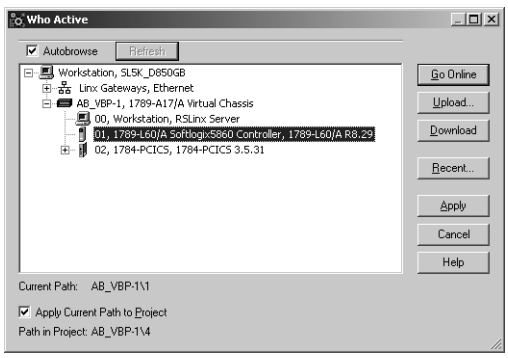
3. Download the project from the Communications menu.

A. In RSLogix5000 software, select Communication → "Who Active".

B. Select the SoftLogix controller in slot 1 of the virtual backplane.

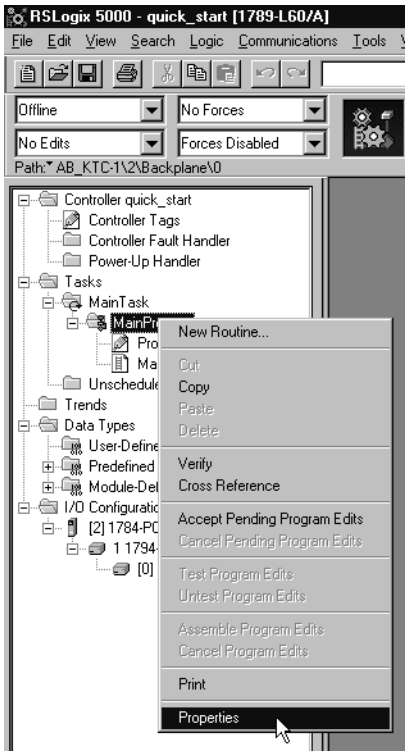
C. Click Download. Confirm the download when prompted.

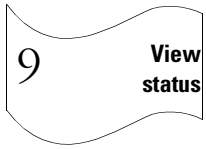
D. Apply the path to the project.



Viewing program scan time

1. **View properties for the MainProgram.**

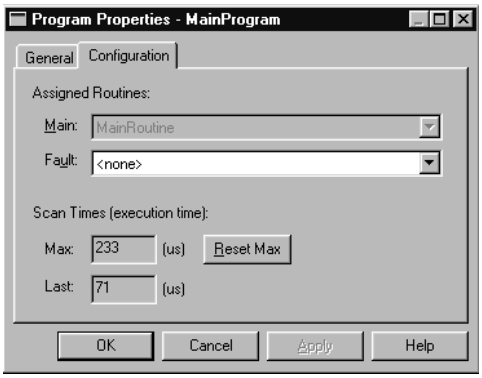




A. Place the cursor over the MainProgram folder.

B. Click the right mouse button and select Properties.

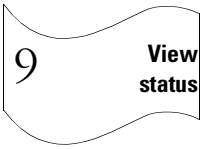
2. **Select the Configuration tab.**



The Configuration tab displays the maximum and last scan times for the program.

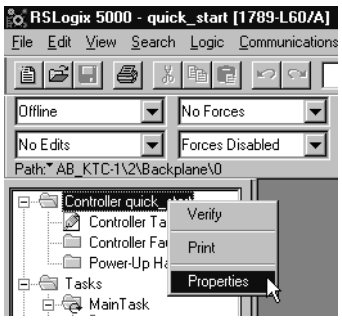
Viewing controller memory usage

1. View properties for Controller quick_start.



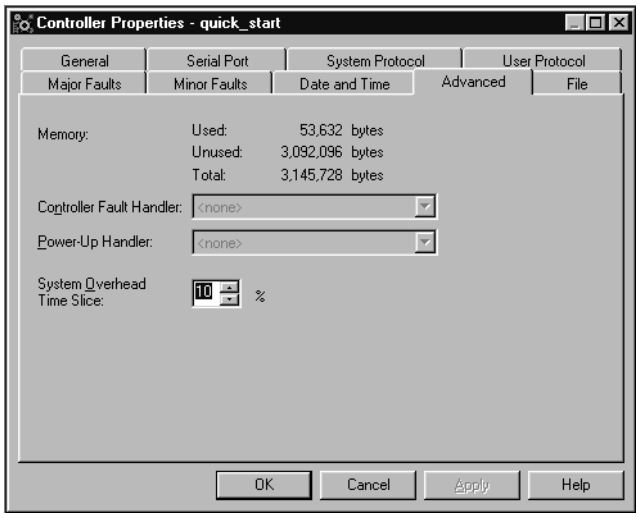
A. Place the cursor over the Controller quick_start folder.

B. Click the right mouse button and select Properties.



2. Select the Advanced tab.

In addition to other information, the Advanced tab displays controller memory usage.



What To Do Next

Once your controller is created and operating, you can use RSLogix5000 programming software to develop and test your control application.

Use the remaining chapters in this manual as reference material for how the SoftLogix controller operates in the Logix environment.

Notes:

What Is SoftLogix?

Using This Chapter

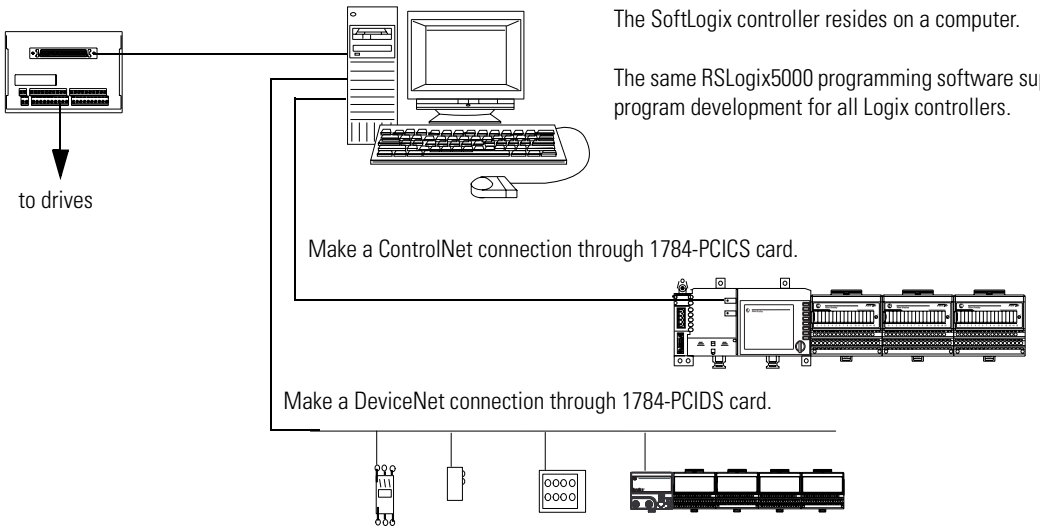
The SoftLogix controller is part of the Logix environment. The SoftLogix controller is a software-based controller that supports the Logix instructions, including the motion instructions. A SoftLogix system can consist of these components:

- RSLogix 5000 programming software that supports every Logix controller.
- 1784-PM02AE motion card that provides integrated motion control
- 1784-PCICS communication card that provides communication and I/O control over a ControlNet network

  or a 1784-PCIC communication card that provides communication over a ControlNet network
- 1784-PCIDS communication card that provides communication and I/O control over a DeviceNet network.

Motion support is an integral part of the SoftLogix controller.
RSLogix 5000 programming software provides all motion support.

The 1784-PM02AE motion card connects to the termination panel.



The SoftLogix controller also supports a software-based, 1789-SIM module that you can use to simulate I/O points for your application. See chapter 2, Configuring and Using Simulated I/O.

Select the SoftLogix5800 product that best fits your application:

If you need (maximum):	Use this controller:	Available slots:
1 SoftLogix5800 controller[1] 1 PCI communication card no motion support	1789-L10	2-slot virtual chassis
2 SoftLogix5800 controllers 5 PCI communication cards[2] 2 analog motion cards (4 axis maximum)	1789-L30	5-slot virtual chassis
6 SoftLogix5800 controllers unlimited PCI communication cards[2] 4 analog motion cards (8 axis maximum)	1789-L60	16-slot virtual chassis

(1) There is a maximum 2048K bytes of controller memory on the 1789-L10 product.

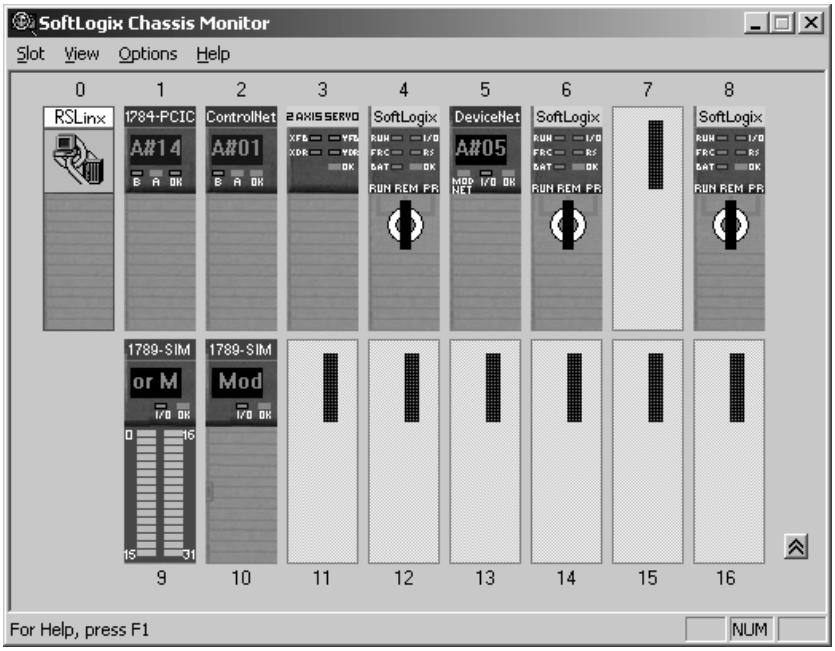
(2) The number of available slots in the virtual chassis is limited by activation level. You can have as many PCI communication cards as you have available slots.

IMPORTANT The same SoftLogix5800 controller is supplied in all of the above products. Regardless of the product you have, select 1789-L60/A in RSLogix5000 software when you specify a controller type.

For information about:	See page
using the chassis monitor	2-3
developing programs	2-6
how the SoftLogix system uses connections	2-9
selecting a system overhead percentage	2-14

Using the Chassis Monitor

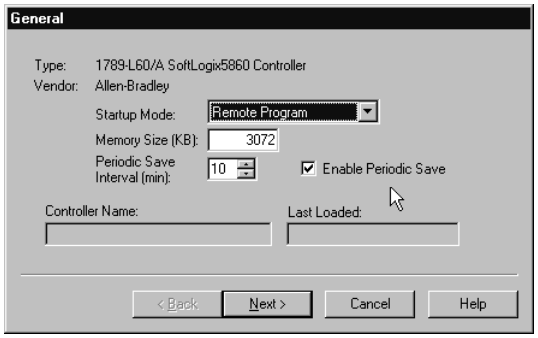
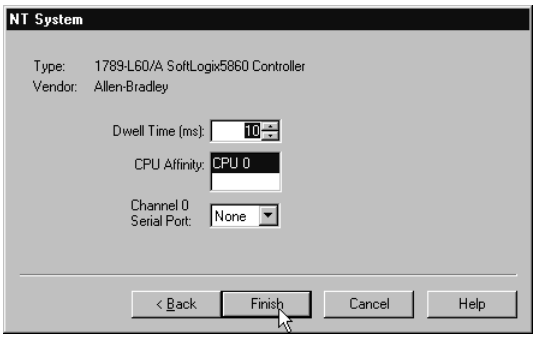
The SoftLogix chassis monitor is your window into the SoftLogix system so you can monitor the system components. The chassis monitor models a chassis. You install virtual devices in the virtual chassis to represent the controller and cards in your system.



The chassis monitor is your interface to the SoftLogix controller. Use the monitor to:

- add and configure controllers
- add and configure communication cards
- add and configure motion cards
- change processor mode
- monitor controller and associated module status
- monitor motion performance

When you install a controller, the chassis monitor lets you configure specific characteristics about the controller:

On this configuration dialog box:	Specify these characteristics:	
	Startup Mode	Specify how the controller should behave when its service is started. Select Remote Program (default) or Last Controller State
	Memory Size	Specify the memory size (KBytes) to allow for the controller. The limit is the amount of RAM in your computer. The default is 3072 KBytes. See the information on the next page about determining an appropriate memory size.
	Periodic Save Interval	Specify whether you want to save the current controller information (program, data, and configuration information) periodically, and if so, specify how often (minutes). The default is enabled for 10 minutes.
	Dwell Time (ms)	Specify the timeslice (0-1000 ms) made available for all other Windows applications. The default is 10 ms. The dwell time is the time between the end of the continuous task and the start of the next execution of the continuous task.
	CPU Affinity	If your computer has multiple CPUs, select which CPU to use for this controller. The default is CPU 0.
	Channel 0 Serial Port	Select which COM port to use for serial communications. Select COM1, COM2, COM3, or COM4. The default is none.

Determining a memory size

IMPORTANT The memory size you specify is the amount of RAM in your computer that you want to allocate to the SoftLogix controller. This allocated RAM is not available to Windows NT or any other application.

The following equations provide an estimate of the memory needed for a controller. Each of these numbers includes a rough estimate of the associated user programming. Depending on the complexity of your application, you might need additional memory.

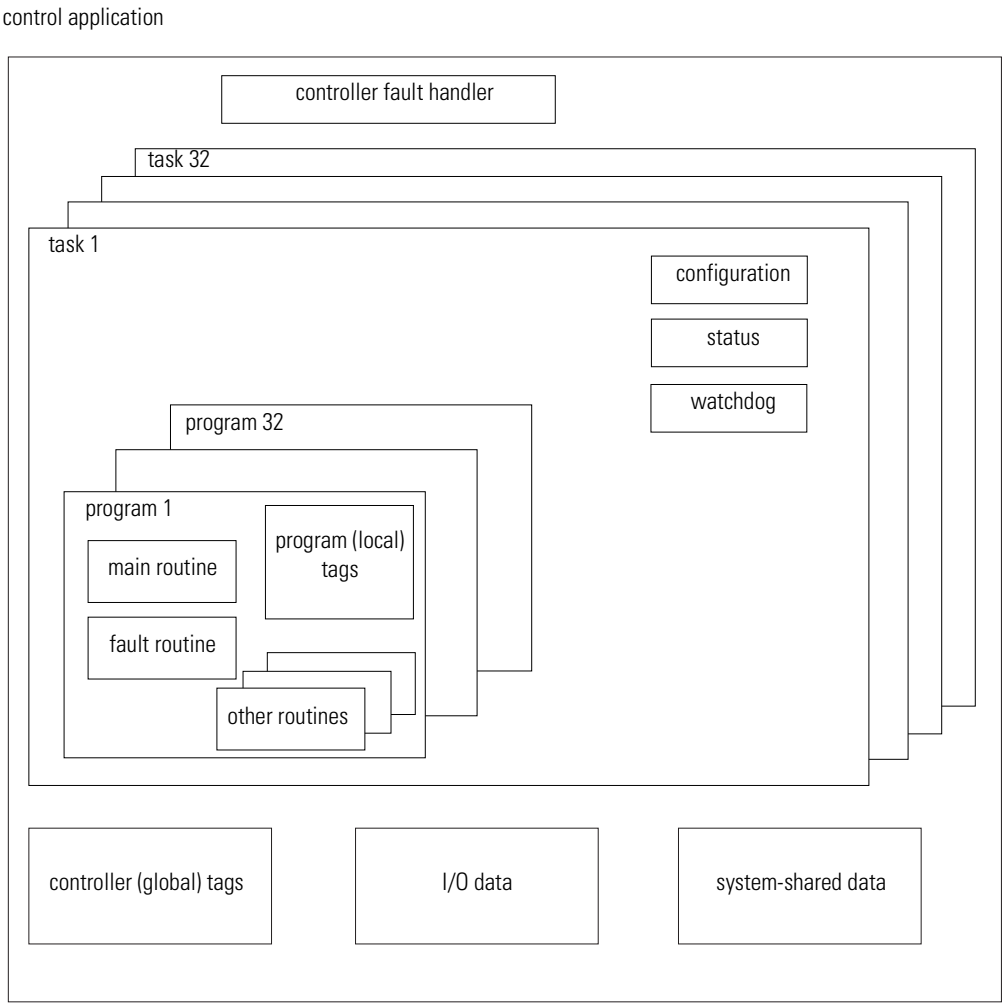
Controller tasks	_____	* 4000 =	_____	bytes (minimum 1 needed)
Discrete I/O points	_____	* 400 =	_____	bytes
Analog I/O points	_____	* 2600 =	_____	bytes
Communication modules	_____	* 2000 =	_____	bytes
Motion axis	_____	* 8000 =	_____	bytes
		Total =	_____	bytes

If you want to change the amount of memory you specified for a controller, you must first remove the controller from the SoftLogix chassis monitor. Then re-install the controller and specify the new memory size.

Developing Programs

The controller's execution model 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 either continuous or periodic. The SoftLogix controller supports as many as 32 tasks, only one of which can be continuous.

A task can have as many as 32 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.

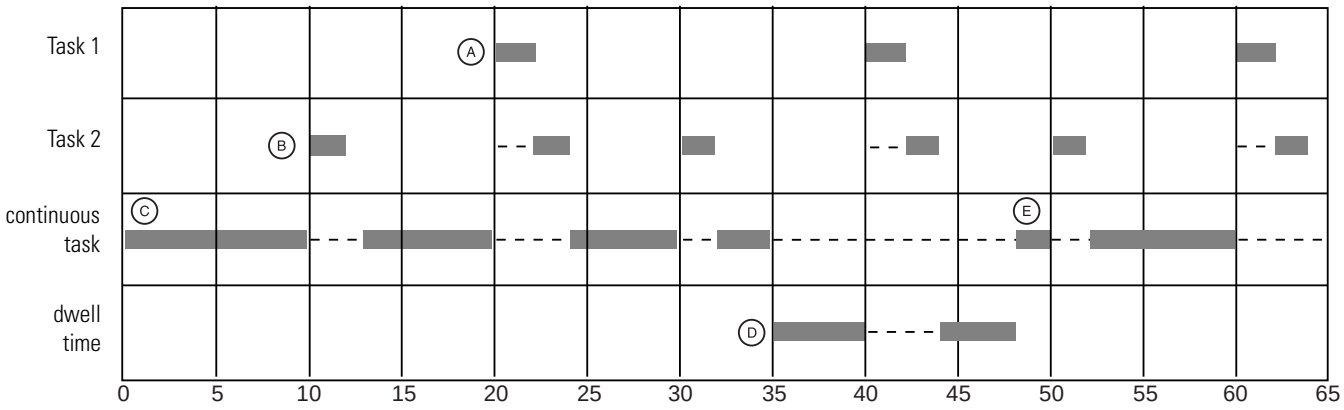
Specifying task priorities

Each task in the controller has a priority level. The controller uses the priority level to determine which task to execute when multiple tasks are triggered. There are 3 configurable priority levels for periodic tasks that range from 1-3, with 1 being the highest priority and 3 being the lowest priority. A higher priority task will interrupt any lower priority task. The continuous task has the lowest priority and is always interrupted by any periodic task.

The dwell time determines how much time to allow for other Windows programs, running at a normal priority, to execute. The dwell time is the time between the end of the continuous task and the start of the next execution of the continuous task. The dwell time is at the same priority as the continuous task. By default, the dwell time lasts 10ms.

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

Task:	Priority Level:	Task Type:	Actual Execution Time:	Worst Case Execution Time:
1	1	20ms periodic task	2ms	2ms
2	2	10ms periodic task	4ms	6ms
na	none (lowest)	continuous task	25ms	35ms
na	none	dwell time	10ms	14ms



Notes:

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

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

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

D. When the continuous task completes, the dwell time starts. The dwell time runs at the same priority as the continuous task so it is interrupted by all other tasks.

E. When the dwell time completes, the continuous tasks restarts, 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 32 programs.

The scheduled programs within a task execute to completion from first to last. Programs that aren't 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 processor.

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 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.

How the SoftLogix System Uses Connections

The SoftLogix system uses a connection to establish a communication link between two devices. Connections take many forms:

- tags for I/O and communication modules
- produced and consumed tags
- message (MSG) instructions

You indirectly determine the number of connections that the controller requires by how you configure the controller to communicate with other devices in the system.

The SoftLogix system supports both scheduled and unscheduled connections.

Connection:	Description:
scheduled	A scheduled connection identifies a specific device and lets you send and receive data repeatedly at a predetermined rate. For example, a connection to an I/O module is a schedule connection because you repeatedly receive data from the module. Other scheduled connections include connections to: • communication devices • produced/consumed tags For ControlNet connections, you must use RSNetWorx for ControlNet to enable all scheduled connections and establish a network update time (NUT) for the network. For DeviceNet connections, you must use RSNetWorx for DeviceNet to define the scan list and map the I/O data for each DeviceNet device.
unscheduled	An unscheduled connection is a message transfer between controllers that is triggered by the program (i.e., the MSG instruction). Unscheduled messaging lets you send and receive data when needed.

Each SoftLogix controller supports 250 connections, which can be any combination of scheduled or unscheduled.

Determining scheduled connections for I/O modules

The SoftLogix system uses direct connections and rack-optimized connections to transmit I/O data.

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.
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 DIN rail. Rather than having individual, direct connections for each I/O module, there is one connection for the entire DIN rail.

IMPORTANT The limit of scheduled connections for I/O depends on the communication device the controller uses to control the I/O. Each 1784-PCICS communication card supports 127 connections.

To conserve the number of connections that are available, place digital I/O modules together in the same location and use a rack-optimized connection. To select a rack-optimized connection, select a "rack-optimized" option for the communication format when you add the communication device and I/O modules to the controller project in RSLogix 5000 programming software.

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

Determining unscheduled connections for messages

Unscheduled connections are used for transferring data to other devices, such as other controllers or operator devices. The 1784-PCICS card supports 128 total connections, 127 of which can be scheduled connections, as described above. The remaining connections (or all 128, if you have no scheduled connections) can be used for unscheduled connections.

Some messages use an unscheduled connection to send or receive data. Some messages also have the option of leaving the connection open (cache) or closing the connection when the message is done transmitting. The following table shows which messages use a connection and whether or not you can cache the connection:

This type of message:	Using this communication method:	Uses a connection:	Which you can cache:
CIP data table read or write	CIP	✓	✓
PLC2, PLC3, PLC5, or SLC (all types)	CIP		
	CIP with Source ID		
	DH+	✓	
CIP generic	N/A		
block-transfer read or write	N/A	✓	✓

Use the following table to select a cache option for a message.

If the message executes:	Then:	Because:
repeatedly	Select the Cache Connections check box	This will keep the connection open and optimize execution time. Opening a connection each time the message executes increases execution time.
infrequently	Clear the Cache Connections check box	This will close the connection upon completion, which frees up that connection for other uses.

You can cache as many as 16 messages (a combination of any type, not including block-transfer) at one time. If you try to cache more than 16, the controller determines the 16 most-currently used messages and caches those. If there are 16 messages cached, and a message is triggered that is currently not cached, the controller drops the connection of the oldest-cached message to make room for the new message.

In addition to 16 cached messages, you can also cache as many as 16 block-transfer messages. The same conditions apply to caching block-transfer messages as described above for caching other types of messages.

Determining scheduled connections for produced/consumed tags

The SoftLogix controller supports the ability to produce (broadcast) and consume (receive) system-shared tags. System-shared data is accessible by multiple controllers over a ControlNet network. Produced and consumed tags each require scheduled connections.

This type of tag:	Requires these connections:
produced	A produced tag allows other controllers to consume the tag, which means that multiple controllers can simultaneously receive the same tag data. The local controller (producing) must have one connection for the produced tag and the first consumer and one more connection for each additional consumer (heartbeat). The default produced tag specifies that two controllers can consume the tag, but you can change the number of consumers. As you increase the number of controllers that can consume a produced tag, you also reduce the number of connections the controller has available for other operations, like communications and I/O.
consumed	Each consumed tag requires one connection for the controller that is consuming the tag.

Determining total connection requirements

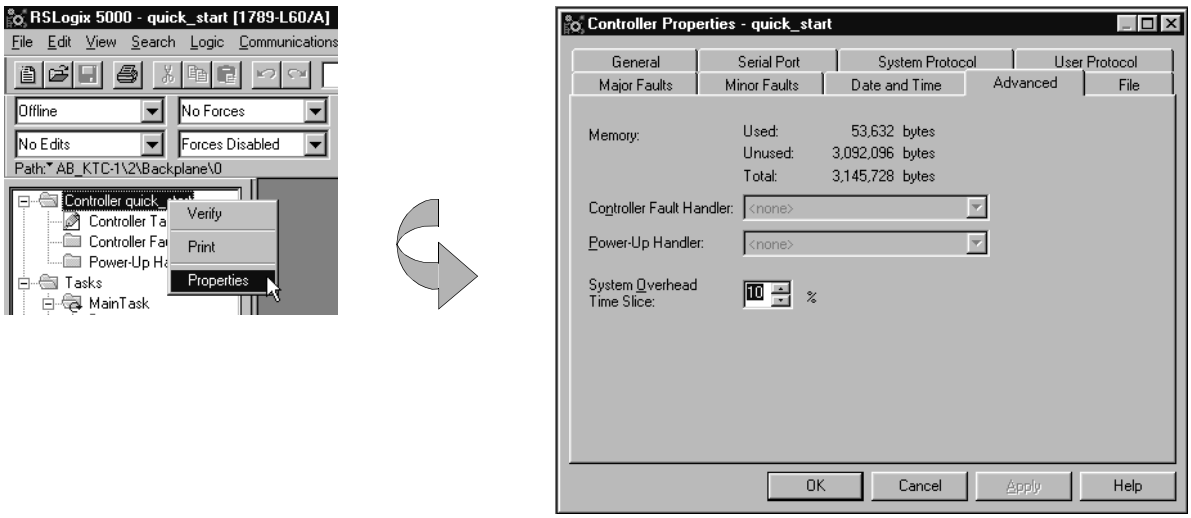
Each SoftLogix controller supports 250 connections. Use the following table to tally connection requirements for a controller:

Connection Type:	**Device Quantity:**	**Connections per Device:**	**Total Connections:**
I/O device (direct connection over a ControlNet link)		1	
1784-PM02AE motion card (4 devices maximum per computer)		3	
local 1784-PCIC, -PCICS communication card		0	0
remote ControlNet communication device (such as a 1794-ACN15, -ACNR15 or 1756-CNB module) configured as a direct (none) connection configured as a rack-optimized connection		0 **or** 1	
remote 1785 ControlNet PLC-5 controller		1	
1784-PCIDS communication card		2	
produced and consumed tag produced tag and one consumer		1	
each additional consumers		1	
consumed tag		1	
block-transfer message		1	
other message		1	
		total	

Selecting a System Overhead Percentage

The Controller Properties 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.



The system overhead function interrupts the continuous task. The percentage you specify determines the amount of the continuous task to allocate to system overhead functions, which include:

- communicating with programming and HMI devices (such as RSLogix 5000 software)
- responding to messages
- sending messages, including block-transfers
- 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 a one communication device to another communication device across the virtual chassis

This function lets the controller take care of communication requests that occur from other controllers or from queued requests from within the controller's application program. If communications are not completing fast enough, increase the system overhead percentage.

Controlling Motion Devices

Using This Chapter

Configuring Your System for a Motion Card

For the SoftLogix controller to control motion applications, you need:

- a 1784-PM02AE motion card (4 per computer maximum)

  The 1784-PM02AE motion card connects to a servo drive and closes a high-speed position and velocity loop. Each 1784-PM02AE module can control up to two axes.

- a 1784-PMCSY4 synchronization cable

  If you have multiple 1784-PM02AE cards, you must link the cards with a 1784-PMCSY4 synchronization cable.

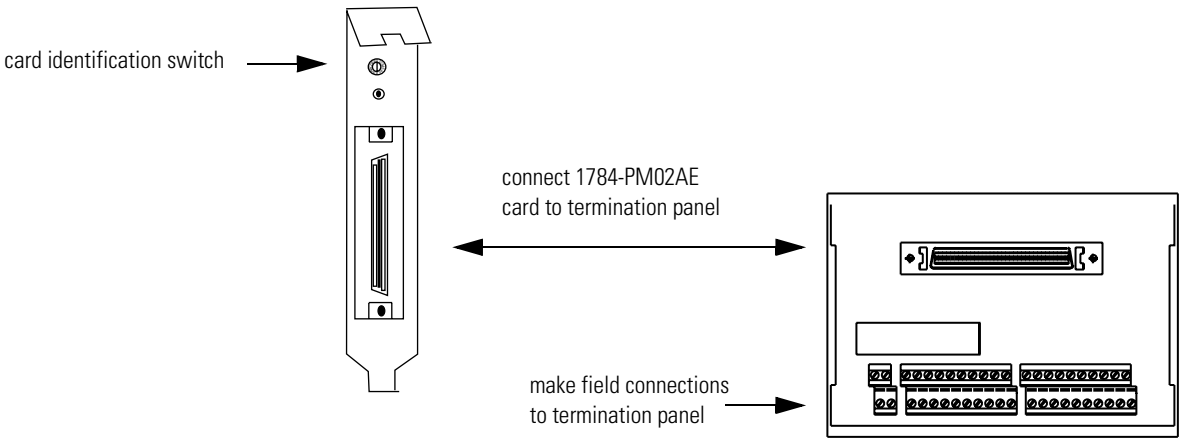
- a 1784-PM02AE-TP01 or 1784-PM02AE-TP03 termination panel

  You make all field terminations to the 1784-PM02AE card using the 1784-PM02AE-TP01 or 1784-PM02AE-TP03 termination panel and associated cable.

- RSLogix5000 programming software to configure the motion card and its associated axes (2 per card)

Step 1: Install the hardware

Make sure the 1784-PM02AE motion card is properly installed in a 32-bit, primary PCI slot in the computer.



- Use the card identification switch to uniquely identify each 1784-PM02AE motion card in your computer. The card identification switch is a slotted, rotary switch with 16 positions (0-9 and A-F). Use a flathead screwdriver to select a setting.

  The switch setting uniquely identifies the card from any other motion cards in your computer. The switch setting and the PCI slot where you install the card **do not** correspond to the backplane slot in the SoftLogix chassis. You use the SoftLogix chassis monitor to place the communication card in a specific backplane slot (see the next page).

- Make a label to place on the front of the card, or use a pen to write on the front of the card. The label should include the card identification switch setting and a name you can use to identify the card from any others you might install in the computer.

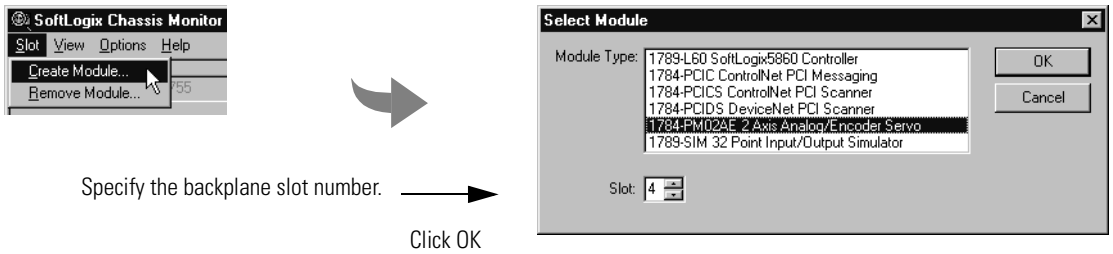
- If you have more than one motion card, use the 1784-PMCSY4 synchronization cable to connect the motion cards within the computer.

For more information about installing a 1784-PM02AE motion card, see the *PCI 2-Axis Servo Card Installation Instructions*, publication 1784-IN005.

Step 2: Create the motion card in the chassis

Before you can operate the motion card, you must create the motion card as part of the SoftLogix chassis.

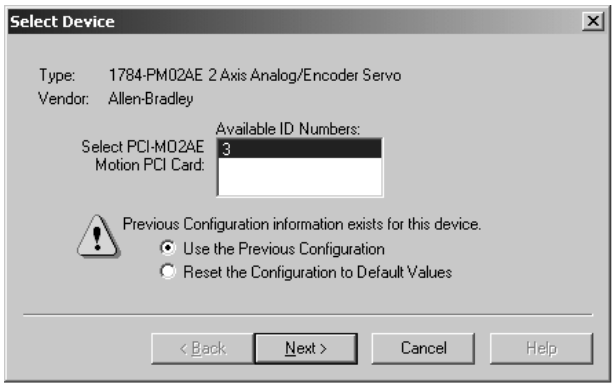
1. From the SoftLogix chassis monitor, select Slot → Create Module or right click the appropriate slot and select Create. Select the motion card.



Specify the backplane slot number.

Click OK

2. Specify which motion card to use by selecting an available ID number, which corresponds to the setting on the card identification switch.

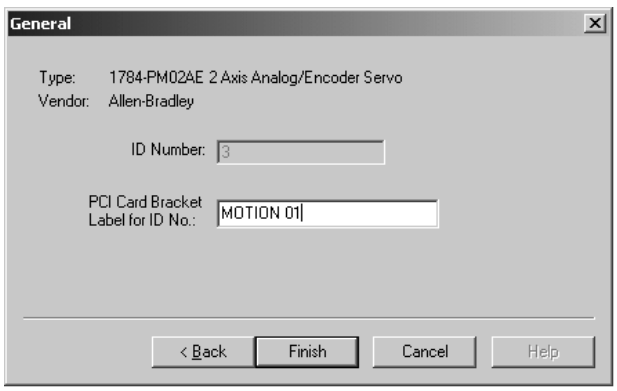


Select the ID number.

If you previously configured the 1784-PM02AE card that you selected by ID number, the chassis monitor remembers the configuration from the last time you used the card (whether in the same or different slot).

Click Next

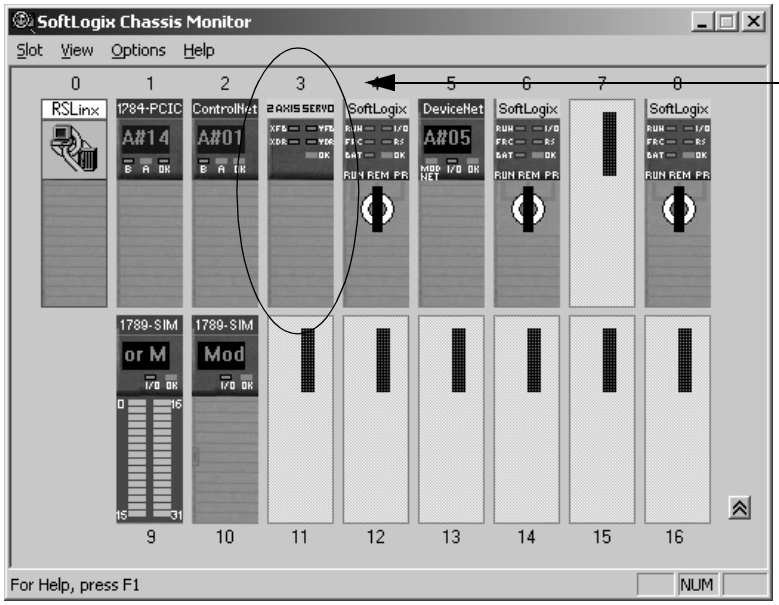
3. Enter the label name for the card (this is the name you wrote on the label of the card to help you identify the card from others in the same computer).



Click Finish

You can specify any slot number greater than 0 for the motion card. RSLinx software resides in slot 0.

The chassis monitor shows the 1784-PM02AE card as a virtual module in the SoftLogix chassis. The LEDs on the virtual monitor emulate a 1756-MO2AE motion module.



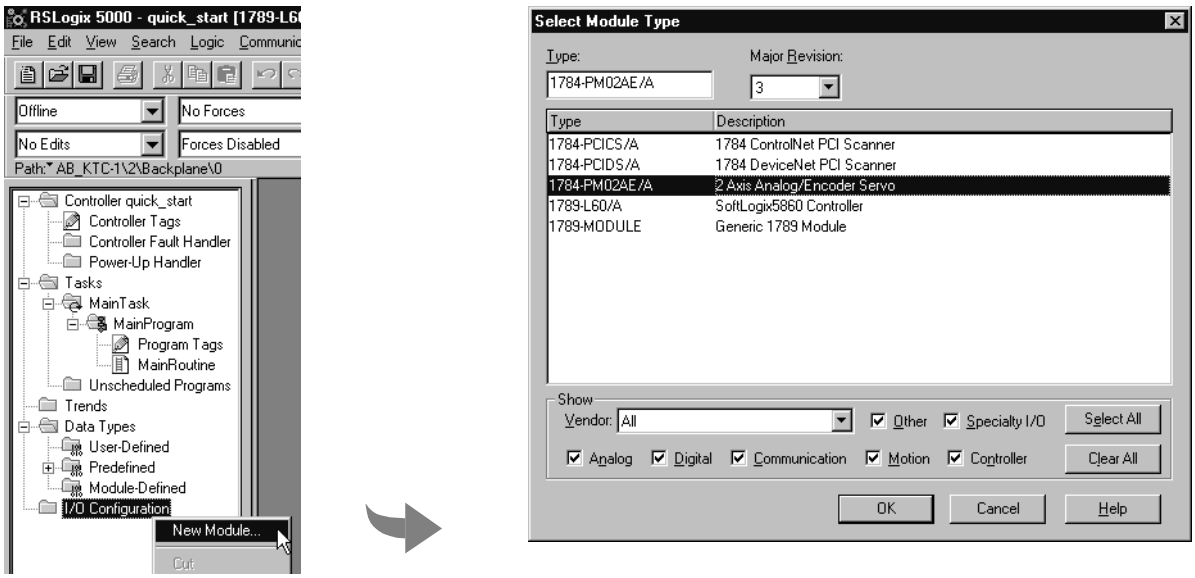
This chassis monitor has a 1784-PM02AE card installed in slot 3.

Step 3: Configure the motion card as part of the project

Use RSLogix 5000 programming software to map the 1784-PM02AE motion card as part of the SoftLogix project. In the Controller Organizer, add the card to the I/O Configuration folder.

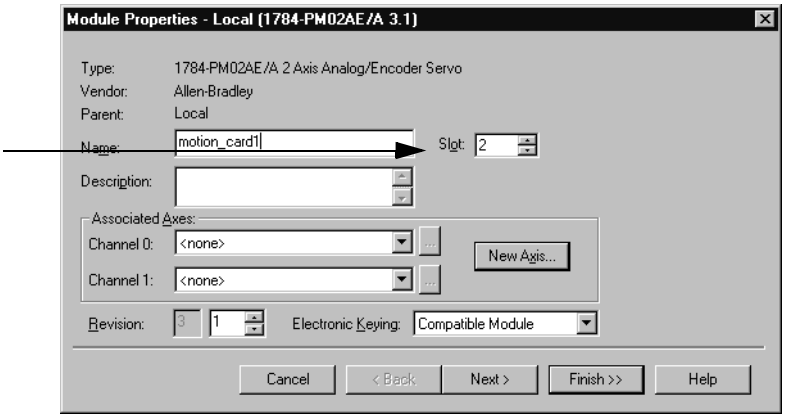
1. In RSLogix 5000 programming software, select the I/O Configuration folder.

2 Right-click to select New Module and add a 1784-PM02AE motion card.



3. Specify the appropriate motion card settings.

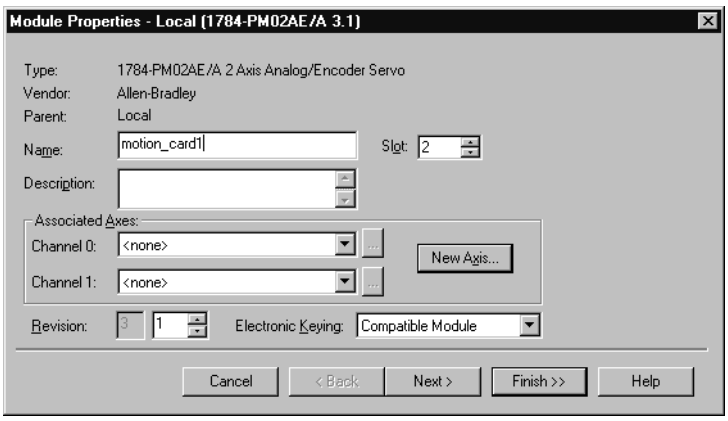
This must be the same slot number you specified on the SoftLogix chassis monitor.



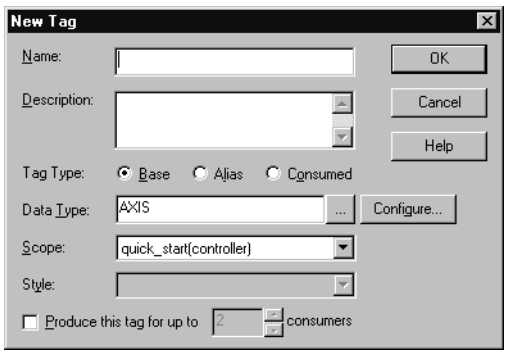
Creating an Axis

To create an axis, click **New Axis** in the module properties window.

Make sure you enter a name.



Specify this information:



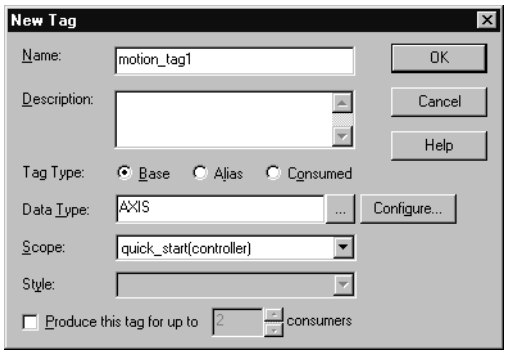
In this field:	Type:
Name	The name of the axis.
Description	A description of the axis (optional).

Configuring a servo axis

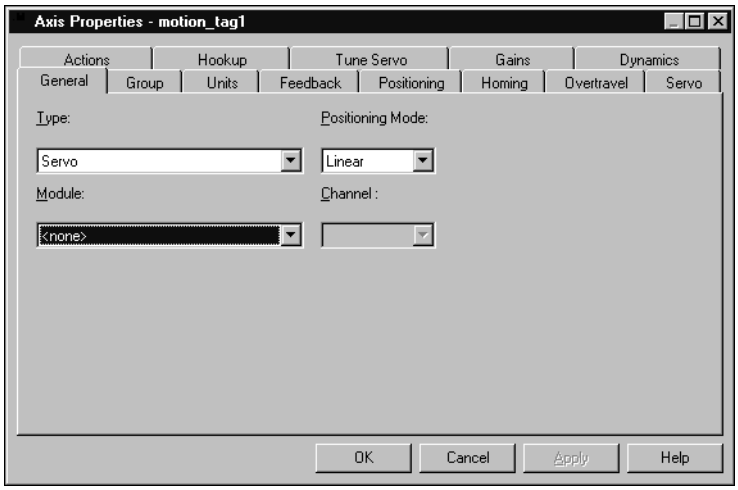
To configure the axis:

1. Click **Configure** in the new tag window.

Enter an axis name.



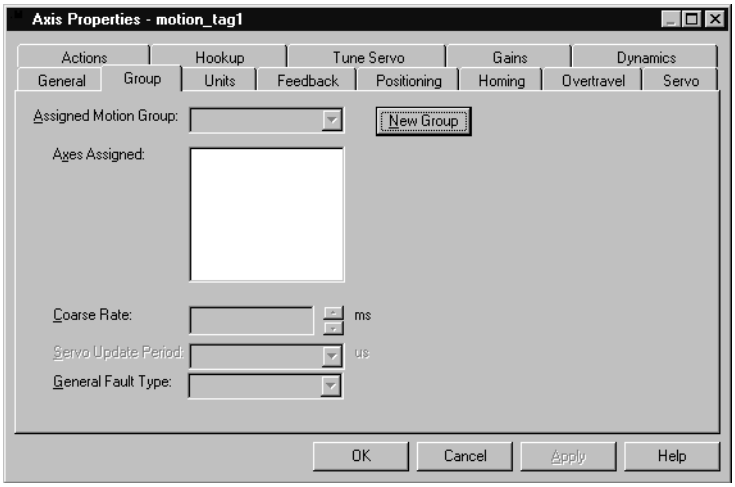
2. On the General tab, select the type of axis and positioning mode. (You assign a motion card and channel to the axis later.)



In this field:	Select the:
Type	Type of axis you want
Positioning Mode	Type of axis positioning you want to use

3. Click **OK**.

4. On the Group tab, assign a motion group.

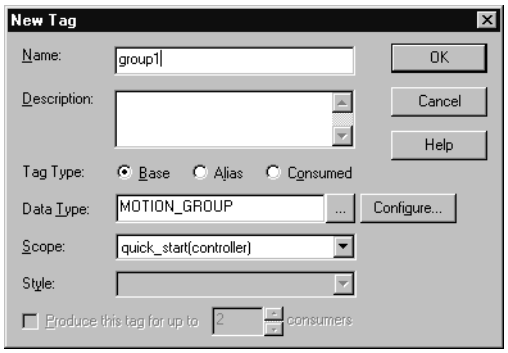


To:	Then:
create a new motion group	Click **New Group**.
use an existing motion group	Go to Step 7.

IMPORTANT During configuration, you must name and configure a motion group, which results in a MOTION_GROUP tag. After configuring the motion group, you can assign your axes to your motion group.

5. Specify this information:

Make sure you enter a group name. →

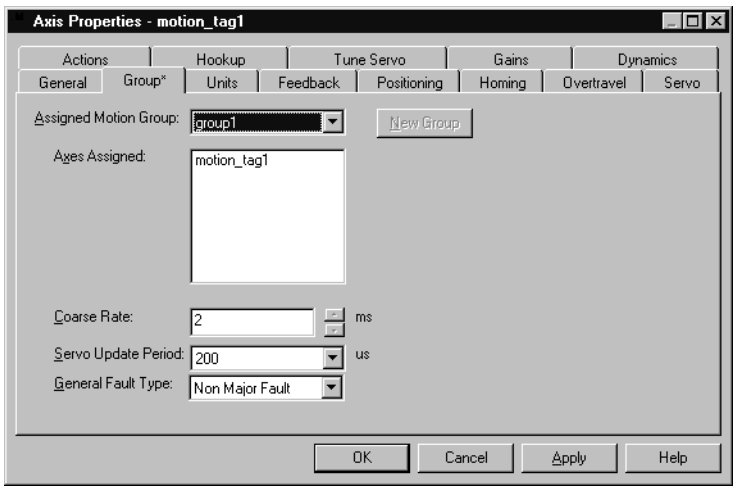


In this field:	Type:
Name	The name of the motion group.
Description	A description of the motion group (optional).

6. Click **OK**.

7. On the Group tab, assign the axis to a motion group and specify this information:

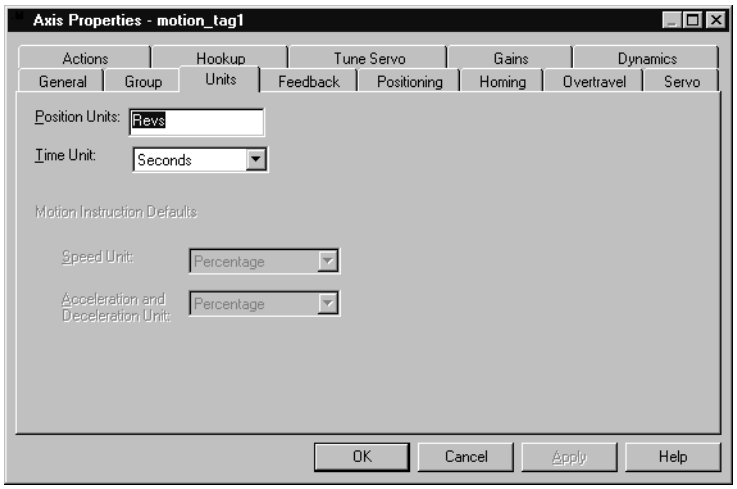
Select the motion group. →



In this field:	Select the:
Assigned Motion Group	Motion group.
Coarse Rate	Update rate for your axis
Servo Update Period	Closure time interval for your axis
General Fault Type	Fault type for your axis

8. Click **OK**.

9. On the Units tab, define the position units in which you want to program (e.g., meters, yards, feet, degrees).

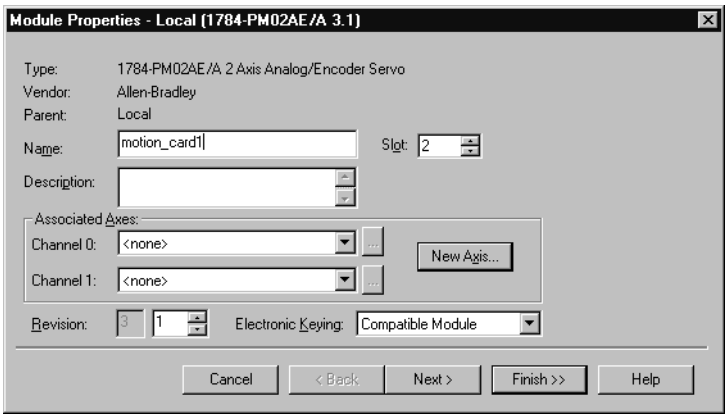


10. Click **OK**.

11. To continue configuring your axis, complete the entries on the other tabs. When finished with the entries on a tab, click **OK**.

IMPORTANT The diagnostic testing and auto tuning options are available only when the controller is online. Before going online, complete the configuration of all the motion cards and download your application program.

12. Assign the axis to a channel (the physical connection on the motion card to which the axis is wired).



To:	Then:
Assign your axis to channel 0	In the *Channel 0* field, select your axis from the drop-down menu
Assign your axis to channel 1	In the *Channel 1* field, select your axis from the drop-down menu
Add another axis	Click **New Axis**.
Complete your configuration	Select **Finish**.

IMPORTANT You can also name and configure axes and motion groups using the controller tag editor. The tag editor supports copy and paste operations, which can make axis naming and configuration easier and faster.

Running Hookup Diagnostics and Autotuning

Once you add and configure the motion cards and their axes, you can download your program. After going online, complete hookup diagnostics and auto tuning.

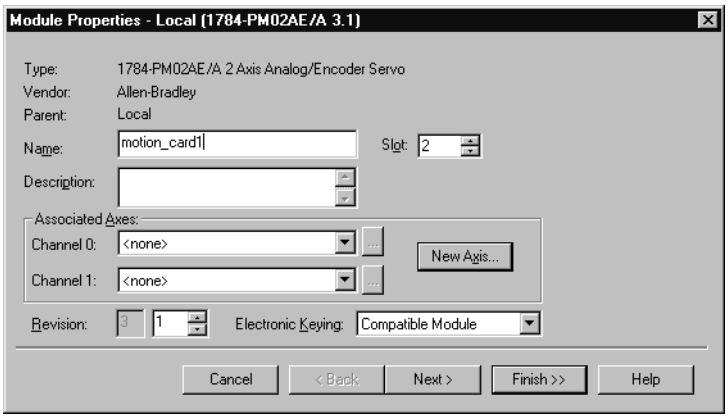
1. Download your project.

TIP

The project can be a blank program, but it must include complete configuration information for all your modules and axes.

2. Verify that a connection is established with each module in the I/O configuration of the controller.

3. In the module properties window for the motion card, select the channel that you assigned to the axis.



If you assign your axis to channel:	Then select the ... button next to:
0	Channel 0
1	Channel 1

4. Select the Hookup tab and run the hookup diagnostics.

   When the test is finished, the dialog box displays “Complete.”

5. Select the Tune Servo tab and run auto tuning.

6. When diagnostic testing and auto tuning are complete, click **OK**.

For more information about hookup diagnostics, see the *SoftLogix Servo Card Setup and Configuration User Manual*, publication 1784-UM003.

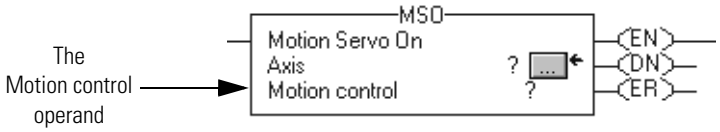
Developing Logic for Motion Control

To write a motion application program, you can insert motion instructions directly into your ladder logic program.

The motion instructions operate on one or more axes. You must identify and configure axes before you can use them.

For more information on individual motion instructions, see the *Logix5000 Controllers Motion Instruction Set Reference Manual*, publication 1756-RM007.

Each motion instruction has an operand named Motion control. This field uses a MOTION_INSTRUCTION tag to store status information during the execution of motion instructions. This status information can include instruction status, errors, etc.



ATTENTION

Tags used for the motion control operand of motion instruction should only be used once. Re-use of the same motion control operand in other instructions can cause unintended operation of the control variables.

For more information about the MOTION_INSTRUCTION tag, refer to the appropriate motion instruction in the *Logix5000 Controllers Motion Instruction Set Reference Manual*, publication 1756-RM007.

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

Method:	Example:
Directly accessing the MOTION_GROUP and AXIS structures	• Axis faults • Motion status • Servo status
Using the GSV instruction	• Actual position • Command position • Actual velocity

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

For more information on the SSV instruction, see the *Logix5000 Controllers General Instruction Set Reference Manual*, publication 1756-RM003.

Handling motion faults

Two types of motion faults exist.

Type	Description	Example
Errors	• Do not impact controller operation • Should be correct to optimize execution time and ensure program accuracy	A Motion Axis Move (MAM) instruction with a parameter out of range
Minor/Major	• Caused by a problem with the servo loop • Can shutdown the controller if you do not correct the fault condition	The application exceeded the PositionErrorTolerance value.

You can configure a fault as either minor or major by using the Axis Wizard-Group window.

Understanding errors

Executing a motion instruction within an application program can generate errors. The MOTION_INSTRUCTION tag has a field that contains the error code. For more information about error codes for individual instructions, see the *Logix5000 Controllers Motion Instruction Set Reference Manual*, publication 1756-RM007.

Understanding minor/major faults

Several faults can occur that are not caused by motion instructions. For example, a loss of encoder feedback or an actual position exceeding an overcareful limit will cause faults. The motion faults are considered type 2 faults with error codes from 1 to 32. See *Logix5000 Controllers Common Procedures Programming Manual*, publication 1756-PM001.

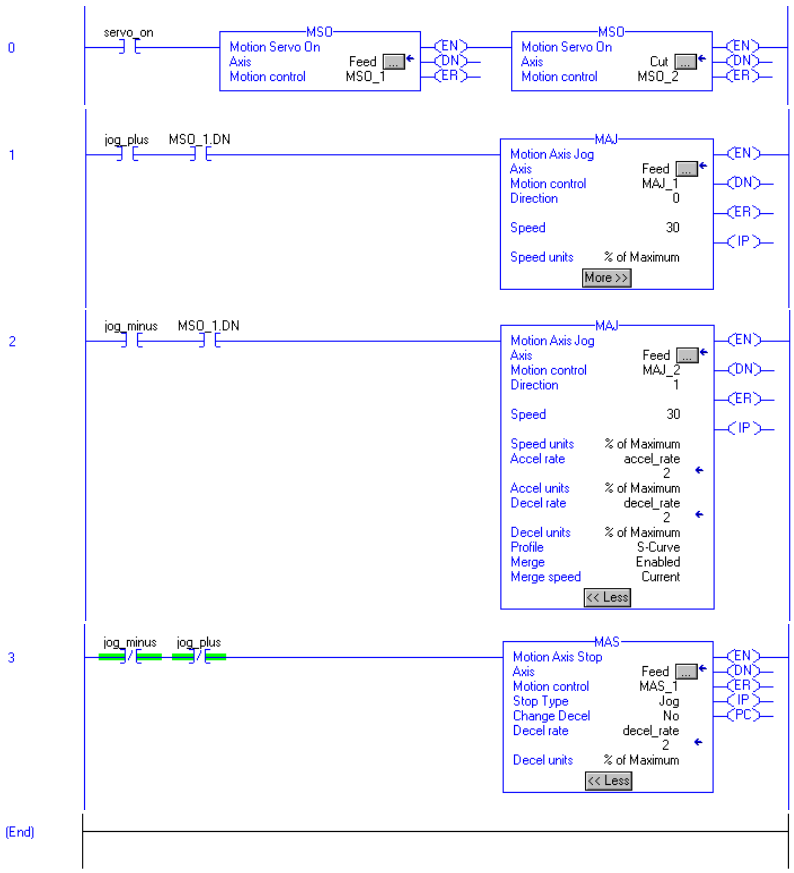
The following figure shows several rungs of a motion control application program.

Rung 0:
Enables the Feed and Cut axes when you press the servo_on button.

Rung 1:
Jogs the Feed axis in the positive direction when you press the jog_plus button.

Rung 2:
Jogs the Feed axis in the reverse direction when you press the jog_minus button.

Rung 3:
Stops the Feed axis when you release with the jog_plus button or the jog_minus button.



Communicating with Devices on a ControlNet Link

Using This Chapter

Configuring Your System for a ControlNet Link

For the SoftLogix controller to operate on a ControlNet network, you need:

- a ControlNet communication card:
  - if you want to send messages and control I/O, including produced/consumed tags, over ControlNet, use a 1784-PCICS card

    This chapter shows how to configure the 1784-PCICS communication card.
  - if you want to only send messages over ControlNet, use a 1784-PCIC card
- RSLinx software to install the virtual backplane driver

  You only install the virtual backplane driver once on the computer where you run the SoftLogix controller. This chapter assumes you have already installed the driver. For an example of installing the driver, see chapter 1, "Getting Started."
- RSLogix 5000 programming software to configure the communication card as part of the SoftLogix system
- RSNetWorx for ControlNet software to schedule the SoftLogix system on the network

Step 1: Install the hardware

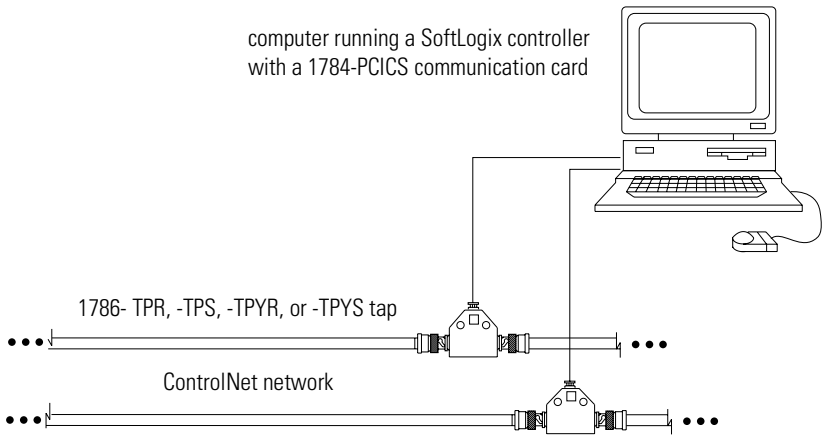
Make sure the 1784-PCICS communication card is properly installed in the computer. You need to:

- Install the card in any available PCI slot within the computer.

  It does not matter which PCI slot you use for the communication card. The PCI slot in the computer **does not** correspond to the backplane slot in the SoftLogix chassis. You use the SoftLogix chassis monitor to place the communication card in a specific backplane slot (see the next page).

- Make a label to place on the front of the card, or use a pen to write on the front of the card. The label should include the serial number of the card and a name you can use to identify the card from any others you might install in the computer.

  Remember the serial number and name of each communication card you install. You use this information to identify which card you want in which slot of the SoftLogix chassis.

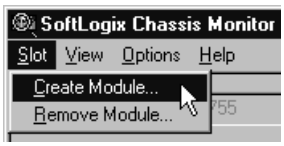


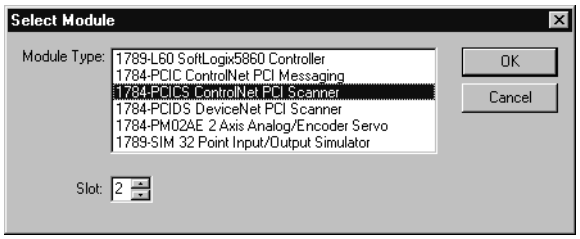
For more information about installing a 1784-PCICS communication card, see the *ControlNet PCI Interface Card Installation Instructions*, publication 1784-IN003.

Step 2: Create the communication card in the chassis

Before you can connect the SoftLogix system to the ControlNet network, you must create the 1784-PCICS card as part of the SoftLogix chassis.

1. From the SoftLogix chassis monitor, select Slot → Create Module or right click the appropriate slot and select Create. Select the 1784-PCICS card.

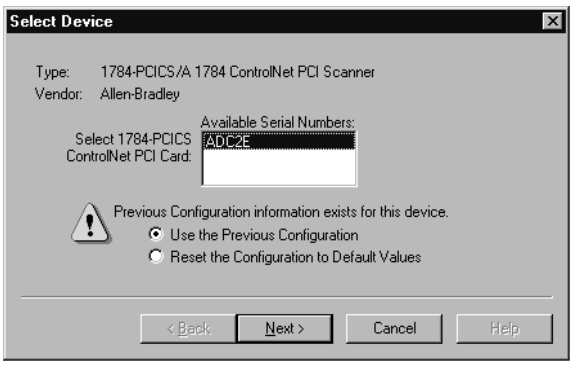




Specify the backplane slot number.

Click OK

2. Select the serial number of the 1784-PCICS card you want.



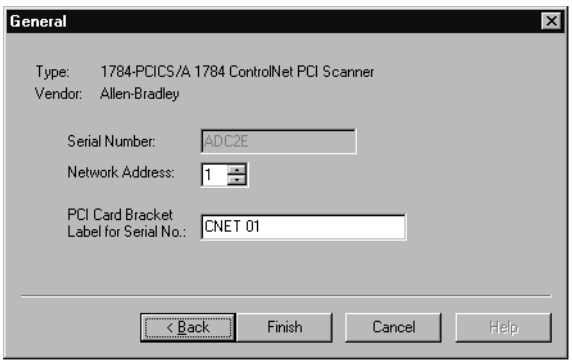
Select the serial number of the card.

If you previously configured the 1784-PCICS card that you selected by serial number, the chassis monitor remembers the configuration from the last time you used the card (whether in the same or different slot).

Click Next

3. Specify configuration settings for the 1784-PCICS card:

- specify the node address (MAC ID) on the ControlNet network
- enter the label name for the card (this is the name you wrote on the label of the card to help you identify the card from others in the same computer)



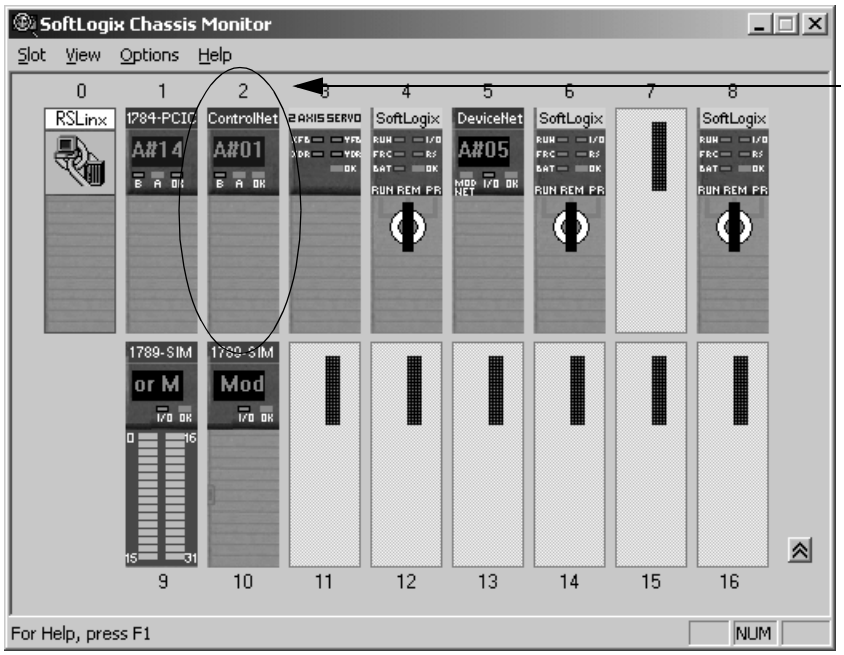
Click Finish

You can specify any slot number greater than 0 for the communication card. RSLinx software resides in slot 0.

By creating the card in the virtual chassis, you automatically install the communication driver information needed by the SoftLogix controller. Do not use RSLinx to install the communication driver for either the 1784-PCICS or 1784-PCIC communication card. Installing the communication driver through RSLinx adds the potential for conflicting configuration between RSLinx and the SoftLogix chassis monitor.

After you add the card to the chassis monitor, you can browse the network by expanding the Virtual Backplane driver and then expanding the port on the desired 1784-PCICS or 1784-PCIC communication card. Browsing ControlNet through the Virtual Backplane driver provides the same functionality as the RSLinx driver.

The chassis monitor shows the 1784-PCICS card as a virtual module in the SoftLogix chassis. The LEDs on the virtual monitor emulate a 1756-CNB communication module.



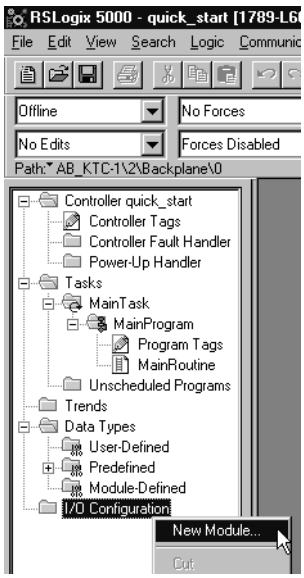
This chassis monitor has a 1784-PCICS card installed in slot 2.

Step 3: Configure the communication card as part of the project

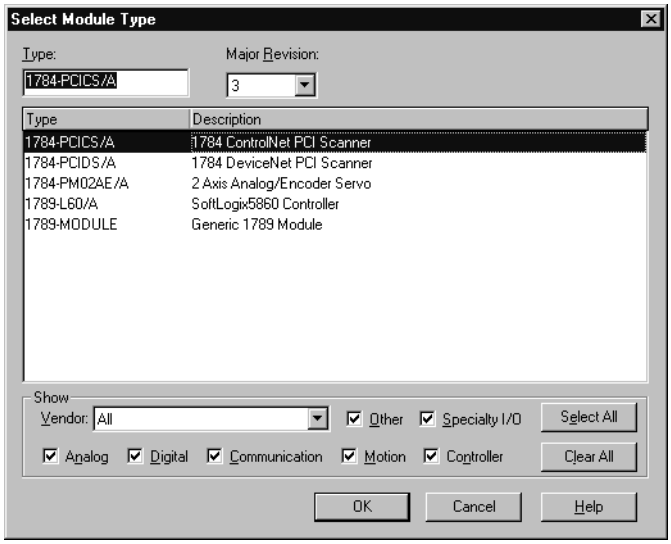
Use RSLogix 5000 programming software to map the 1784-PCICS communication card as part of the SoftLogix project. In the Controller Organizer, add the communication card to the I/O Configuration folder.

1. In RSLogix 5000 programming software, select the I/O Configuration folder.

2 Right-click to select New Module and add a 1784-PCICS communication card.

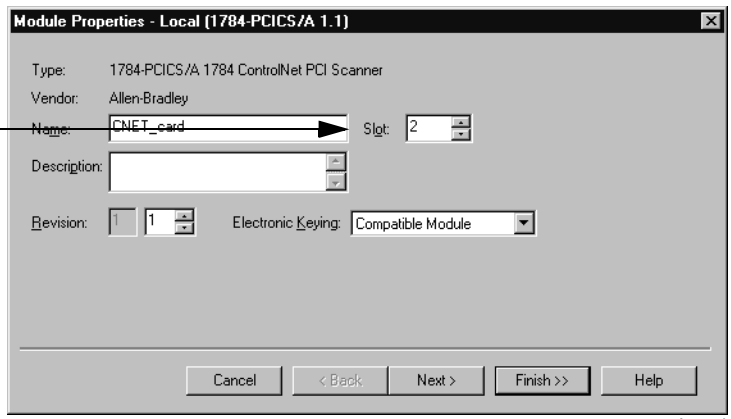


Click OK



3. Specify the appropriate communication card settings.

This must be the same slot number you specified on the SoftLogix chassis monitor.



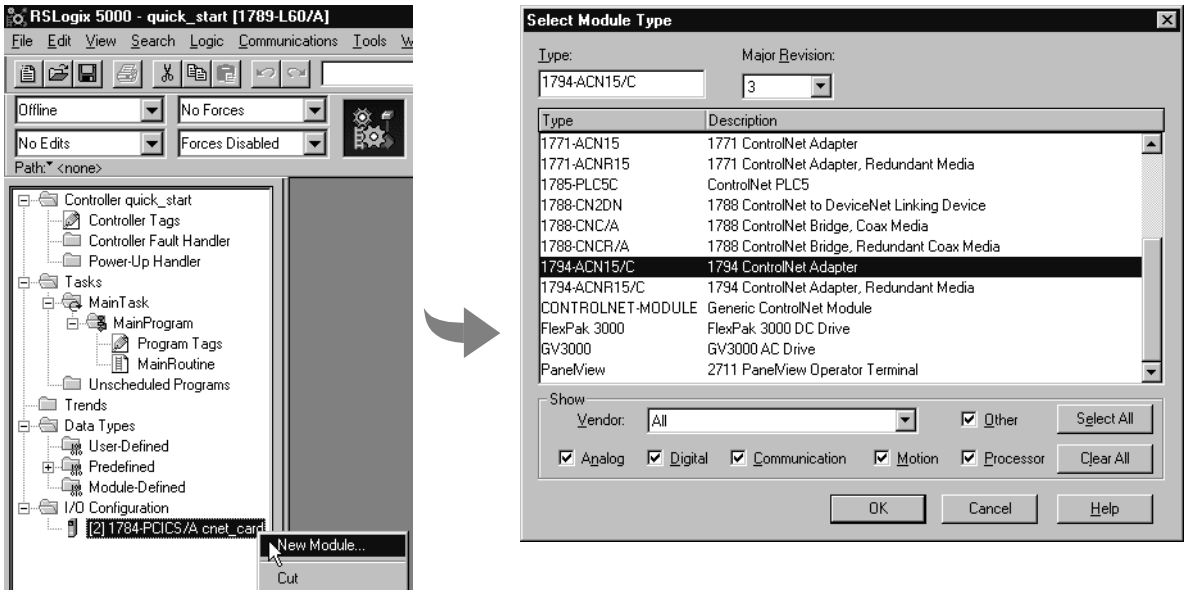
continued

The virtual backplane driver must be installed via RSLinx software before you can download a project to the SoftLogix controller.

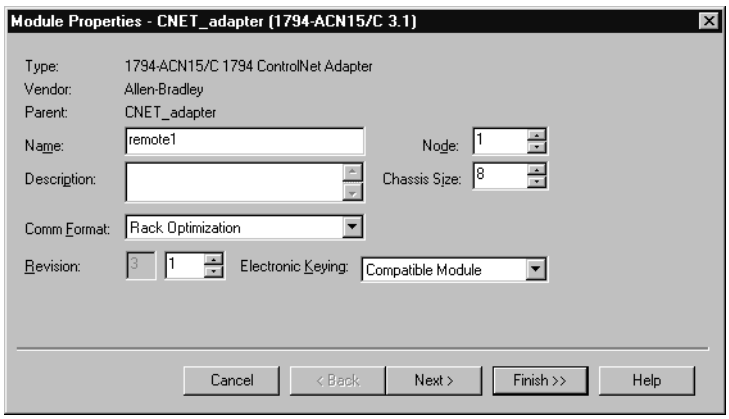
IMPORTANT Even if you plan to remotely program the controller over a ControlNet or Ethernet link, you must add the virtual backplane driver via RSLinx software. If you do not, persistent storage will not function and when you reboot the computer, the controller will come up with cleared memory (the program will not get re-loaded).

Complete your system configuration by adding the remote communication devices and appropriate I/O modules.

5. In the Controller Organizer, select the local 1784-PCICS communication you just added. Add and configure the remote communication device (1794-ACN15 in this example)



6. Specify the appropriate communication module settings.



7. Add and configure the I/O modules for the remote communication module you just added.

8. Save the current project and download it to the controller.

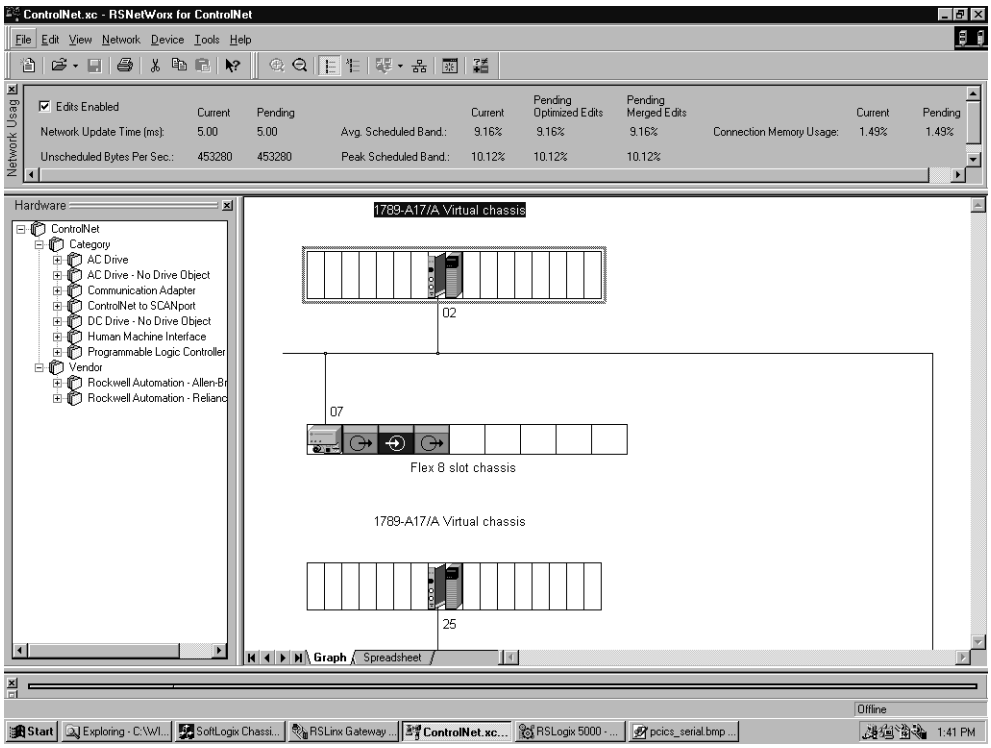
The local card becomes the “parent module” to the remote device. The controller organizer shows this parent/child relationship between local and remote communication devices.

Configure I/O modules for the remote communication module by adding them to the remote communication module (i.e., right-click the 1784-PCICS card and select New Module).

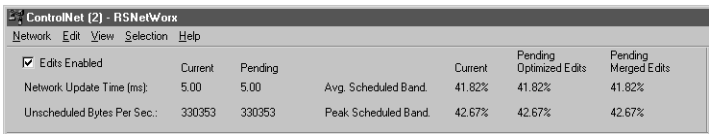
Step 4: Schedule the network

Use RSNetWorx software to schedule the ControlNet network. The controller project must already be downloaded from RSLogix 5000 programming software to the controller and the controller must be in Program or Remote Program mode.

1. In RSNetWorx software, go online, enable edits, and survey the network.



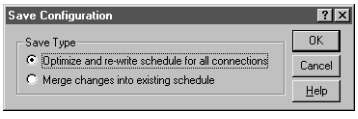
2. Specify the network update time (NUT).



The default NUT is 5ms.

The NUT you specify must be lower than or equal to the lowest RPI in your system.

3. After you specify the NUT, save and re-write the schedule for all connections.



Every device on the network must be in Program or Remote Program mode for the software to re-write all its connections. If a device is not in the correct mode, the software prompts you to let it change the device's mode.

Placing ControlNet I/O

The SoftLogix controller supports as many communication cards as you have PCI slots in the computer.

Each Logix-based communication module supports a limited number of scheduled and unscheduled connections. Take these limits into account when designing your system:

Device:	Description:	Maximum Scheduled Connections per Module:	Maximum Unscheduled Connections per Module:
1784-PCICS	SoftLogix ControlNet communication module	128 total connections, 127 of which can be scheduled connections	
1788-CNC 1788-CNCR	FlexLogix ControlNet communication card	10 scheduled connections	32 unscheduled connections (each scheduled connection reduces the number of unscheduled connections by 1)
1756-CNB 1756-CNBR	ControlLogix ControlNet communication module	64 total connections, any combination of scheduled and unscheduled	
1794-ACN15 1794-ACNR15	FLEX ControlNet adapter module	9 total connections, any combination of scheduled and unscheduled	

Accessing I/O

I/O information is presented as a structure of multiple fields, which depend on the specific features of the I/O module. The name of the structure is based on the location of the I/O module in the system. Each I/O tag is automatically created when you configure the I/O module through the programming software. Each tag name follows this format:

Location:SlotNumber:Type.MemberName.SubMemberName.Bit

where:

This address variable:	Is:
Location	Identifies network location ADAPTER_NAME = identifies remote adapter or bridge device
SlotNumber	Slot number of I/O module in its chassis
Type	Type of data I = input O = output C = configuration S = status
MemberName	Specific data from the I/O module; depends on the type of data the module can store For example, Data and Fault are possible fields of data for an I/O module. Data is the common name for values the are sent to or received from I/O points.
SubMemberName	Specific data related to a MemberName.
Bit (optional)	Specific point on the I/O module; depends on the size of the I/O module (0-31 for a 32-point module)

EXAMPLE

The tags created for the remote device (1794-ACN15 in this example) depend on the communication format you select for that device when you add the device to the I/O Configuration folder.

If you select:	The automatically-created tags are for a:
Rack Optimization	rack-optimized connection to the remote communication device
Listen Only - Rack Optimization	rack-optimized connection to the remote communication device (not available on all communication devices)
None	direct connection to the individual I/O modules with no connection to the remote communication device

Working with a rack-optimized connection

The rack-optimized connection creates a DINT element for each possible I/O module connected to the device “remote_flex.” The array remote_flex:I.Data contains the possible input elements; the remote_flex:O.Data contains the possible output elements.

remote_flex:I			AB:1794_ACN15_8SLOT:I:0	
remote_flex:I.SlotStatusBits			DINT	Binary
remote_flex:I.Data			INT[8]	Binary
remote_flex:I.Data[0]			INT	Binary
remote_flex:I.Data[1]			INT	Binary
remote_flex:I.Data[2]			INT	Binary
remote_flex:I.Data[3]			INT	Binary
remote_flex:I.Data[4]			INT	Binary
remote_flex:I.Data[5]			INT	Binary
remote_flex:I.Data[6]			INT	Binary
remote_flex:I.Data[7]			INT	Binary
remote_flex:O			AB:1794_ACN15_8SLOT:O:0	
remote_flex:O.Data			INT[8]	Binary
remote_flex:0:I	remote_flex:I.Data[0]	remote_flex:I.Data[0]	INT	Binary
remote_flex:0:C			AB:1794_DI_Delay8:C:0	
remote_flex:1:O	remote_flex:O.Data[1]	remote_flex:O.Data[1]	INT	Binary
remote_flex:1:C			AB:1794_DO8:C:0	
remote_flex:2:I			AB:1794_IF2XOF2I:I:0	
remote_flex:2:O			AB:1794_IF2XOF2I:O:0	
remote_flex:2:C			AB:1794_IF2XOF2I:C:0	

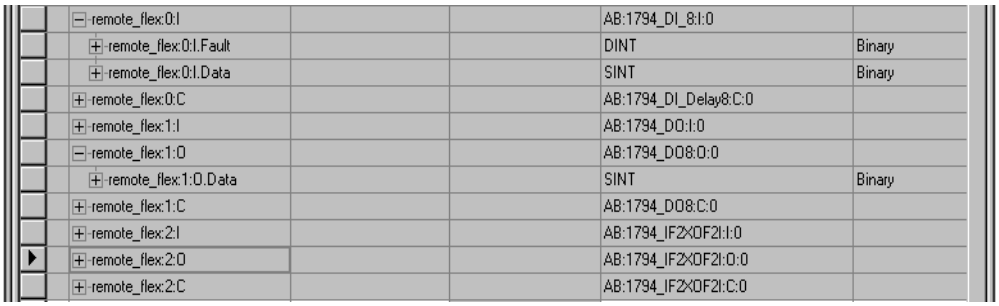
The tags for the individual, digital I/O modules are actually aliases back into the rack-optimized array tag. For example “remote_flex:0:I” is an alias to “remote_flex:I.Data[0]. These digital I/O modules were configured with a rack-optimized communication format to take advantage of the rack-optimized array tag created for the communication device.

The index number on the array element refers to the slot number on “remote_flex.” For example, Data[2] refers to the module in slot 2. You can have only one I/O module in a given slot, so Data[2] is only used in either the input or output array. That same element in the other array still exists even though it does not contain actual data. You can create aliases to the elements you actually use to more readily identify the data you need.

Note that the tags for the analog module (“remote_flex:2:I,” “remote_flex:2:O,” and “remote_flex:2:C”) are not aliases. Analog modules require direct connections to operate. Do not use the element of the rack-optimized array tag to control the analog module. Use the individual, slot-referenced tag.

Working with direct connections

If you select None for the communication format to the communication device, the software assumes that you want a direct connection for each I/O module connected to that device. The software creates slot-referenced tags for each I/O module, but not for the communication device.



Sending Messages

The SoftLogix controller can send MSG instructions to other controllers over a ControlNet link. Each MSG instruction requires you to specify a target and an address within the target. The number of messages that a device can support depends on the type of message and the type of device:

This device:	Support this many unconnected messages:	Support this many connected messages:
1756-CNB module (for a Logix5550 controller)	20	64
1784-PCICS card (for a SoftLogix controller)	50	128
1788-CNx daughtercard (for a FlexLogix controller)	depends on amount of memory used for each MSG	32
ControlNet PLC-5 controller	32	32

MSG instructions are unscheduled connections. The type of MSG determines whether or not it requires a connection. If the MSG instruction requires a connection, it opens the needed connection when it is executed. You can configure the MSG instruction to keep the connection open (cache) or to close it after sending the message.

This type of MSG:	Using this communication method:	Uses a connection:	Which you can cache:
CIP data table read or write	CIP	✓	✓
PLC-2, PLC-3, PLC-5, or SLC (all types)	CIP		
	CIP with Source ID		
	DH+	✓	
CIP generic	na		
block-transfer read or write	na	✓	✓

Communicating with another Logix-based controller

All Logix-based controllers can use MSG instructions to communicate with each other. The following examples show how to use tags in MSG instructions between Logix-based controllers.

Type of MSG Instruction:	Example Source and Destination:	
Logix-based controller writes to Logix-based controller (CIP Data Table Write)	source tag	*array_1*
	destination tag	*array_2*
Logix-based controller reads from Logix-based controller (CIP Data Table Read)	source tag	*array_1*
	destination tag	*array_2*

The source and destination tags:

- must be controller-scoped tags.
- can be of any data type, except for AXIS, MESSAGE, or MOTION_GROUP.

Communicating with other controllers over ControlNet

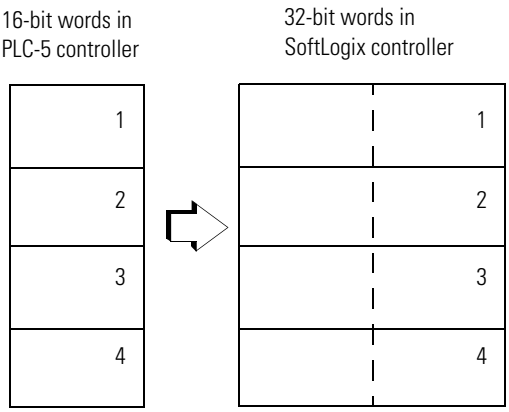
The SoftLogix controller also uses MSG instructions to communicate with PLC and SLC controllers. The MSG instructions differ depending on which controller initiates the instruction.

For MSG instructions originating from a SoftLogix controller to a PLC or SLC controller:

Type of MSG Instruction:	Supported Source File Types:	Supported Destination File Types:
SoftLogix writes to PLC-5 or SLC	In the SoftLogix controller, specify the source data type based on the destination device: PLC-5: SINT, INT, DINT, or REAL SLC: INT Example source element: *array_1*	Specify the destination file type based on the destination device: PLC-5 typed write: S, B, N, or F PLC-5 word-range write: S, B, N, F, I, O, A, or D SLC: B or N Example destination tag: *N7:10*
SoftLogix writes to PLC-2	In the SoftLogix controller, select one of these data types: SINT, INT, DINT, or REAL Example source element: *array_1*	Use the PLC-2 compatibility file. Example destination tag: *010*
SoftLogix reads from PLC-5 or SLC	Specify the destination file type based on the destination device: PLC-5 typed read: S, B, N, or F PLC-5 word-range read: S, B, N, F, I, O, A, or D SLC: B or N Example source element: *N7:10*	In the SoftLogix controller, specify the destination data type based on the destination device: PLC-5: SINT, INT, DINT, or REAL SLC: INT Example destination tag: *array_1*
SoftLogix reads from PLC-2	Use the PLC-2 compatibility file. Example source element: *010*	In the SoftLogix controller, select one of these data types: SINT, INT, DINT, or REAL Example destination tag: *array_1*

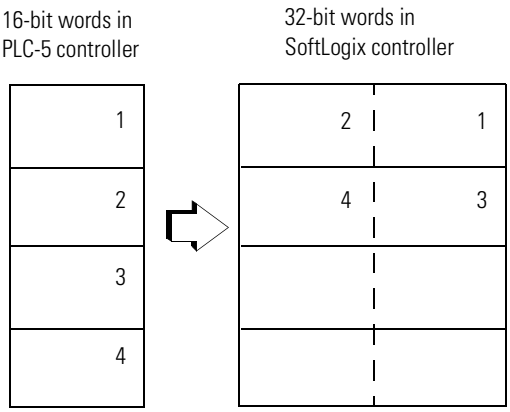
The SoftLogix controller can send typed or word-range commands to PLC-5 controllers. These commands read and write data differently. The following diagrams show how the typed and word-range commands differ.

Typed read command



The typed commands maintain data structure and value.

Word-range read command



The word-range commands fill the destination tag contiguously. Data structure and value change depending on the destination data type.

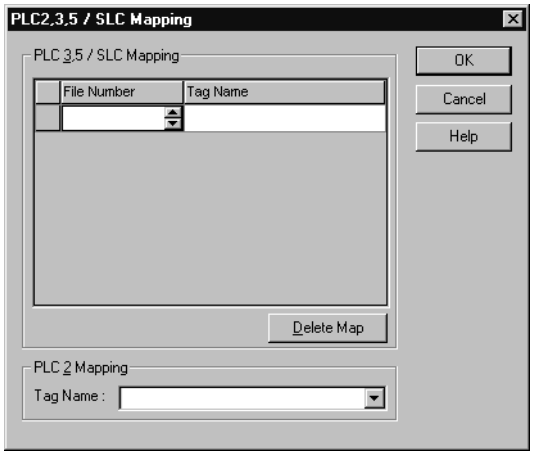
The SoftLogix controller can process messages initiated from PLC or SLC controllers. These messages use data table addresses. In order for these controllers to access tags within the SoftLogix controller, you map tags to data table addresses.

Mapping addresses

The programming software includes a PLC/SLC mapping tool which allows you to make an existing controller array tag in the local controller available to PLC-2, PLC-3, PLC-5, or SLC controllers.

To map addresses:

1. From the Logic menu, select Map PLC/SLC Messages.



2. Specify this information:

For:	In this field:	Specify:	For example:
PLC-3, PLC-5, and SLC controllers	File Number	Type the file number of the data table in the PLC/SLC controller.	*10*
	Tag Name	Type the array tag name the local controller uses to refer to the PLC/SLC data table address. The tag must be an integer array (SINT, INT, or DINT) that is large enough for the message data.	*array_1*
PLC-2 controllers	Tag Name	Type the tag name to be the PLC-2 compatibility file.	*200*

TIP

You can map as many tags as you want to a PLC-3, PLC-5, or SLC controller. You can map only one tag to a PLC-2 controller.

The following table shows example source and destination tags and elements for different controller combinations.

Type of MSG Instruction:	Example Source and Destination:	
PLC-5 writes to SoftLogix SLC writes to SoftLogix SLC 5/05 SLC 5/04 OS402 and above SLC 5/03 OS303 and above	source element	*N7:10*
	destination tag	*"array_1"*
	The PLC-5, PLC-3, and SLC controllers support logical ASCII addressing so you do not have to map a compatibility file for MSG instructions initiated by a PLC-5, PLC-3, or SLC controller. Place the SoftLogix tag name in double quotes (").	
	You could optionally map a compatibility file. For example, if you enter *10* for the compatibility file, you enter *N10:0* for the destination tag.	
PLC-2 writes to SoftLogix	source element	*010*
	destination tag	*200*
	The destination tag is the three-digit PLC-2 address you specified for PLC-2 mapping.	
PLC-5 reads from SoftLogix SLC reads from SoftLogix SLC 5/05 SLC 5/04 OS402 and above SLC 5/03 OS303 and above	source tag	*"array_1"*
	destination element	*N7:10*
	The PLC-5, PLC-3, and SLC controllers support logical ASCII addressing so you do not have to map a compatibility file for MSG instructions initiated by a PLC-5, PLC-3, or SLC controller. Place the SoftLogix tag name in double quotes (").	
	You could optionally map a compatibility file. For example, if you enter *10* for the compatibility file, you enter *N10:0* for the source tag.	
PLC-2 reads from SoftLogix	source tag	*200*
	destination element	*010*
	The source tag is the three-digit PLC-2 address you specified for PLC-2 mapping.	

When the SoftLogix controller initiates messages to PLC or SLC controllers, you do not have to map compatibility files. You enter the data table address of the target device just as you would a tag name.

SLC 5/05 controllers, SLC 5/04 controllers (OS402 and above), and SLC 5/03 controllers (OS303 and above) support logical ASCII addressing and support PLC/SLC mapping (see the examples above). For all other SLC or MicroLogix1000 controllers, you must map a PLC-2 compatibility file (see the PLC-2 examples above).

Producing and Consuming Data

The SoftLogix controller supports the ability to produce (broadcast) and consume (receive) system-shared tags over a ControlNet link. Produced data is accessible by multiple controllers over a ControlNet network. Produced and consumed data are scheduled connections because the controller sends or receives data at a predetermined rate.

Produced and consumed tags must be controller-scoped tags of DINT or REAL data type, or in an array or structure.

Tag type:	Description:	Specify:
produced	These are tags that the controller produced for other controllers to consume.	• Enabled for producing • How many consumers allowed
consumed	These are tags whose values are produced by another controller.	• Controller name that owns the tag that the local controller wants to consume • Tag name or instance that the controller wants to consume • Data type of the tag to consume • Update interval of how often the local controller consumes the tag

The producer and consumer must be configured correctly for the specified data to be shared. A produced tag in the producer must be specified exactly the same as a consumed tag in the consumer. A produced tag must also be controller-scoped.

If any produced/consumed tag between a producer and consumer is not specified correctly, none of the produced/consumed tags for that producer and consumer will be transferred. However, other consumers can still access their shared tags, as long as their tags are specified correctly. One consumer failing to access shared data does not affect other consumers accessing the same data.

Maximum number of produced and consumed tags

The maximum number of produced/consumed tags that you can configure depends on the connection limits of the communication device that transfers the produced/consumed data.

Each produced tag uses one connection for the tag and the first configured consumer of the tag. Each consumer thereafter uses an additional connection.

Size limit of a produced or consumed tag

A produced or consumed tag can be as large as 488 bytes, but it must also fit within the bandwidth of the ControlNet network:

- As the number of connections over a ControlNet network increases, several connections, including produced or consumed tags, may need to share a network update.
- Since a ControlNet network can only pass 500 bytes in one update, the data of each connection must be less than 488 bytes to fit into the update.

If a produced or consumed tag is too large for your ControlNet network, make one or more of the following adjustments:

- Reduce the Network Update Time (NUT). At a faster NUT, less connections are able to share an update slot.
- Increase the Requested Packet Interval (RPI) of all connections. At a higher RPI, connections can take turns sending data during an update slot.
- For a ControlNet bridge module in a remote chassis, select the most efficient communication format for that chassis:.

Are most of the modules in the chassis non-diagnostic, digital I/O modules?	Then select this communication format for the remote communication module:
yes	rack optimization
no	none

  The Rack Optimization format uses an additional 8 bytes for each slot in its chassis. Analog modules or modules that are sending or getting diagnostic, fuse, or timestamp data require direct connections and cannot take advantage of the rack optimized form. Selecting “None” frees up the 8 bytes per slot for other uses, such as produced or consumed tags.

- Separate the tag into two or more smaller tags:
  - Group the data according to similar update rates. For example, you could create one tag for data that is critical and another tag for data that is not as critical.
  - Assign a different RPI to each tag.
- Create logic to transfer the data in smaller sections (packets).

Producing a tag

Produced data can be:

- tag of DINT or REAL data type.
- array of DINT or REAL elements.
- user-defined structure with any type elements. Use a user-defined structure to group BOOL, SINT, and INT data.

To create a produced tag:

1. You must be programming offline.

2. In the controller organizer, double-click the Controller Tags folder and then click the Edit Tags tab (at the bottom of the window).

3. Select the tag that you want to produce, or right-click to enter a new tag, and display the Tag Properties dialog box.

4. Select the “Produce this tag” check box. Specify how many controllers can consume the tag.

You can produce a base, alias, or consumed tag.

The consumed tag in a receiving controller must have the same data type as the produced tag in the originating controller. The controller performs type checking to ensure proper data is being received.

Produced tags require connections. The number of connections depend on the amount of data and how many controllers are producing and consuming tags.

Consuming a tag

A consumed tag represents data that is produced (broadcast) by one controller and received and stored by the consuming controller. To create a consumed tag:

1. You must be programming offline.
2. In the controller organizer, double-click the Controller Tags folder and then click the Edit Tags tab.
3. Select the tag that you want to consume, or enter a new tag, and display the Tag Properties dialog box.
4. Specify:

In this field:	Type or select:
Tag Type	Select Consumed.
Controller	Select the name of the other controller. You must have already created the controller in the controller organizer for the controller name to be available.
Remote Tag Name Remote Instance	Type a name for the tag in the other controller you want to consume. **Important:** The name must match the name in the remote controller exactly, or the connection faults. If the remote controller is a ControlNet PLC-5, this field is Remote Instance. Select the instance number (1-128) of the data on the remote controller.
RPI (requested packet interval)	Type the amount of time in msec between updates of the data from the remote controller. The local controller will receive data at least this fast.
Display Style	If you are creating a consumed tag that refers to a tag whose data type is BOOL, SINT, INT, DINT, or REAL, you can select a display style. This display style defines how the tag value will be displayed in the data monitor and ladder editor. The display style does not have to match the display style of the tag in the remote controller.

All consumed tags are automatically controller-scope.

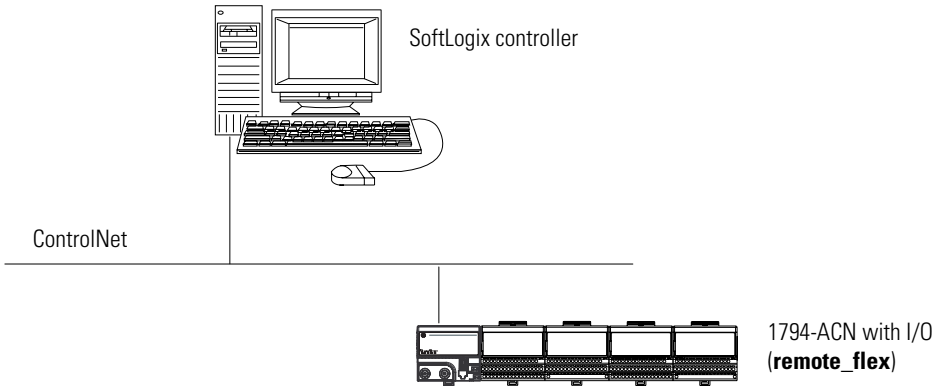
To consume data from a remote controller, use RSNetWorx software to schedule the connection over the ControlNet network.

The produced tag in the originating SoftLogix controller must have the same data type as the consumed tag in the other SoftLogix controller. The SoftLogix controller performs type checking to ensure proper data is being received.

IMPORTANT If a consumed-tag connection fails, all of the other tags being consumed from that remote controller stop receiving data.

Example 1: SoftLogix Controller and I/O

In the following example, one SoftLogix controller controls I/O through a 1794-ACN15 module.



Example 1: Controlling I/O

This example has the SoftLogix controller controlling the I/O connected to the remote 1794-ACN15 module. The data the SoftLogix controller receives from the I/O modules depends on how you configure the I/O modules. You can configure each module as a direct connection or as a rack-optimized connection. One location (chassis or DIN rail) can have a combination of some modules configured as a direct connection and others as rack optimized.

Example 1: Total connections required by the SoftLogix controller

The following table calculates the connections used in this example.

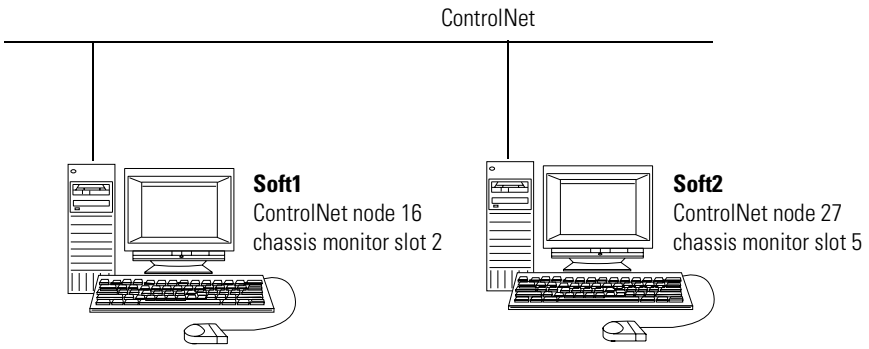
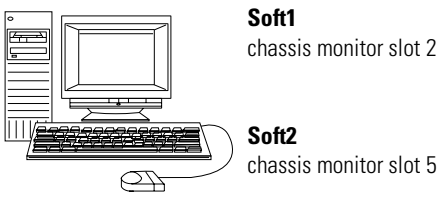
Connection:	Amount:
SoftLogix controller to 1784-PCICS card	0
SoftLogix controller to remote 1794-ACNR15 (communication format is “none”)	0
SoftLogix controller to 4 I/O modules (through 1794-ACNR15) all modules configured as direct connection	4
total connections used:	4

If you configure the 1794-ACNR15 as rack-optimized and the I/O modules as rack-optimized, you only use one connection to the 1794-ACN15 module, reducing the above example by 3 connections. The following table calculates the connections for this rack-optimized configuration.

Connection:	Amount:
SoftLogix controller to 1784-PCICS card	0
SoftLogix controller to remote 1794-ACNR15 (communication format is "rack optimization")	1
SoftLogix controller to 4 I/O modules (through 1794-ACNR15) all modules configured as rack optimized connections	0
total connections used:	1

Example 2: SoftLogix Controller to SoftLogix Controller

In the following example, one SoftLogix controller communicates with another SoftLogix controller over ControlNet. The two controllers can be in separate computers or in the same computer.

Example:	Illustration:
Each SoftLogix controller resides in its own computer	
Each SoftLogix controller resides in the same computer	

Example 2: Sending a MSG instruction

To send a MSG from Soft1 to Soft2:

1. For Soft1, create a controller-scoped tag and select the MESSAGE data type.

2. Enter a MSG instruction.

   In this example logic, a message is sent when a specific condition is met. When count_send is set, send count_msg.

3. Configure the MSG instruction. On the Configuration tab:

For this item:	Specify:
Message Type	CIP Data Table Read or CIP Data Table Write
Source Tag	Tag containing the data to be transferred
Number of Elements	Number of array elements to transfer
Destination Tag	Tag to which the data will be transferred

4. On the Communication tab, specify the communication path.

   A communication path requires pairs of numbers. The first number in the pair identifies the port from which the message exits. The second number in the pair designates the node address of the next device.

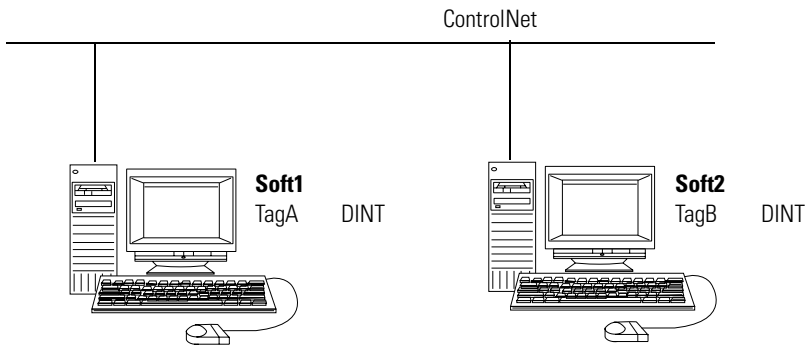
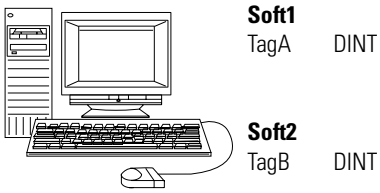
For this item:	Specify:
Communication Path (each SoftLogix controller resides in its own computer)	1,2,2,27,1,5 where: 1 is the SoftLogix backplane of Soft1 2 is 1784-PCICS card in slot 2 2 is the ControlNet port 27 is the ControlNet node of Soft2 1 is the SoftLogix backplane of Soft2 5 is the controller slot of Soft2
Communication Path (each SoftLogix controller resides in the same computer)	1,5 where: 1 is the SoftLogix backplane of Soft1 5 is the controller slot of Soft2

Example 2: Producing and consuming tags

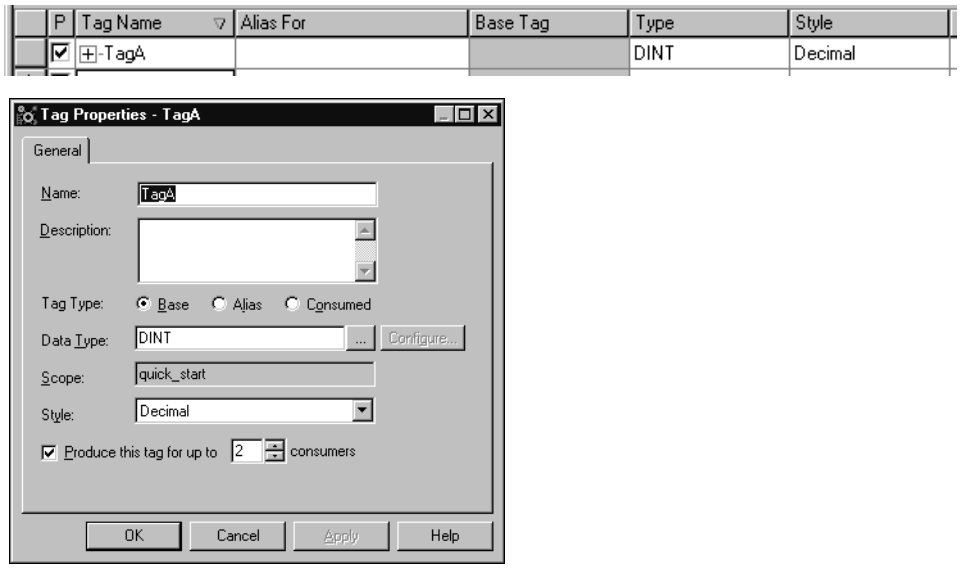
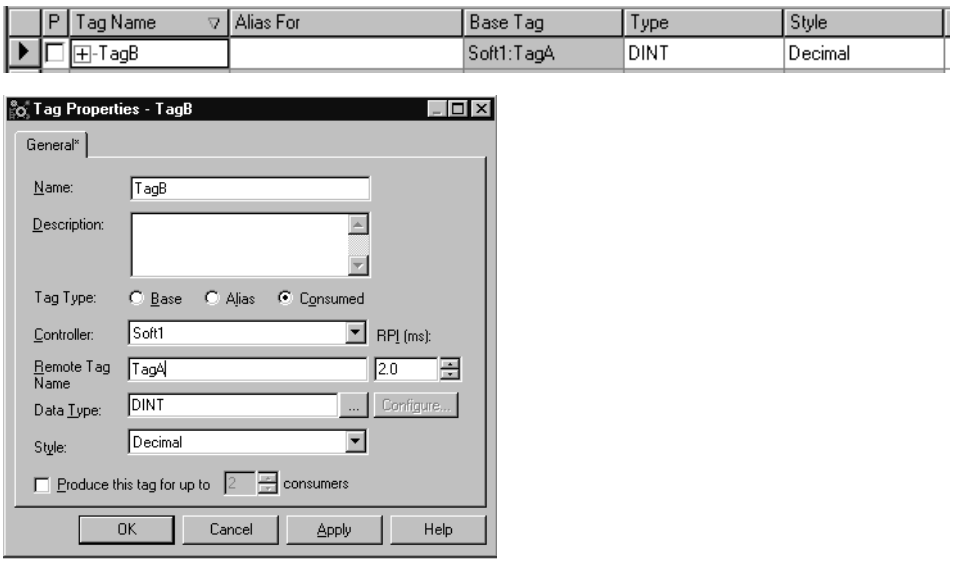
You can produce a base, alias, or consumed tag. Produced data can be:

- tag of DINT or REAL data type.
- array of DINT or REAL elements.
- user-defined structure with any type elements. Use a user-defined structure to group BOOL, SINT, and INT data.

The consumed tag must have the same data type as the produced tag in the originating controller. The controller performs type checking to ensure proper data is being received.

Example:	Illustration:
Each SoftLogix controller resides in its own computer	
Each SoftLogix controller resides in the same computer, using different CPUs	

This example shows Soft1 as producing TagA. Soft2 consumes TagA and stores is as TagB:

In this controller:	The tags look like:
Soft1	
Soft2	

Each produced tags requires one connection for the producing controller and an additional connection for each consuming controller. Each consumed tag requires one connection.

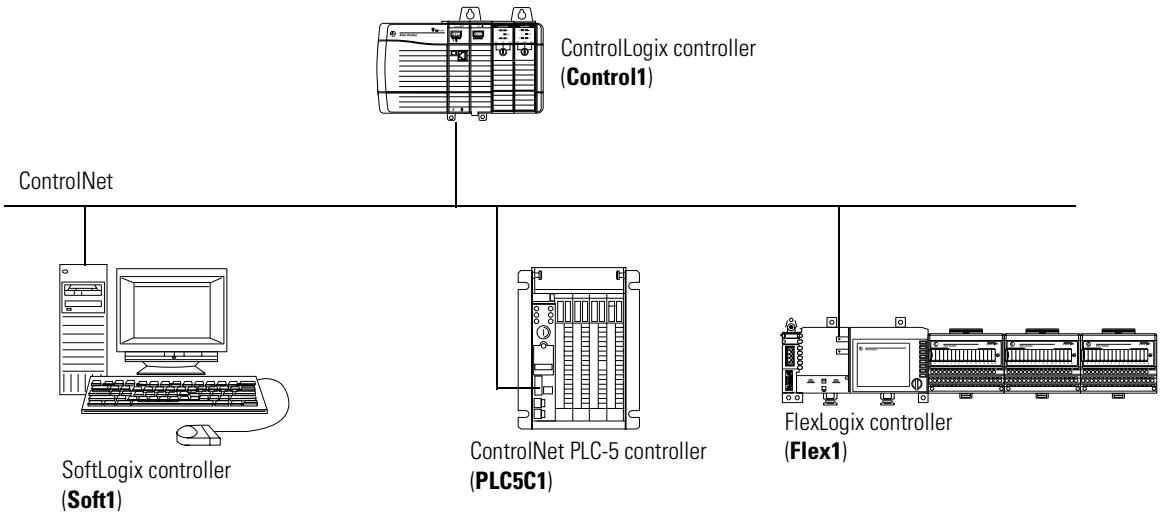
Example 2: Total connections required by Soft1 controller

The following table calculates the connections used in this example.

Connection:	Amount:
Soft1 controller to 1784-PCICS card	0
Soft1 controller to remote 1784-PCICS card	0
connected, cached MSG from Soft1 to Soft2	1
produced TagA produced from Soft1 to Soft2 other consumer (2 are configured)	 1 1
consumed TagB	1
total connections used:	4

Example 3: SoftLogix Controller to Other Devices

In the following example, one SoftLogix controller communicates with other controllers over ControlNet.



Example 3: Sending MSG instructions

You configure a MSG instruction to a ControlLogix and FlexLogix controller the same as you do for a SoftLogix controller. All Logix-based controllers follow the same MSG configuration requirements. See Example 2 above.

Configuring a MSG instruction for a PLC-5 controller depends on the originating controller.

For MSG instructions originating from the SoftLogix controller to the ControlNet PLC-5 controller:

Type of Logix MSG instruction:	Source:	Destination:
Typed Read	any integer element (such as B3:0, T4:0.ACC, C5:0.ACC, N7:0, etc.)	SINT, INT, or DINT tag
	any floating point element (such as F8:0, PD10:0.SP, etc.)	REAL tag
Typed Write	SINT or INT tag	any integer element (such as B3:0, T4:0.ACC, C5:0.ACC, N7:0, etc.)
	REAL tag	any floating point element (such as F8:0, PD10:0.SP, etc.)
Word Range Read	any data type (such as B3:0, T4:0, C5:0, R6:0, N7:0, F8:0, etc.)	SINT, INT, DINT, or REAL
Word Range Write	SINT, INT, DINT, or REAL	any data type (such as B3:0, T4:0, C5:0, R6:0, N7:0, F8:0, etc.)

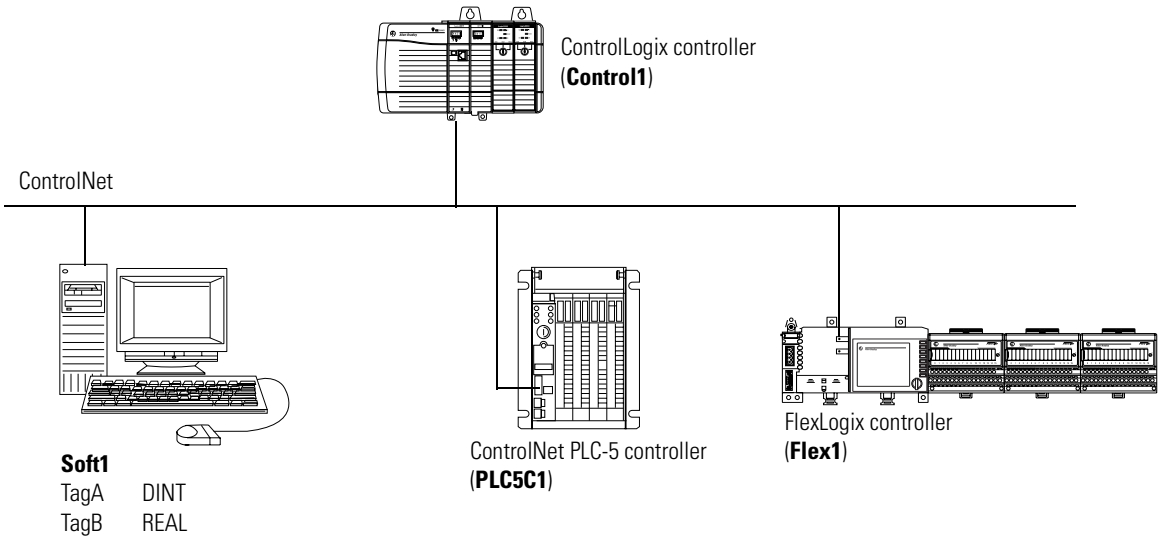
The PLC-5 controller supports logical ASCII addressing so you do not have to map a compatibility file for MSG instructions initiated by a PLC-5 controller. Place the SoftLogix tag name in double quotes (“).

Type of MSG Instruction:	Example Source and Destination:	
PLC-5 writes to SoftLogix	source element	*N7:10*
	destination tag	*“array_1”*
PLC-5 reads from SoftLogix	source tag	*“array_1”*
	destination element	*N7:10*

Example 3: Producing and consuming tags

You can produce and consume tags with any Logix controller the same as you do with a SoftLogix controller. All Logix controllers follow the same requirements for producing and consuming tags. See Example 2 above.

Producing and consuming tags with a ControlNet PLC-5 controller depends on the type of data.



Producing a tag to a ControlNet PLC-5 controller

To produce a tag that a ControlNet PLC-5 controller can consume:

1. Determine the type of data to produce?

If:	And you are producing:	Then:
INT	na	A. Create a user-defined data type that contains an array of INTs with an even number of elements, such as INT[2]. When you produce INTs, you must produce two or more. B. Create a produced tag and select the user-defined data type you created.
DINT or REAL	Only one DINT or REAL value	Create a produced tag and select the DINT or REAL data type, as appropriate.
	More than one DINT or REAL	A. Create a user-defined data type that contains an array of DINTs or REALs, as appropriate. B. Create a produced tag and select the user-defined data type you created.

2. In RSNetWorx software, open the ControlNet configuration for the target ControlNet PLC-5 controller, insert a Receive Scheduled Message and enter the following Message size:

If the produced tag contains:	Then, for the Message size, enter:
INTs	The number of integers in the produced tag
DINTs	Two times the number of DINTs or REALs in the produced tag. For example, if the produced tag contains 10 DINTs, enter 20 for the Message size.
REALs	

3. In the RSNetWorx software, reschedule (save) the network.

The ControlNet PLC-5 controller does not perform type checking. Make sure the PLC-5 data type can correctly receive the SoftLogix produced tag to ensure proper data is being received.

When a ControlNet PLC-5 controller consumes a tag that is produced by a Logix5000 controller, it stores the data in consecutive 16-bit integers. The ControlNet PLC-5 controller stores floating-point data, which requires 32-bits regardless of the type of controller, as follows:

- The first integer contains the upper (left-most) bits of the value.
- The second integer contains the lower (right-most) bits of the value.

To re-construct the floating point data within the ControlNet PLC-5 controller, first reverse the order of the integers and then copy them to a floating-point file.

Consuming a tag from a ControlNet PLC-5 controller

To consume a tag from a ControlNet PLC-5 controller,:

1. In RSNetWorx software, open the ControlNet configuration of the ControlNet PLC-5 controller, insert a Send Scheduled Message.

2. In RSLogix 5000 software, add the ControlNet PLC-5 controller to the Controller Organizer.

3. Create a user-defined data type that contains these members:

Data type:	Description:
DINT	Status
INT[x], where "x" is the output size of the data from the ControlNet PLC-5 controller. (If you are consuming only one INT, no dimension is required.)	Data produced by a ControlNet PLC-5 controller

4. Create a consumed tag with the following properties:

For this tag property:	Type or select:
Tag Type	Consumed
Controller	The ControlNet PLC-5 that is producing the data
Remote Instance	The message number from the ControlNet configuration of the ControlNet PLC-5 controller
RPI	A power of two times the NUT of the ControlNet network. For example, if the NUT is 5ms, select an RPI of 5, 10, 20, 40, etc.
Data Type	The user-defined data type that you created.

5. In the RSNetWorx for ControlNet software, reschedule (save) the network.

Example 3: Total connections required by Soft1

The following table calculates the connections used in this example.

Connection:	Amount:
Soft1 controller to 1784-PCICS card	0
Soft1 controller to remote 1756-CNB module	0
Soft1 controller to remote 1788-CNC card	0
Soft1 controller to remote PLC5C1	1
connected, cached MSG from Soft1 to Control1	1
connected, cached MSG from Soft1 to Flex1	1
connected, cached MSG from Soft1 to PLC5C1	1
produced TagA produced from Soft1 to Control1 consumed by PLC5C1	 1 1
consumed TagB from Control1	1
total connections used:	7

The remote 1756-CNB and 1788-CNC card are configured as “none” for the communication format, so the SoftLogix controller would require a direct connection for any I/O modules connected to these devices that you want in the configuration for the SoftLogix controller.

Example 4: Using SoftLogix as a Gateway

The SoftLogix controller supports bridging over a ControlNet network.

Any SoftLogix MSG instruction that bridges one network has multiple pairs of numbers in its communication path. To construct a communication path:

1. Specify the port where the message exits.

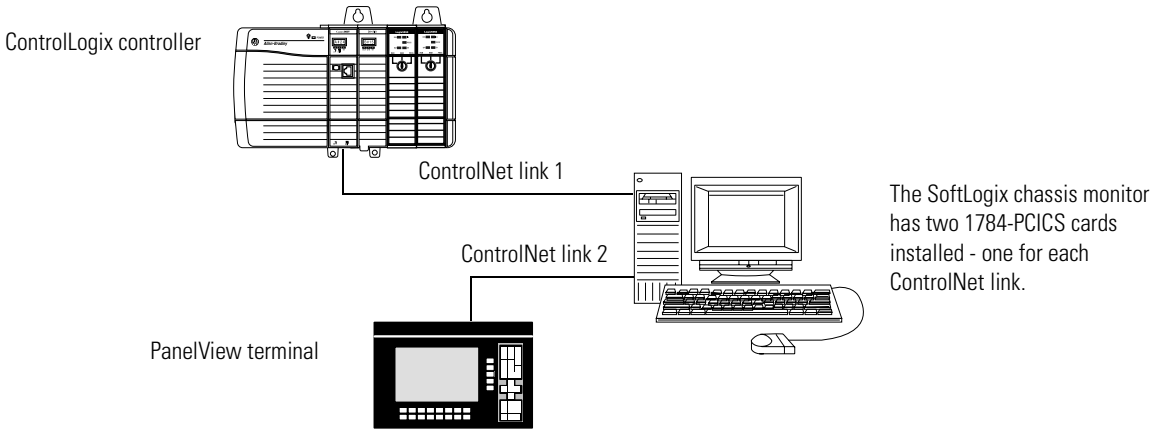
For this port:	Specify:
backplane port	1
DF1 port from the controller	2
ControlNet port from a communication card/module	
Ethernet port from a communication card/module	
DH+ port over channel A from a 1756-DHRIO module	
DH+ port over channel B from a 1756-DHRIO module	3

2. Specify the next device.

For a device on a:	Specify:
ControlLogix backplane	slot number
DF1 network	station address (0-254)
ControlNet network	node number (1-99 decimal)
DH+ network	node number (1-77 decimal)
Ethernet network	IP address

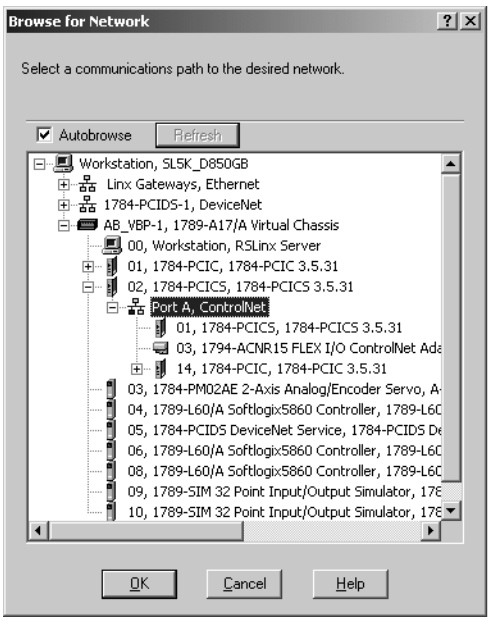
3. Repeat steps 1 and 2 until you specify the target device.

In the following example, the ControlLogix controller can remotely access a PanelView terminal over ControlNet links. The SoftLogix chassis monitor resides on the computer. A SoftLogix controller is not required for the gateway - you only need a 1784-PCICS card for each ControlNet link.



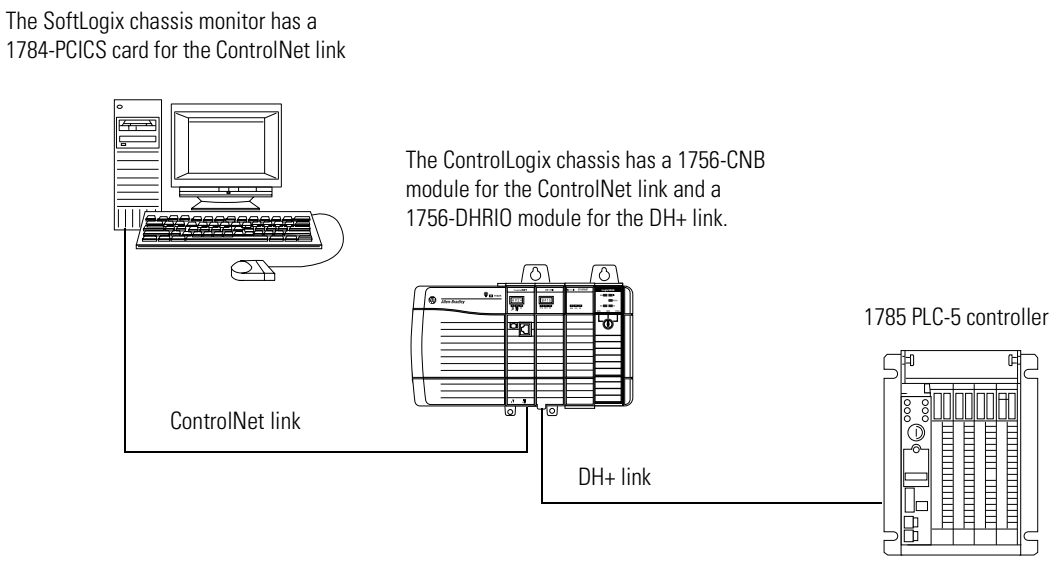
If you want to select the 1784-PCICS card from an online list of available devices, such as Browse Network in RSNetWorx for ControlNet software, select the 1784-PCICS card in the virtual chassis.

Select the 1784-PCICS card from within the virtual chassis. →



Example 5: Using ControlLogix as a Gateway

The SoftLogix controller can use the ControlLogix chassis to remotely access devices on ControlNet, Ethernet, DH+, or Universal Remote I/O networks.



Communicating with Devices on a DeviceNet Link

Using This Chapter

Configuring Your System for a DeviceNet Link

For the SoftLogix controller to operate on a DeviceNet network, you need:

- a 1784-PCIDS DeviceNet communication card
- RSLinx software to install the DeviceNet communication driver
- RSLinx software to install the virtual backplane driver

  You only install the virtual backplane driver once on the computer where you run the SoftLogix controller. This chapter assumes you have already installed the driver. For an example of installing the driver, see chapter 1, “Getting Started.”

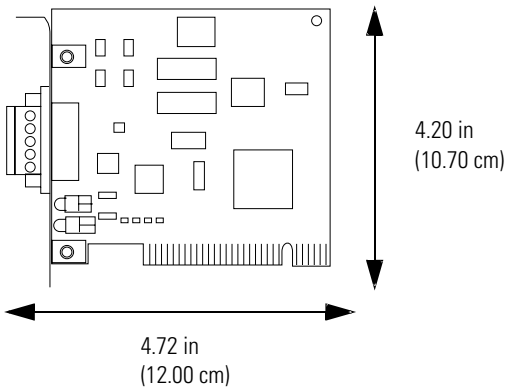
- RSLogix5000 programming software to configure the communication card as part of the SoftLogix system
- RSNetWorx for DeviceNet software to configure the devices on the network
- IOLinx software must be installed for the SoftLogix controller to be able to read and write I/O data

Step 1: Install the hardware

Make sure the 1784-PCIDS communication card is properly installed in the computer. You need to:

- Install the card in any PCI slot within the computer.

  It does not matter which PCI slot you use for the communication card. The PCI slot in the computer **does not** correspond to the backplane slot in the SoftLogix chassis. You use the SoftLogix chassis monitor to place the communication card in a specific backplane slot (see the next page).



- Install IOLinx software so the SoftLogix controller can use the 1784-PCIDS communication card to control DeviceNet I/O.
- Make a label to place on the front of the card, or use a pen to write on the front of the card. The label should include the serial number of the card and a name you can use to identify the card from any others you might install in the computer

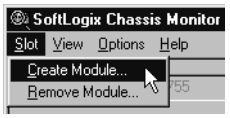
  Remember the serial number of each communication card you install. You use the serial number to identify which card you want in which slot of the SoftLogix chassis.

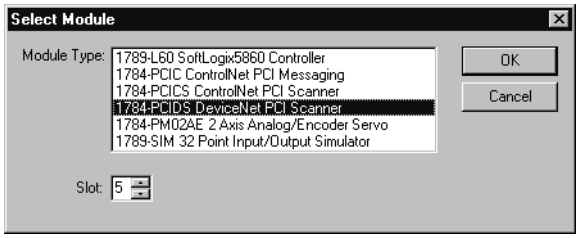
For more information about installing a 1784-PCIDS communication card, see the *DeviceNet PCI Interface Card Installation Instructions*, publication 1784-5.31.

Step 2: Create the communication card in the chassis

Before you can connect the SoftLogix system to the DeviceNet network, you must create the 1784-PCIDS card as part of the SoftLogix chassis.

1. From the SoftLogix chassis monitor, select Slot → Create Module or right click the appropriate slot and select Create. Select the 1784-PCIDS card.

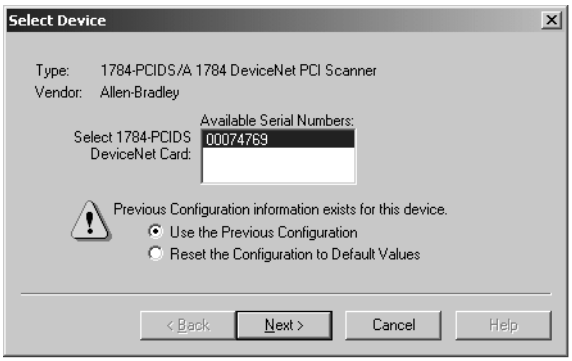




Specify the backplane slot number.

Click OK

2. Select the serial number of the 1784-PCIDS card you want.



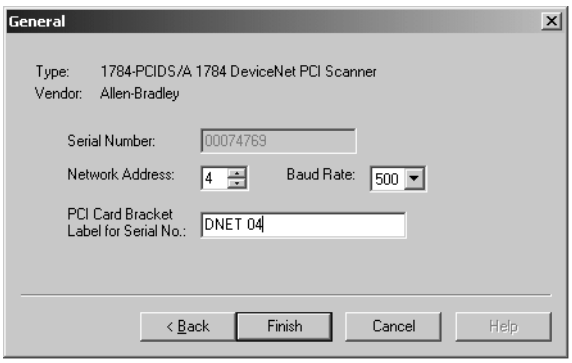
Select the serial number of the card.

If you previously configured the 1784-PCIDS card that you selected by serial number, the chassis monitor remembers the configuration from the last time you used the card (whether in the same or different slot).

Click Next

3. Specify configuration settings for the 1784-PCIDS card:

- specify the node address (MAC ID) on the DeviceNet network
- specify the data rate
- enter the label name for the card (this is the name you wrote on the label of the card to help you identify the card from others in the same computer)

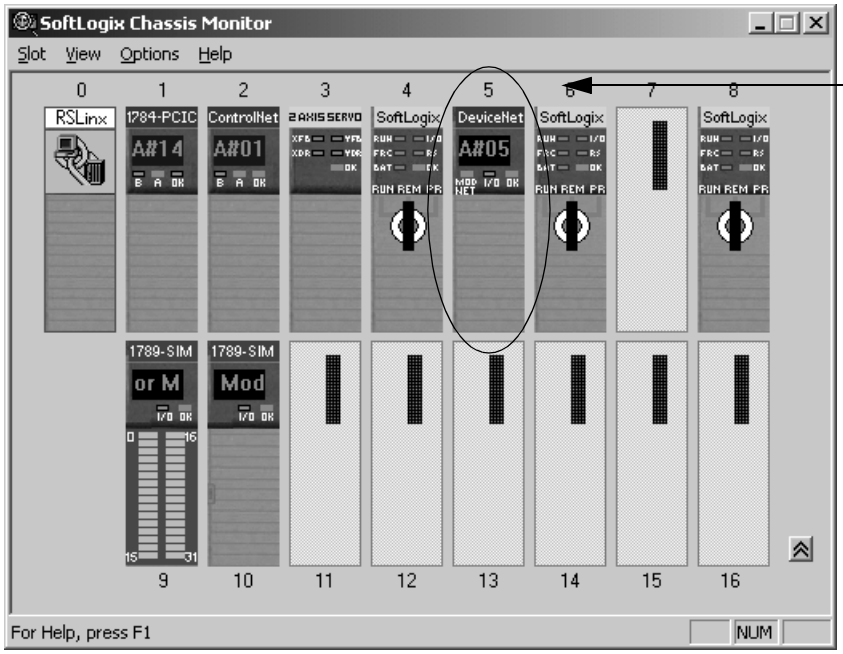


Click Finish

You can specify any slot number greater than 0 for the communication card. RSLinx software resides in slot 0.

IMPORTANT When you add a 1784-PCIDS card to the chassis monitor, the card must be connected to a valid, powered DeviceNet network. And, the baud rate you choose for card must be same as the baud rate for the DeviceNet network. Otherwise, the card will fail to insert in the chassis monitor.

The chassis monitor shows the 1784-PCIDS card as a virtual module in the SoftLogix chassis. The LEDs on the virtual monitor emulate a 1756-DNB communication module.

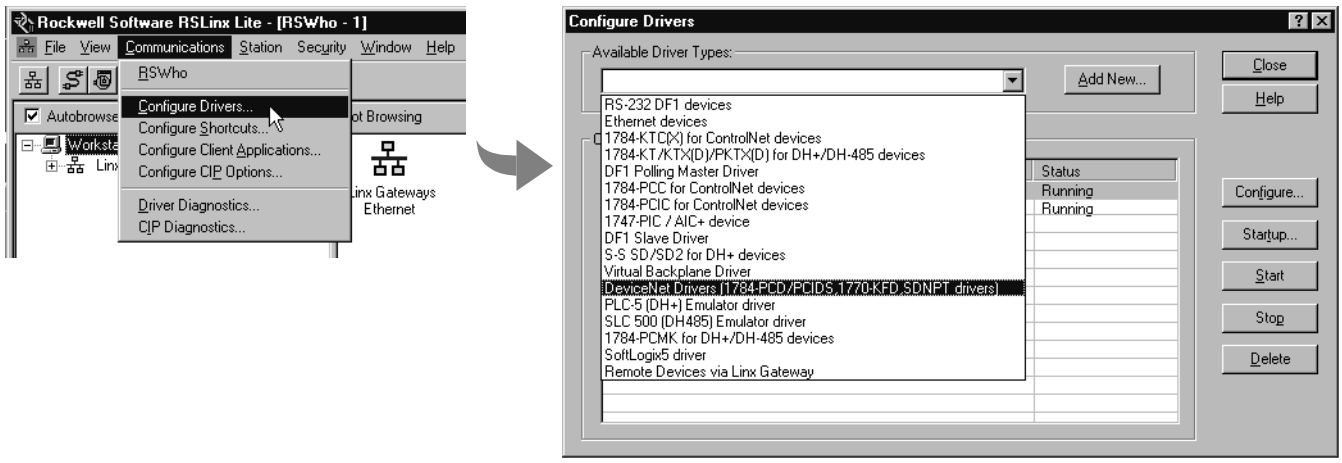


This chassis monitor has a 1784-PCIDS card installed in slot 5.

Step 3: Install the communication driver

Use RSLinx software to configure the DeviceNet communication driver for the 1784-PCIDS communication card.

1. In RSLinx software, select Configure Driver. Select the appropriate driver.



The device settings will be grayed out because you specified the baud rate and node address when you created the module in the SoftLogix chassis.

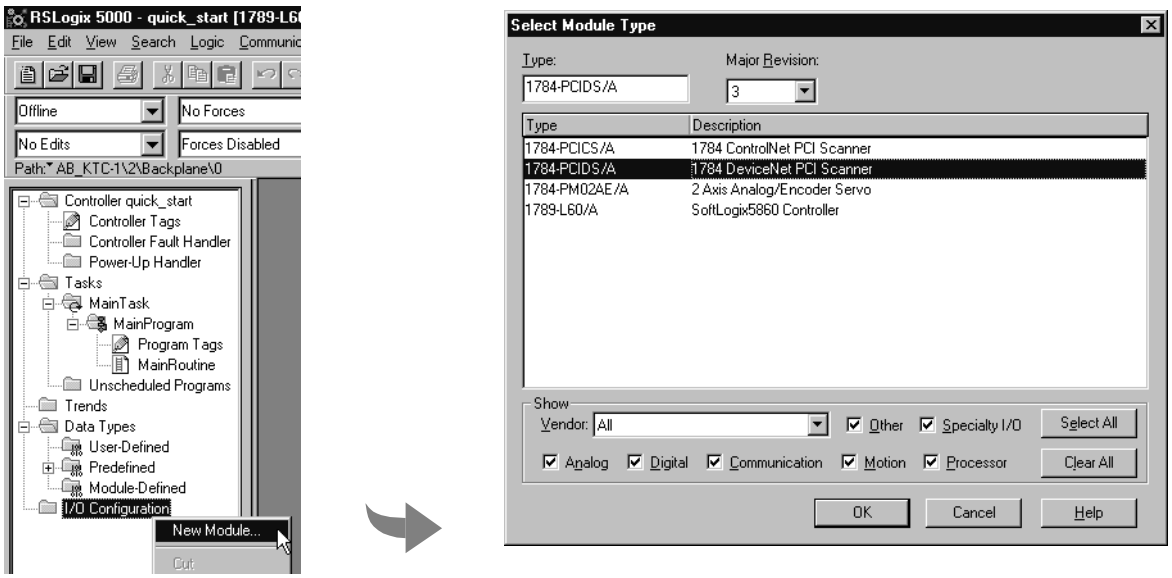
You only have to install the DeviceNet communication driver on the computer that you use to run RSNetWorx for DeviceNet. This example assumes that you are running the SoftLogix controller and RSNetWorx on the same computer.

Step 4: Configure the communication card as part of the project

Use RSLogix 5000 programming software to map the 1784-PCIDS communication card as part of the SoftLogix project. In the Controller Organizer, add the communication card to the I/O Configuration folder.

1. In RSLogix 5000 programming software, select the I/O Configuration folder.

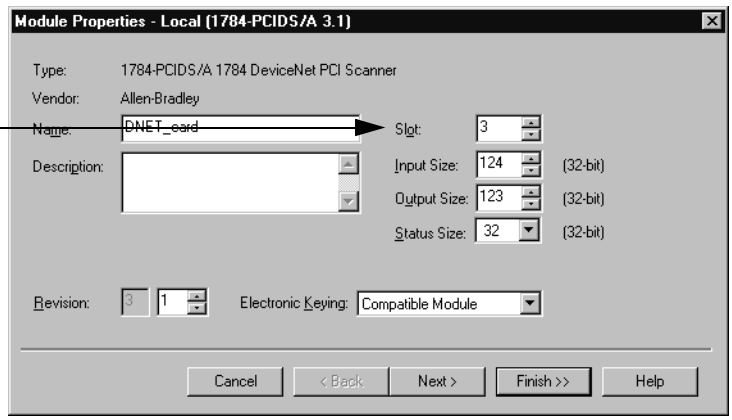
2 Right-click to select New Module and add a 1784-PCIDS communication card.



3. Specify the appropriate communication card settings.

4. This must be the same slot number you specified on the SoftLogix chassis monitor.

Make sure your selections for Input Size, Output Size, and Status Size are big enough to hold the data you expect. If the sizes are too small, data will be truncated. If the sizes are too big, the software zero pads the data blocks.



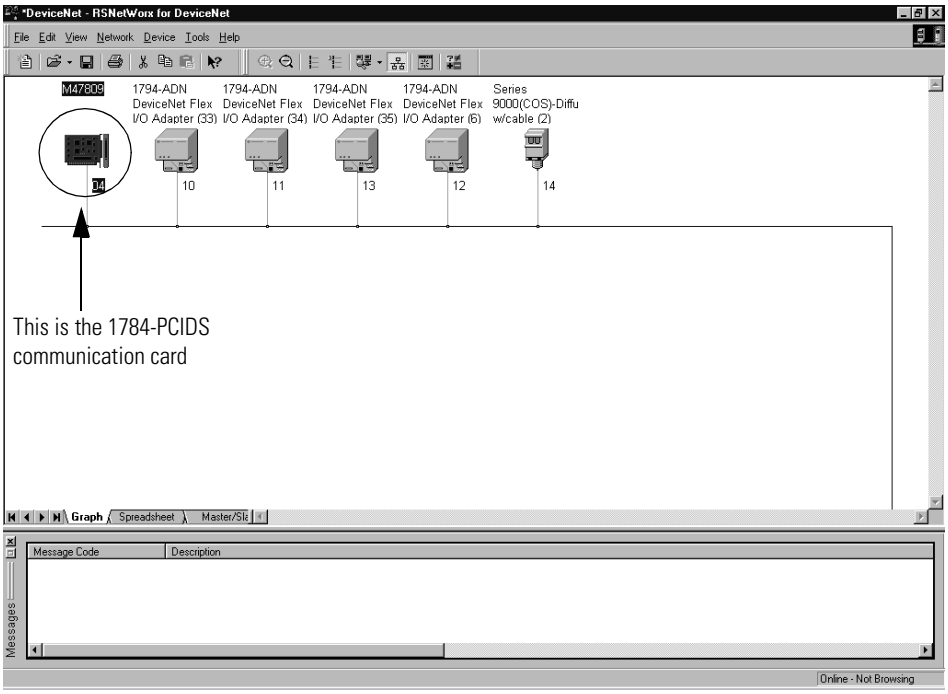
The virtual backplane driver must be installed via RSLinx software before you can download a project to the SoftLogix controller.

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

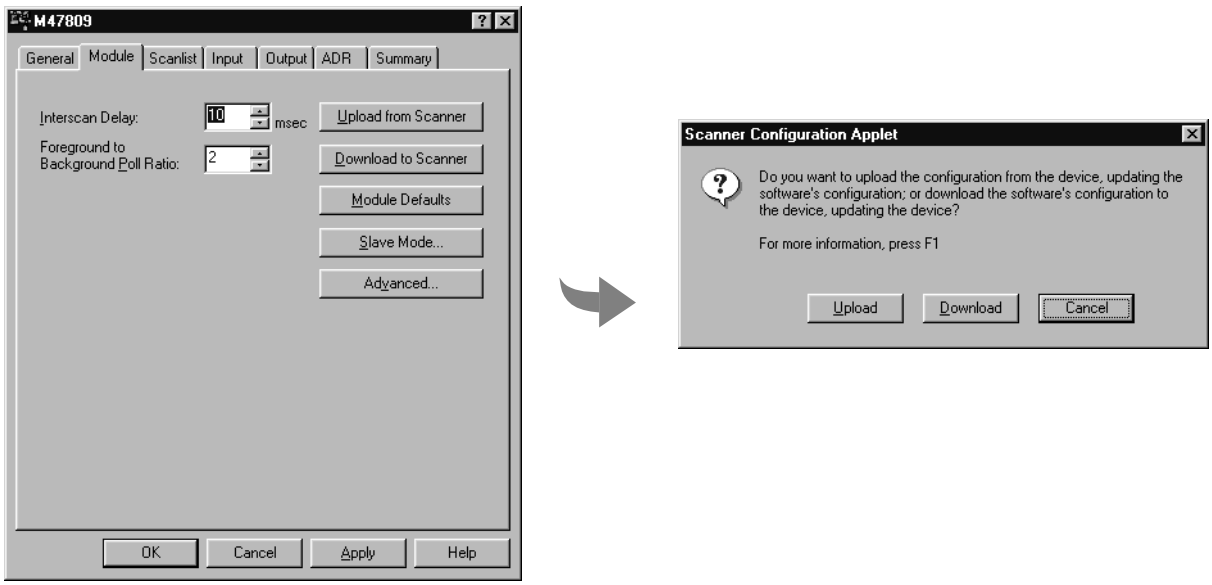
Step 5: Define the scan list

Use RSNetWorx for DeviceNet software to define the scan list. The project must already be downloaded from RSLogix 5000 programming software to the controller and the controller must be in Program or Remote Program mode.

1. In RSNetWorx software, go online, enable edits, and survey the network.



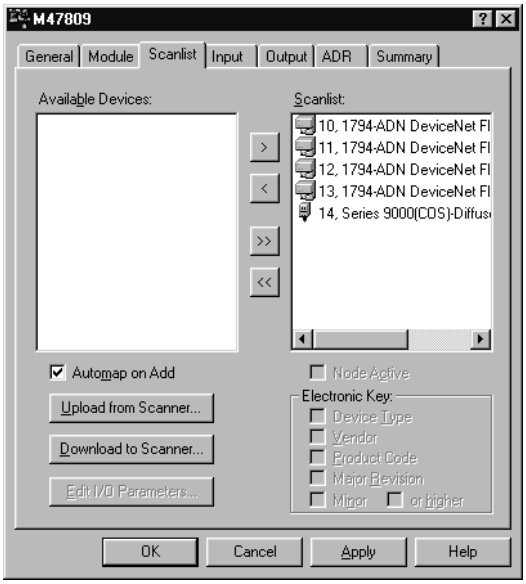
2. Double-click the 1784-PCIDS card and select the Module tab to configure the card. Upload the network information when prompted.



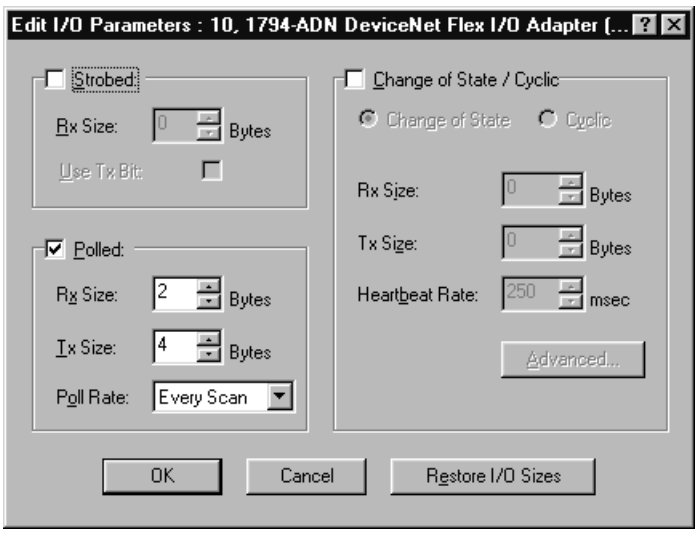
Every device on the network must be in Program or Remote Program mode for the software to re-write all of its connections. If a device is not in the correct mode, the software prompts you to let it change the device's mode.

continued

3. Use the ScanList tab to define the scanning order of the DeviceNet devices.

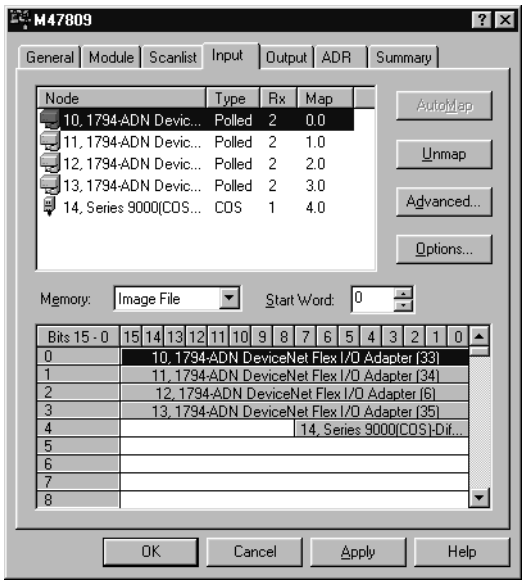


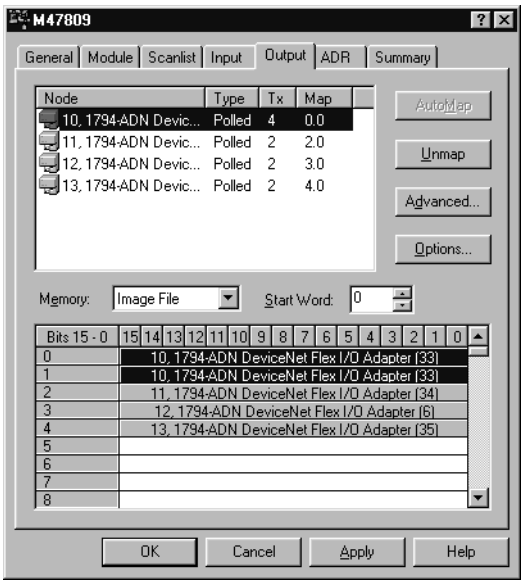
4. Click Edit I/O Parameters to define how many inputs (Rx) and Outputs (Tx) you expect from each DeviceNet device.



continued

5. Use the ScanList tab to define the scanning order of the DeviceNet devices.





If you place the SoftLogix5800 controller in Program mode with DeviceNet I/O currently mapped through a 1784-PCIDS module, and then you use RSNetWorx to change the data mapping on the network, the controller does not detect this change until the 1784-PCIDS module is reset. You can reset the module in the RSLogix 5000 Controller Organizer. Right-mouse click over the module and select Properties; then select the Module Info tab and click the Reset Module button. You can also reset the module by removing and re-inserting the module in the SoftLogix chassis. You can reset the module while the SoftLogix controller is running. The connections are automatically re-established after the 1784-PCIDS module is reset.

ATTENTION

Do not reset a module that is currently being used for control. The connection to the module will be broken and control might be interrupted.

The SoftLogix controller supports 32-bit words of data. You can have 124 words of input data, 123 words of output data, and 32 words of device status data. How you configure the DeviceNet devices determines how many words you use per device.

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 word-align the device data. While this might simplify the organization of the data, it might also limit the data you have available.

Accessing DeviceNet I/O

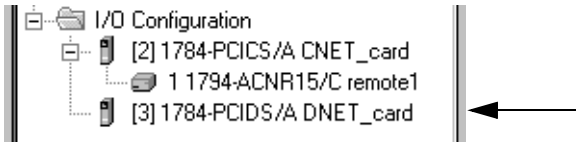
I/O information is presented as a structure of multiple fields, which depend on the specific features of the I/O module. The name of the structure is based on the location of the I/O module in the system. Each I/O tag is automatically created when you configure the I/O module through the programming software. Each tag name follows this format:

Location:SlotNumber:Type.MemberName.SubMemberName.Bit

where:

This address variable:	Is:
Location	Identifies network location LOCAL = identifies communication card within the computer
SlotNumber	Slot number of I/O module in its chassis
Type	Type of data I = input O = output C = configuration S = status
MemberName	Specific data from the I/O module; depends on the type of data the module can store For example, Data and Fault are possible fields of data for an I/O module. Data is the common name for values the are sent to or received from I/O points.
SubMemberName	Specific data related to a MemberName.
Bit (optional)	Specific point on the I/O module; depends on the size of the I/O module (0-31 for a 32-point module)

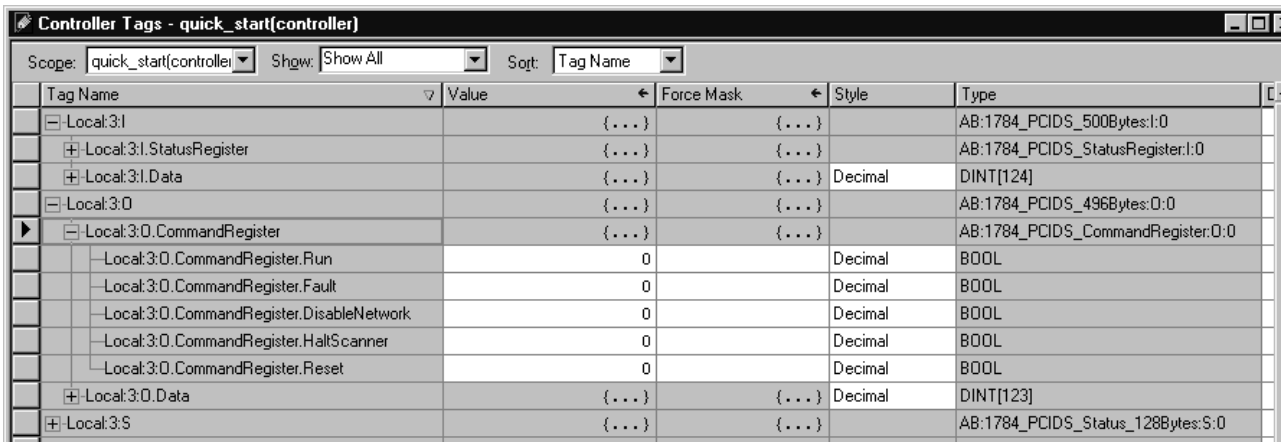
EXAMPLE



The 1784-PCIDS card in this example is in slot 3.

The data for a 1784-PCIDS card is always configured as a rack-optimized connection.

The rack-optimized connection creates a DINT element for each possible I/O module connected to the device in slot 3, “Local:3.” The array Local:3:I.Data contains the possible input elements; the Local:3:O.Data contains the possible output elements.



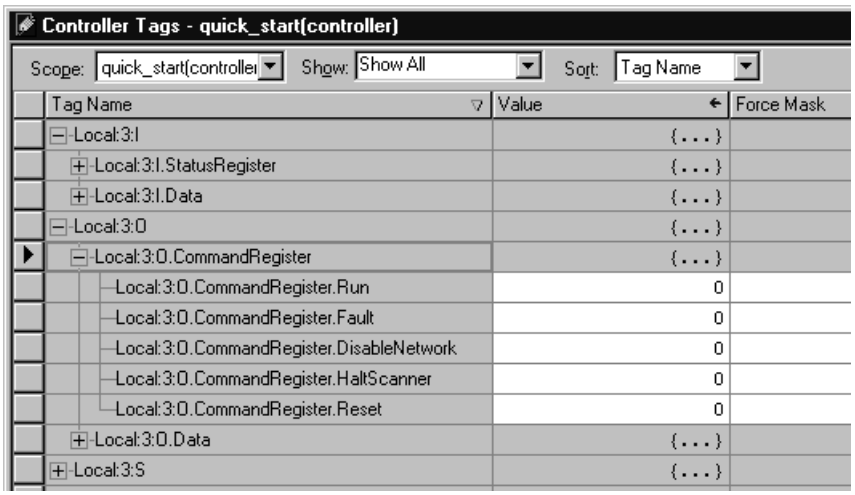
Tag Name	Value	Force Mask	Style	Type
Local:3:I	{...}	{...}		AB:1784_PCIDS_500Bytes:I:0
Local:3:I.StatusRegister	{...}	{...}		AB:1784_PCIDS_StatusRegister:I:0
Local:3:I.Data	{...}	{...}	Decimal	DINT[124]
Local:3:O	{...}	{...}		AB:1784_PCIDS_496Bytes:O:0
Local:3:O.CommandRegister	{...}	{...}		AB:1784_PCIDS_CommandRegister:O:0
Local:3:O.CommandRegister.Run	0		Decimal	BOOL
Local:3:O.CommandRegister.Fault	0		Decimal	BOOL
Local:3:O.CommandRegister.DisableNetwork	0		Decimal	BOOL
Local:3:O.CommandRegister.HaltScanner	0		Decimal	BOOL
Local:3:O.CommandRegister.Reset	0		Decimal	BOOL
Local:3:O.Data	{...}	{...}	Decimal	DINT[123]
Local:3:S	{...}	{...}		AB:1784_PCIDS_Status_128Bytes:S:0

The index number on the array element refers to the same numbered word mapped to the device in RSNetWorx for DeviceNet. Depending on the device, there can be several words mapped to on device. You can create aliases to the elements you actually use to more identify the data you need.

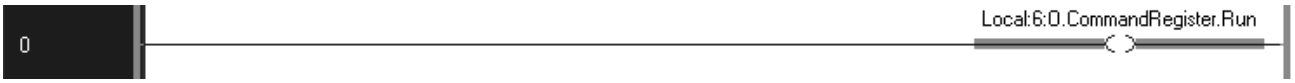
Placing the Communication Card in Run Mode

To place the 1784-PCIDS card in Run mode, your program logic needs to set the CommandRegister.Run bit in the output word for the card.

Set this bit →



For example:



Using the CommandRegister bits

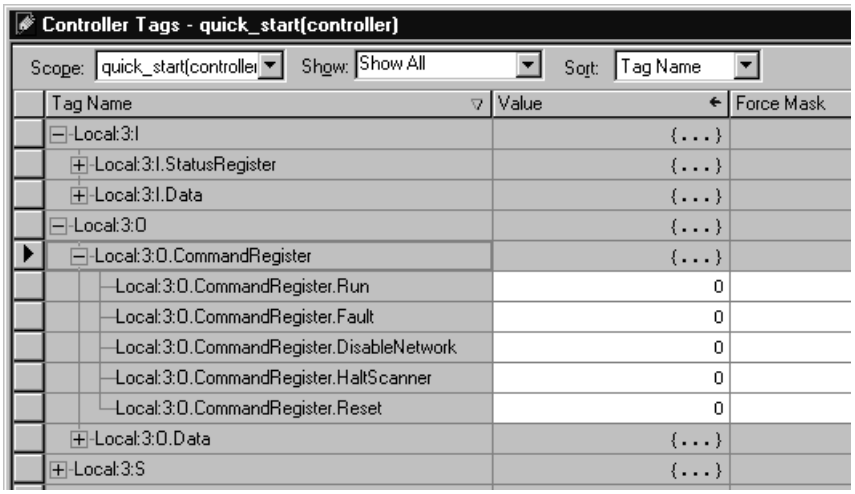
The following table describes how the 1784-PCIDS card uses the CommandRegister bits.

When CommandRegister.Run is set to:	The 1784-PCIDS card:
zero (0)	is in Idle mode In Idle mode, the card still receives inputs from its slave devices on the network, but the card does not send active output data to the devices.
one (1)	is in Run mode In Run mode, the card sends active outputs on the network and receives inputs.

Monitoring the 1784-PCIDS Card

The input data for the 1784-PCIDS card includes a StatusRegister.

StatusRegister bits →



The following table describes how the 1784-PCIDS card uses the StatusRegister bits.

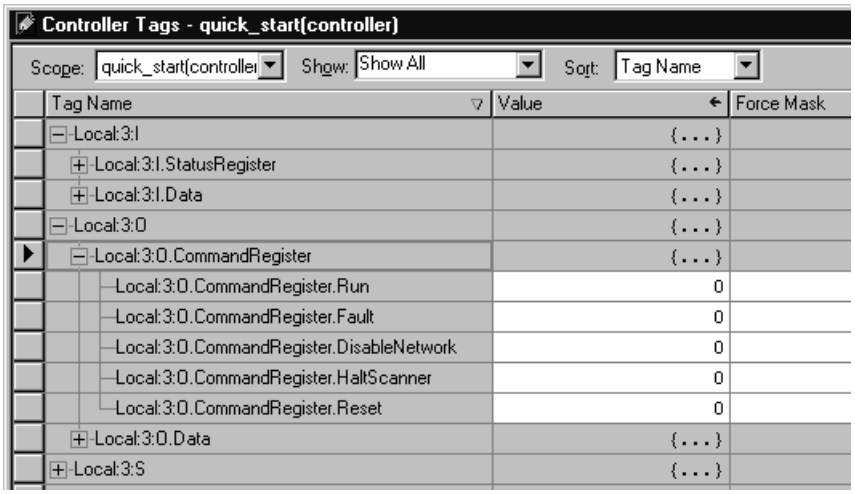
StatusRegister Bit:	Description:
StatusRegister.Run	This bit echoes the CommandRegister.Run bit to determine if card is in Run or Idle mode. A 0 in this bit means the card is in Idle. A 1 means the card is in Run mode.
StatusRegister.Fault	This bit identifies whether the card is in Fault mode. The SoftLogix controller sets this bit based on the corresponding IOLinx status.
StatusRegister.DisableNetwork	The SoftLogix controller does not use this bit. The controller clears this bit to 0.
StatusRegister.DeviceFailure	This bit determines if general communication is OK between the card and its slave nodes. A node falling off the network or other communication problems to any device on the card's scan list sets this bit to 1. This bit is used in conjunction with the DeviceFailure table in the Status section to determine which node(s) are having communication problems. A 0 in this bit means that all the slave nodes are being successfully communicated to. A 1 means the card has at least one device with communication problems.
StatusRegister.Autoverify	This bit determines if the data Transmit and Receive sizes in the scan list are correct. Any node whose data sizes don't match the sizes defined in the scan list cause the bit to be set to 1. This bit is used in conjunction with the AutoVerify table in the Status section to determine which node(s) have incorrect data sizes. A 0 in this bit means that all the slaves have correct data sizes. A 1 means the card has at least one device on its scan list with an incorrect data size.

StatusRegister Bit:	Description:
StatusRegister.CommFailure	This bit identifies whether a channel wide communication fault is happening with the card. For example if the card detects severe communication problems on the network it will go into a Bus Off condition. This also cause the StatusRegister.CommFailure bit to turn on. A 0 in this bit means that the card is communicating correctly. A 1 means the card detected a channel wide communication problem.
StatusRegister.DupNodeFail	This bit shows if the card is attempting to go online on a DeviceNet network with the same node number as an existing device on the network. A 0 in this bit means that the card has NOT detected another node on the network with the same node number as the card. A 1 means that the card has the same node number as an existing device on the network.
StatusRegister.DnetPowerDetect	This bit shows if the card has detected that the DeviceNet 24VDC power is connected to its network connector and is also energized. A 0 in this bit means that the card has detected DeviceNet power on its network connector. A 1 means that the card has NOT detected DeviceNet power on its network connector.

Using the Status data

The status data for the 1784-PCIDS card includes several elements.

Status data elements

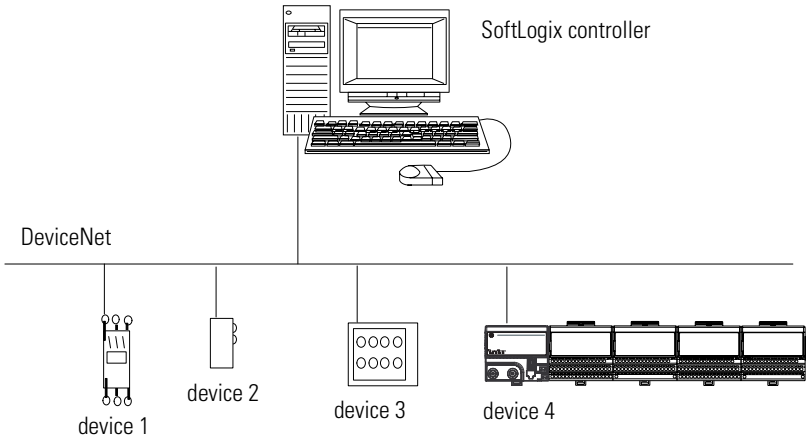


The following table describes the status data for a 1784-PCIDS card.

Status Element:	Description:
S.ScanCounter	This 32-bit word is incremented every time the card completes a network scan. By reading this value and counting how many network updates are done in a certain time, you can calculate an average scan time.
S.DeviceFailureRegister	This data area is an array of 8 bytes that make a 64-bit table. There is one bit for every one of the 64 possible node numbers on the network. The bit associated with a node number on the scan list is set to 1 if that node number is having communication problems.
S.AutoVerifyRegister	This data area is an array of 8 bytes that make a 64-bit table. There is one bit for every one of the 64 possible node numbers on the network. The bit associated with a node number on the scan list is set to 1 if that node number has a transmit and/or receive data size that does not match the scan list.
S.DeviceIdleRegister	This data area is an array of 8 bytes that make a 64-bit table. There is one bit for every one of the 64 possible node numbers on the network. The bit associated with a node number on the scan list is set to 1 if that node is not sending back input data to the card. This normally means that the node is in some kind of idle mode in which it stops sending output data back to the card's input table.
S.DeviceActiveRegister	This data area is an array of 8 bytes that make up a 64 bit table. There is one bit for every one of the 64 possible node numbers on the network. The bit associated with a node number on the scan list will go to a 1 if that node number has an active scan list entry in the 1784-PCIDS. This means that the 1784-PCIDS is communicating with that node.
S.DeviceStatusDisplay	This data area is an array of 4 bytes. There is one byte associated with each of the 4 characters of the alphanumeric display on the SoftLogix chassis monitor. Reading these 4 bytes as ASCII characters to determine the exact message being displayed on the SoftLogix chassis monitor.

Example: SoftLogix Controller and I/O

In the following example, one SoftLogix controller controls I/O through a 1784-PCIDS communication card.



This example has a SoftLogix controller controlling four DeviceNet devices. The controller automatically creates a rack-optimized connection for the I/O data.

The tag name for the rack-optimized array tag is based on the slot number of the 1784-PCIDS card. For example, if you install the 1784-PCIDS card in slot 3 of the controller, the software automatically creates Local:3:I and Local:3:O data structures.

Creating alias tags

You might want to create alias tags to better represent the elements of the input and output array tags. An alias for an I/O point:

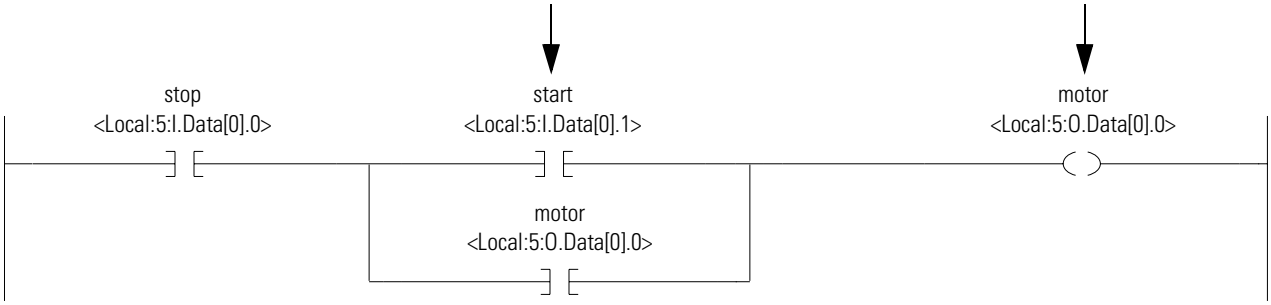
- provides a descriptive name for the device that is wired to the point
- represents the value of the point. When one changes, the other reflects the change.

When you enter an alias tag into programming logic, the programming software displays the original tag, along with the alias.

EXAMPLE The following logic was initially programmed using descriptive tag names, such as *start* and *motor*. Later, the tags were converted to aliases for the corresponding I/O devices.

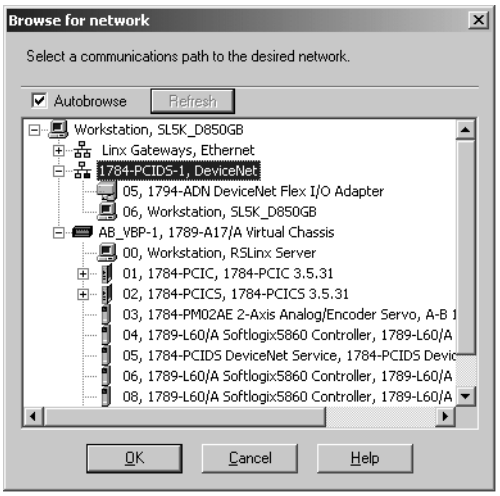
start is an alias for the push button at bit 1 of word 0 of the module in slot 5 of the local chassis. When the push button is on, *start* is on.

motor is an alias for the starter contactor at bit 0 of word 0 of the module in slot 5 of the local chassis. When *motor* turns on, the starter contactor turns on.



If you want to select the 1784-PCIDS card from an online list of available devices, such as Browse Network in RSNetWorx for DeviceNet software, select the 1784-PCIDS card from outside of the virtual chassis.

Select the 1784-PCIDS card from outside of the virtual chassis.



Notes:

Programming Over an Ethernet Link

Using This Chapter

For information about:	See page
Configuring your system for an Ethernet link	6-1
Example: Workstation remotely connected to a SoftLogix controller	6-5

Configuring Your System for an Ethernet Link

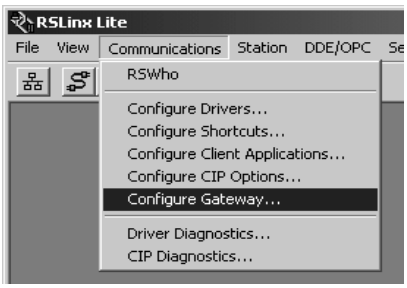
To be able to program the SoftLogix controller remotely over Ethernet, you need:

- the remote computer where the SoftLogix controller resides must have an Ethernet communication card.
- the computer where the RSLogix 5000 programming software resides must have an Ethernet communication card.
- RSLinx software to configure the Ethernet communication driver for each computer
- RSLogix5000 programming software to program the remote SoftLogix controller

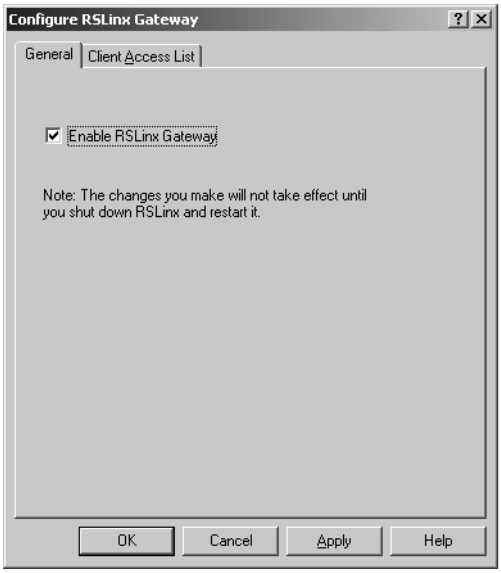
Step 1: Enable RSLinx Gateway for the controller

The SoftLogix controller comes with enough functionality of RSLinx gateway to allow you to program the controller remotely over an Ethernet link.

1. From RSLinx software on the computer with the controller, enable RSlinx gateway.



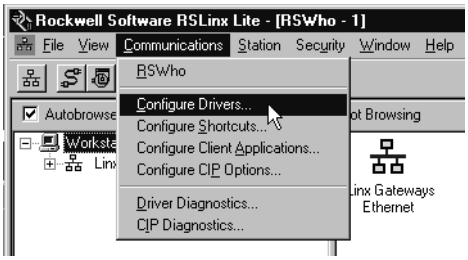
On the General tab, enable gateway.

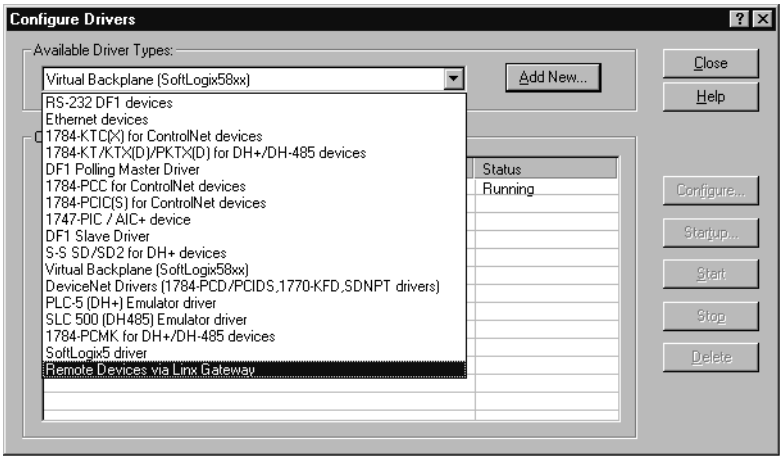


Click OK

Step 2: Configure the Ethernet communication driver on the computer with the programming software

1. In RSLinx software, select Configure Driver. Select the Remote Devices via Linx Gateway driver.

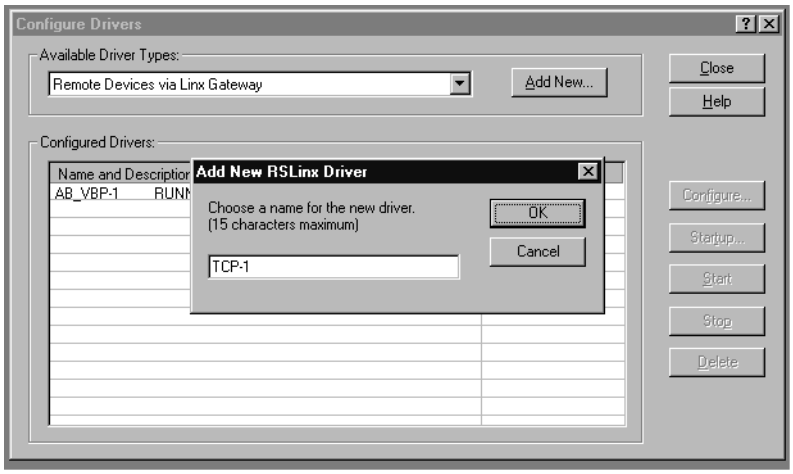




Click Add New

2. Add the Ethernet driver.

Enter a name for the driver. You must create one driver for each SoftLogix controller you want to program remotely.



Click OK

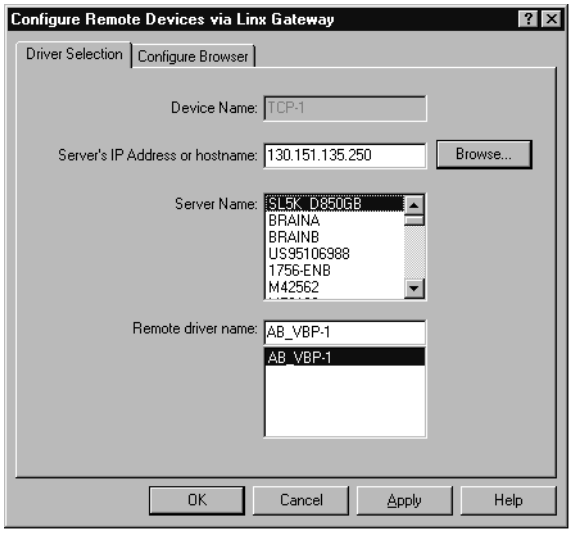
continued

After you create the Ethernet driver, configure it to correspond to the virtual chassis on the SoftLogix computer that you want to access.

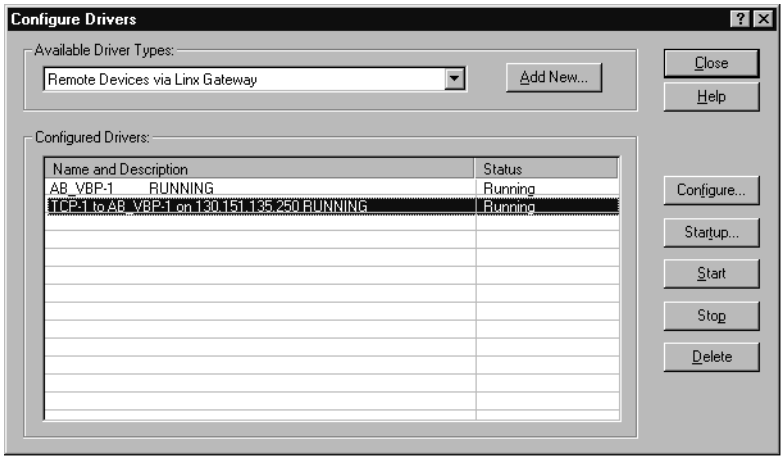
3. Press the Configure button to configure the Ethernet driver.

Select the Ethernet address (or server name) of the SoftLogix controller you want to program.

Click OK



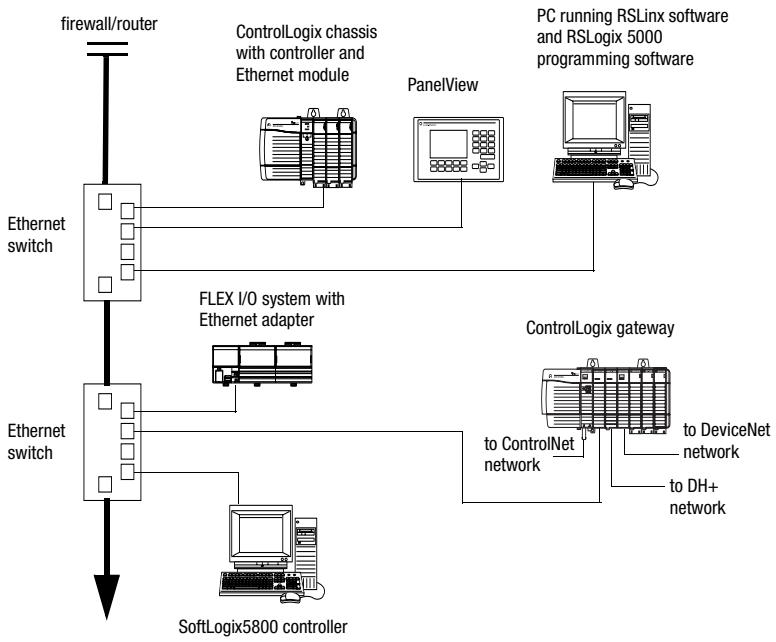
4. The Ethernet driver is now available and you can select the SoftLogix controller from Who Active in RSLogix 5000 programming software.



After you add the Ethernet driver, you can path to all the SoftLogix controllers that are running in various slots on the remote SoftLogix computer.

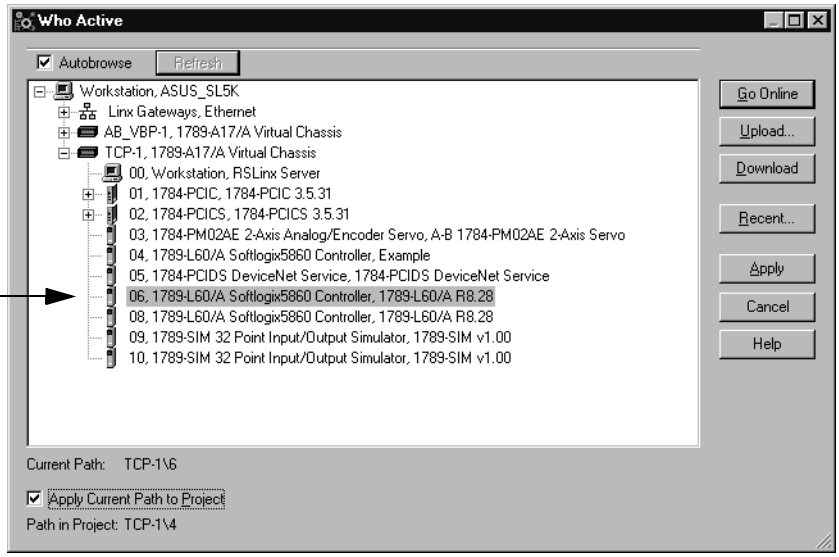
Example: Workstation Remotely Connected to a SoftLogix Controller

In the following example, a workstation remotely connects to a SoftLogix controller over an Ethernet link.



Use Who Active in RSLogix 5000 programming software to select the controller you want to program.

Select the controller from the Ethernet driver's remote virtual chassis.



Notes:

Communicating with Devices on a Serial Link

Using This Chapter

IMPORTANT Limit the length of serial (RS-232) cables to 15.2m (50 ft.).

Configuring Your System for a Serial Link

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

- the computer where the SoftLogix controller resides must have a serial port
- RSLinx software to configure the serial communication driver

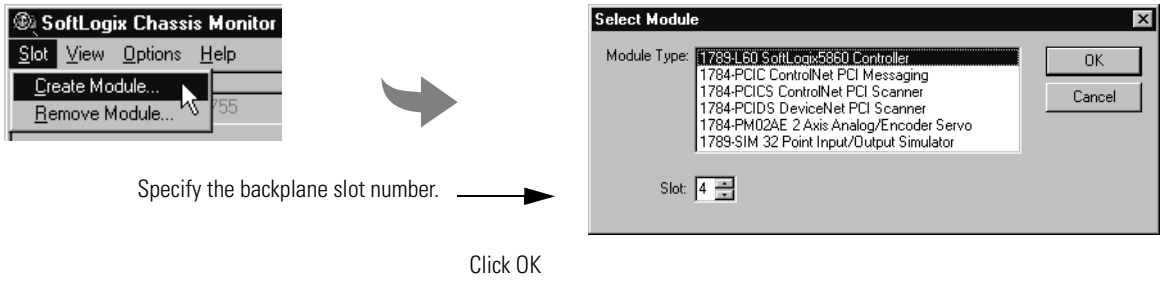
  If a remote computer communicates with the SoftLogix controller via a serial connection, the remote computer must have the serial driver installed. The computer with the SoftLogix controller does not need the serial driver to connect to other devices over a serial link.

- RSLogix5000 programming software to configure the serial port of the controller

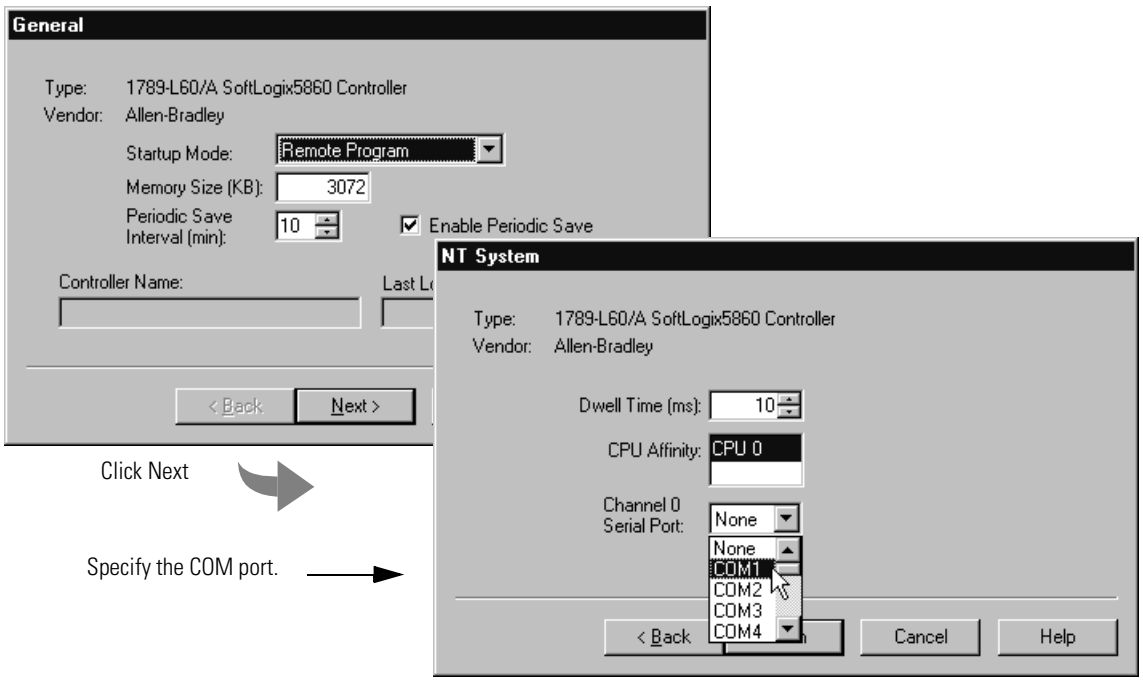
Step 1: Configure the serial port

Use the SoftLogix chassis monitor to select which COM port to use for serial communications. The controller supports only one COM port for DF1 communications.

1. From the SoftLogix chassis monitor, select Slot → Create Module or right click the appropriate slot and select Create. Select the controller.



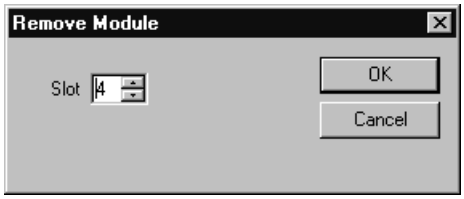
2. Specify configuration settings for the controller. On the second window, select the COM port for serial communications.



Changing the COM port setting

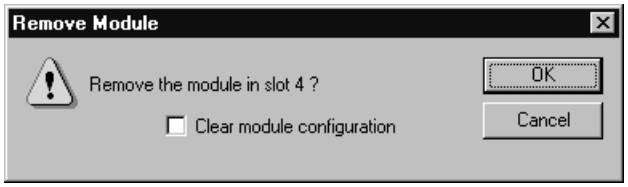
Once you select a COM port for the controller, you can only change the setting by removing the controller from the chassis and reinstalling the controller.

1. From the SoftLogix chassis monitor, select Slot → Remove Module. Accept the chassis monitor prompts to remove the controller

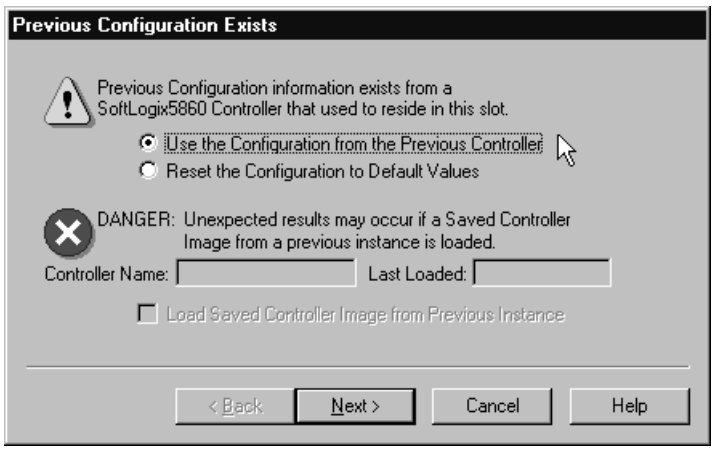


Click OK

Do not clear the module configuration, unless you want to enter all new information.



2. In the same slot, re-add the controller. The chassis monitor prompts whether to use the same configuration that was previously installed.



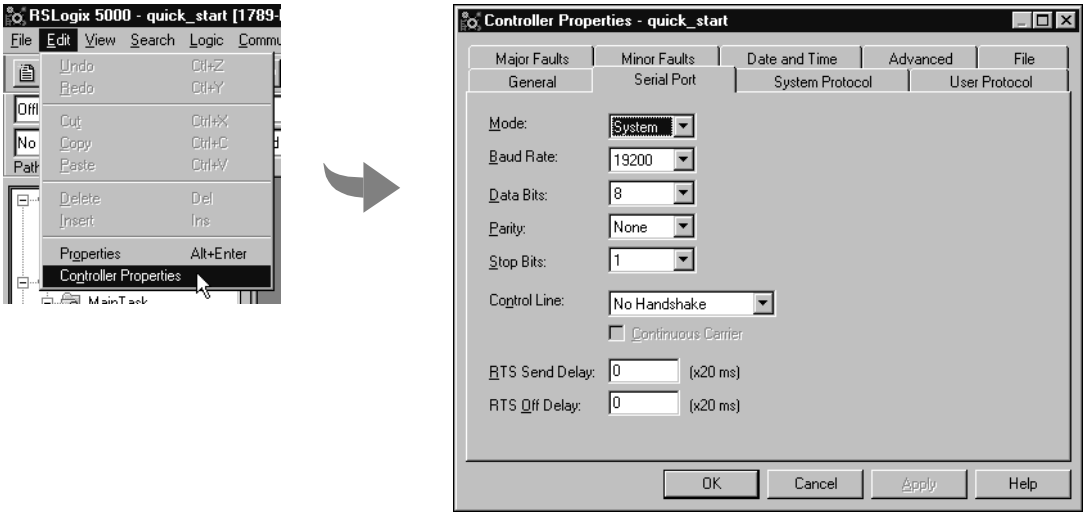
Click Next

3. Follow the steps on the previous page to select a COM port for serial communications.

Step 2: Configure the serial port of the controller

1. In RSLogix 5000 software, select Edit → Controller Properties.

2. On the Serial Port tab, specify the appropriate serial communication settings.



3. On the System Protocol tab, select the appropriate DF1 communication mode for point-to-point or master/slave communications. Or on the User Protocol tab, select ASCII to communicate with an ASCII device.

Specifying serial port characteristics

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

Characteristic:	Description (default is shown in bold):
Mode	Select **System** (for DF1 communication) or User mode (for ASCII communication).
Baud rate	Specifies the communication rate for the serial port. Select a baud rate that all devices in your system support. Select 110, 300 600, 1200, 2400, 4800, 9600, or **19200** Kbps.
Data bits	Specifies the number of bits per message packet. Select **8**.
Parity	Specifies the parity setting for the serial port. Parity provides additional message-packet error detection. Select **None** or Even.
Stop bits	Specifies the number of stop bits to the device with which the controller is communicating. Select **1** or 2.

Characteristic:	Description (default is shown in bold):
Control line	Specifies the mode in which the serial driver operates. Select **No Handshake**, Full-Duplex, Half-Duplex with Continuous Carrier, or Half-Duplex without Continuous Carrier. If you are not using a modem, select No Handshake If both modems in a point-to-point link are full-duplex, select Full-Duplex for both controllers. If the master modem is full duplex and the slave modem is half-duplex, select Full-Duplex for the master controller and select Half-Duplex with Continuous Carrier for the slave controller. If all the modems in the system are half-duplex, select Half-Duplex without Continuous Carrier for the controller.
RTS send delay	Enter a count that represents the number of 20msec periods of time that elapse between the assertion of the RTS signal and the beginning of a message transmission. This time delay lets the modem prepare to transmit a message. The CTS signal must be high for the transmission to occur. The range is **0**-32767 periods.
RTS off delay	Enter a count that represents the number of 20msec periods of time that elapse between the end of a message transmission and the de-assertion of the RTS signal. This time delay is a buffer to make sure the modem successfully transmits the entire message. The range is **0**-32767 periods. Normally leave at zero.

Specifying system protocol characteristics

The available system modes are:

Use this mode:	For:	See page:
DF1 point-to-point	communication between the controller and one other DF1-protocol-compatible device. This is the default system mode. This mode is typically used to program the controller through its serial port.	7-7
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 0-254. 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).	7-9
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. 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 2-255 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.	7-9
User mode	communicating with ASCII devices This requires your program logic to use the ASCII instructions to read and write data from and to an ASCII device.	7-12

Monitoring the Controller LEDs

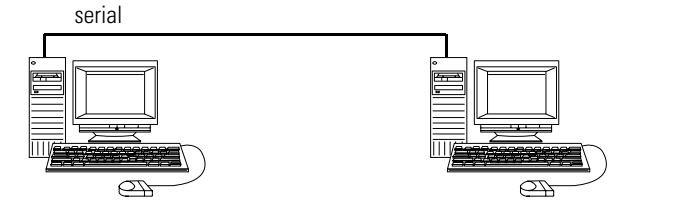
The SoftLogix controller has an RS-232 LED that follows this behavior:

This state:	Means:
off	You selected "None" for the COM port selection of the controller.
green	The COM port you selected was successfully assigned to channel 0 of the controller.
red	There is conflict with COM port or the COM port number you selected is invalid.

Please note that these LED states are different than for the ControlLogix controller.

Example 1: Workstation Directly Connected to a SoftLogix Controller

In the following example, a workstation directly connects to a SoftLogix controller over a serial link.



Use RSLogix 5000 programming software to configure the controller's serial port for the DF1 point-to-point (full-duplex) protocol. This type of protocol supports simultaneous transmission between two devices in both directions. The DF1 point-to-point protocol controls message flow, detects and signals errors, and retries if errors are detected.

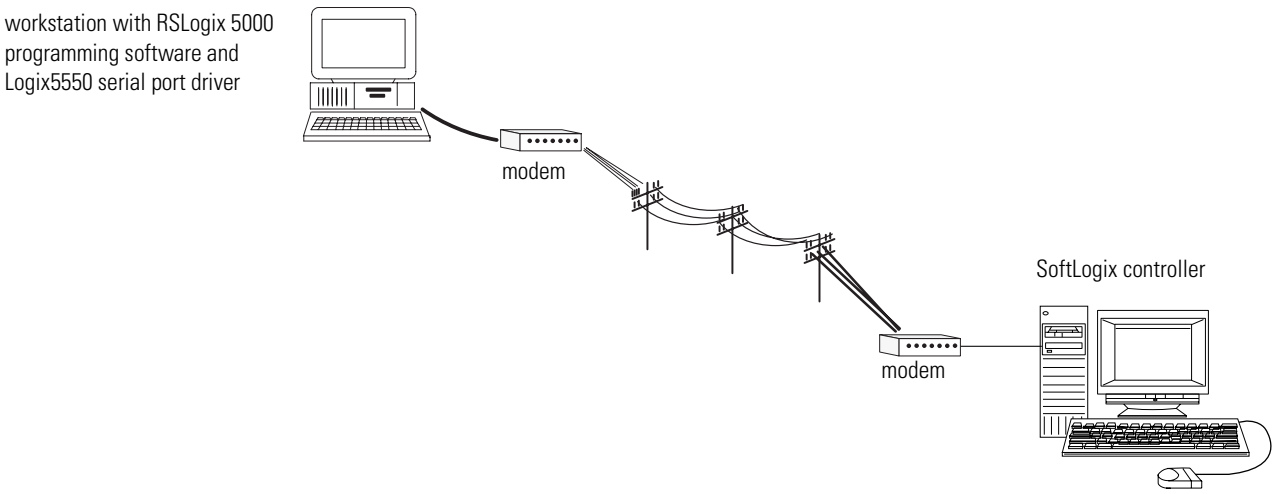
IMPORTANT The workstation with RSLogix 5000 programming software must also have the Logix5550 serial port driver installed through RSLinx software.

Configuring a DF1 point-to-point station

This field:	Description:
Station address	The station address for the serial port on the DF1 point-to-point network. Enter a valid DF1 address (0-254). Address 255 is reserved for broadcast messages. The default is 0.
NAK receive limit	Specifies the number of NAKs the controller can receive in response to a message transmission. Enter a value 0-127. The default is 3.
ENQ transmit limit	Specifies the number of inquiries (ENQs) you want the controller to send after an ACK timeout. Enter a value 0-127. The default is 3.
ACK timeout	Specifies the amount of time you want the controller to wait for an acknowledgment to its message transmission. Enter a value 0-32767. Limits are defined in 20ms intervals. The default is 50 (1000ms).
Embedded response	Specifies how to enable embedded responses. Select Autodetect (enabled only after receiving one embedded response) or Enabled. The default is Autodetect.
Error detection	Select BCC or CRC error detection. Configure both stations to use the same type of error checking. **BCC:** the controller sends and accepts messages that end with a BCC byte for error checking. BCC is quicker and easier to implement in a computer driver. This is the default. **CRC:** the controller sends and accepts messages with a 2-byte CRC for error checking. CRC is a more complete method.
Enable duplicate detection	Select whether or not the controller should detect duplicate messages. The default is duplicate detection enabled.

Example 2: Workstation Remotely Connected to a SoftLogix Controller

In the following example, a workstation remotely connects to a SoftLogix controller over a serial link. A modem is connected to the controller to provide remote access.



If you use a modem to remotely connect the controller to one workstation, use RSLogix 5000 programming software to configure the serial port of the controller for the DF1 point-to-point (full-duplex) protocol, as in the previous example. If the controller is part of a master/slave serial network, configure the serial port of the controller for either the DF1 master or DF1 slave protocol (both half-duplex).

Master/slave communication methods

A master station can communicate with a slave station in two ways:

Name:	This method:	Benefits:
standard communication mode	Initiates polling packets to slave stations according to their position in the polling array(s). Polling packets are formed based on the contents of the normal poll array and the priority poll array.	This communication method is most often used for point-to-multipoint configurations. This method provides these capabilities: • slave stations can send messages to the master station (polled report-by-exception) • slave stations can send messages to each other via the master • master maintains an active station array The poll array resides in a user-designated data file. You can configure the master: • to send messages during its turn in the poll array *or* • for between-station polls (master transmits any message that it needs to send before polling the next slave station) In either case, configure the master to receive multiple messages or a single message per scan from each slave station.
message-based communication mode	initiates communication to slave stations using only user-programmed message (MSG) instructions. Each request for data from a slave station must be programmed via a MSG instruction. The master polls the slave station for a reply to the message after waiting a user-configured period of time. The waiting period gives the slave station time to formulate a reply and prepare the reply for transmission. After all of the messages in the master's message-out queue are transmitted, the slave-to-slave queue is checked for messages to send.	If your application uses satellite transmission or public switched-telephone-network transmission, consider choosing message-based communication. Communication to a slave station can be initiated on an as-needed basis. Also choose this method if you need to communicate with non-intelligent remote terminal units (RTUs).

Configuring a DF1 slave station

This field:	Description:
Station address	The station address for the serial port on the DF1 slave. Enter a valid DF1 address (0-254). Address 255 is reserved for broadcast messages. The default is 0.
Transmit retries	The number of times the remote station retries a message after the first attempt before the station declares the message undeliverable. Enter a value 0-127. The default is 3.
Slave poll timeout	Specifies the amount of time the slave station waits to be polled by a master before indicating a fault. Enter a value 0-32767. Limits are defined in 20ms intervals. The default is 3000 (60,000ms).
EOT suppression	Select whether or not to suppress sending EOT packets in response to a poll. The default is not to suppress sending EOT packets.
Error detection	Select BCC or CRC error detection. Configure both stations to use the same type of error checking. **BCC:** the controller sends and accepts messages that end with a BCC byte for error checking. BCC is quicker and easier to implement in a computer driver. This is the default. **CRC:** the controller sends and accepts messages with a 2-byte CRC for error checking. CRC is a more complete method.
Enable duplicate detection	Select whether or not the controller should detect duplicate messages. The default is duplicate detection enabled.

Configuring a DF1 master station

This field:	Description:
Station address	The station address for the serial port on the DF1 master. Enter a valid DF1 address (0-254). Address 255 is reserved for broadcast messages. The default is 0.
Transmit retries	Specifies the number of times a message is retried after the first attempt before being declared undeliverable. Enter a value 0-127. The default is 3.
ACK timeout	Specifies the amount of time you want the controller to wait for an acknowledgment to its message transmission. Enter a value 0-32767. Limits are defined in 20ms intervals. The default is 50 (1000ms).
Reply message wait	**Message-based polling mode only** Specifies the amount of time the master station waits after receiving an ACK to a master-initiated message before polling the slave station for a reply. Enter a value 0-65535. Limits are defined in 20ms intervals. The default is 5 (100ms).
Polling mode	Select one of these: • Message Based (slave cannot initiate messages) • Message Based (slave can initiate messages) - default • Standard (multiple message transfer per node scan) • Standard (single message transfer per node scan)
Master transmit	**Standard polling modes only** Select when the master station sends messages: • between station polls (default) • in polling sequence

This field:	Description:
Normal poll node tag	**Standard polling modes only** An integer tag array that contains the station addresses of the slave stations. Create a single-dimension array of data type INT that is large enough to hold all the normal station addresses. The minimum size is three elements. This tag must be controller-scoped. The format is: *list[0]* contains total number of stations to poll *list[1]* contains address of station currently being polled *list[2]* contains address of first slave station to poll *list[3]* contains address of second slave station to poll *list[n]* contains address of last slave station to poll
Normal poll group size	**Standard polling modes only** The number of stations the master station polls after polling all the stations in the priority poll array. Enter 0 (default) to poll the entire array.
Priority poll node tag	**Standard polling modes only** An integer tag array that contains the station addresses of the slave stations you need to poll more frequently. Create a single-dimension array of data type INT that is large enough to hold all the priority station addresses. The minimum size is three elements. This tag must be controller-scoped. The format is: *list[0]* contains total number of stations to be polled *list[1]* contains address of station currently being polled *list[2]* contains address of first slave station to poll *list[3]* contains address of second slave station to poll *list[n]* contains address of last slave station to poll
Active station tag	**Standard polling modes only** An array that stores a flag for each of the active stations on the DF1 link. Both the normal poll array and the priority poll array can have active and inactive stations. A station becomes inactive when it does not respond to the master's poll. Create a single-dimension array of data type SINT that has 32 elements (256 bits). This tag must be controller-scoped.
Error detection	Select BCC or CRC error detection. Configure both stations to use the same type of error checking. **BCC:** the controller sends and accepts messages that end with a BCC byte for error checking. BCC is quicker and easier to implement in a computer driver. This is the default. **CRC:** the controller sends and accepts messages with a 2-byte CRC for error checking. CRC is a more complete method.
Enable duplicate detection	Select whether or not the controller should detect duplicate messages. The default is duplicate detection enabled.

If you choose one of the standard polling modes

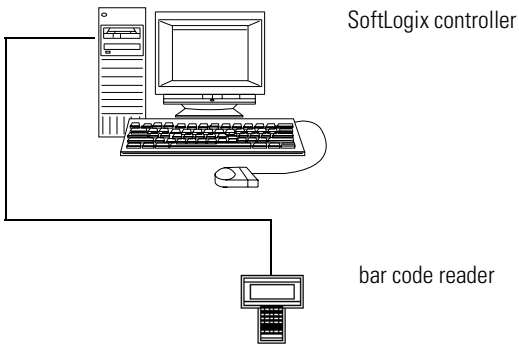
The master station polls the slave stations in this order:

1. all stations that are active in the priority poll array
2. one station that is inactive in the priority poll array
3. the specified number (normal poll group size) of active stations in the normal poll array
4. one inactive station, after all the active stations in the normal poll array have been polled

Use the programming software to change the display style of the active station array to binary so you can view which stations are active.

Example 3: SoftLogix Controller to a Bar Code Reader

In the following example, the SoftLogix controller connects to a bar code reader. A bar code reader is an ASCII device, so you configure the serial port differently than in the previous examples. Configure the serial port for user mode, rather than a DF1 mode.

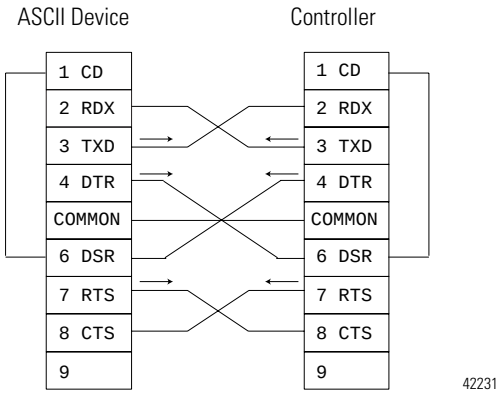
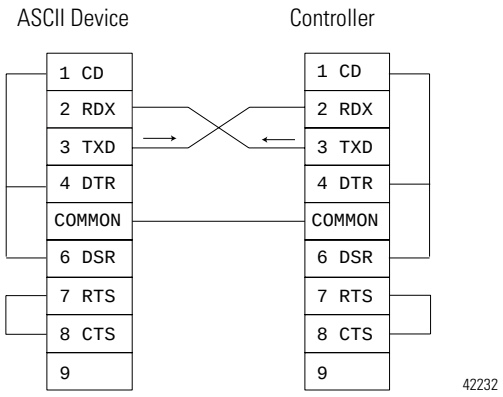


Connect the ASCII device to the controller

To connect the ASCII device to the serial port of the controller:

1. For the serial port of the ASCII device, determine which pins send signals and which pins receive signals.

2. Connect the sending pins to the corresponding receiving pins and attach jumpers:

If the communications:	Then wire the connectors as follows:
handshake	
do not handshake	

3. Attach the cable shield to both connectors and tie the cable to both connectors.

4. Connect the cable to the controller and the ASCII device.

The following table lists the default serial port configuration settings for the ASCII protocol. You specify these settings on the User Protocol tab under Controller Properties.

Configuring user mode

This field:	Description:
Buffer size	Specify the maximum size (in bytes) of the data array you plan to send and receive. The default is 82 bytes.
Termination characters	Specify the characters you will use to designate the end of a line. The default characters are '$r' and '$FF'.
Append characters	Specify the characters you will append to the end of a line. The default characters are '$r' and '$l'.
XON/XOFF	Select whether or not to regulate the flow of incoming data. The default is disabled.
Echo mode	Select whether or not to echo data back to the device from which it was sent. The default is disabled.
Delete mode	Select Ignore, CTR, or Printer for the delete mode. The default is Ignore.

Programming ASCII instructions

The controller supports ASCII instructions to communicate with ASCII devices. Your RSLogix5000 programming software CDROM includes programming examples using ASCII instructions.

For information about using these examples, see the *Logix5000 Controllers Common Procedures Programming Manua*l, publication 1756-PM001.

Notes:

Configuring and Using Simulated I/O

Using This Chapter

The 1789-SIM module is a software-only module that comes with the SoftLogix controller, absolutely no hardware required. You can put as many SIM modules as you have available slots according to your activation level.

The 1789-SIM module lets you change inputs and monitor outputs of your application by toggling input bits and monitoring output bits on the 1789-SIM module. Use this module to test logic without having physical I/O attached to the system.

Configuring Your System for a 1789-SIM Module

For the SoftLogix controller to simulate local I/O, you need:

- a 1789-SIM module (comes with the SoftLogix5800 controller)

  You are limited by the activation level of your SoftLogix controller as to how many modules you can install.

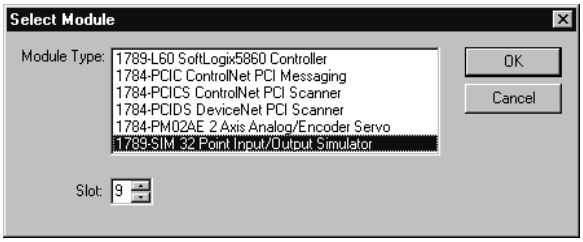
- RSLogix5000 programming software to configure the 1789-SIM module

Even though the 1789-SIM module is a software-based module, each module you create uses communication resources. If you are controlling actual I/O and simulating I/O, the 1789-SIM module(s) in your application use communication resources that could impact control performance. If this occurs, increase the RPI of your 1789-SIM module(s). This maintains control performance because the greater RPI of the 1789-SIM module(s) lessens the load on the SoftLogix system.

Step 1: Create the 1789-SIM module in the chassis

Before you can operate the module, you must create the 1789-SIM module as part of the SoftLogix chassis. You can install as many 1789-SIM modules as allowed by your activation level of the controller.

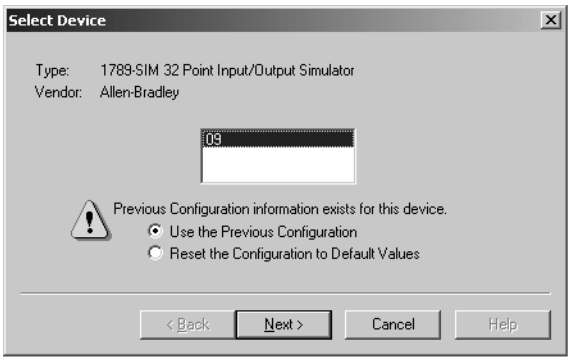
1. From the SoftLogix chassis monitor, select Slot → Create Module or right click the appropriate slot and select Create. Select the motion module.



Specify the backplane slot number.

Click OK

2. Verify the slot number for the 1789-SIM module.

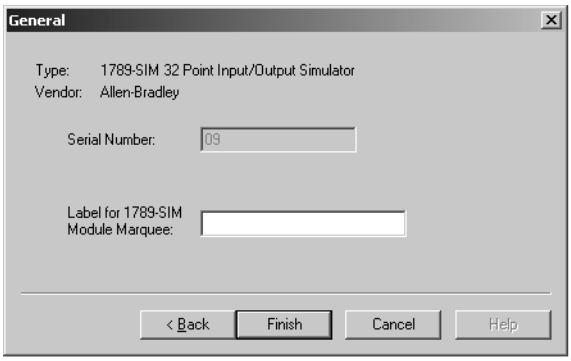


Verify the slot number.

If you previously had a 1789-SIM module configured in this slot, the chassis monitor remembers the configuration of that previous module.

Click Next

3. Specify a label for the 1789-SIM module.

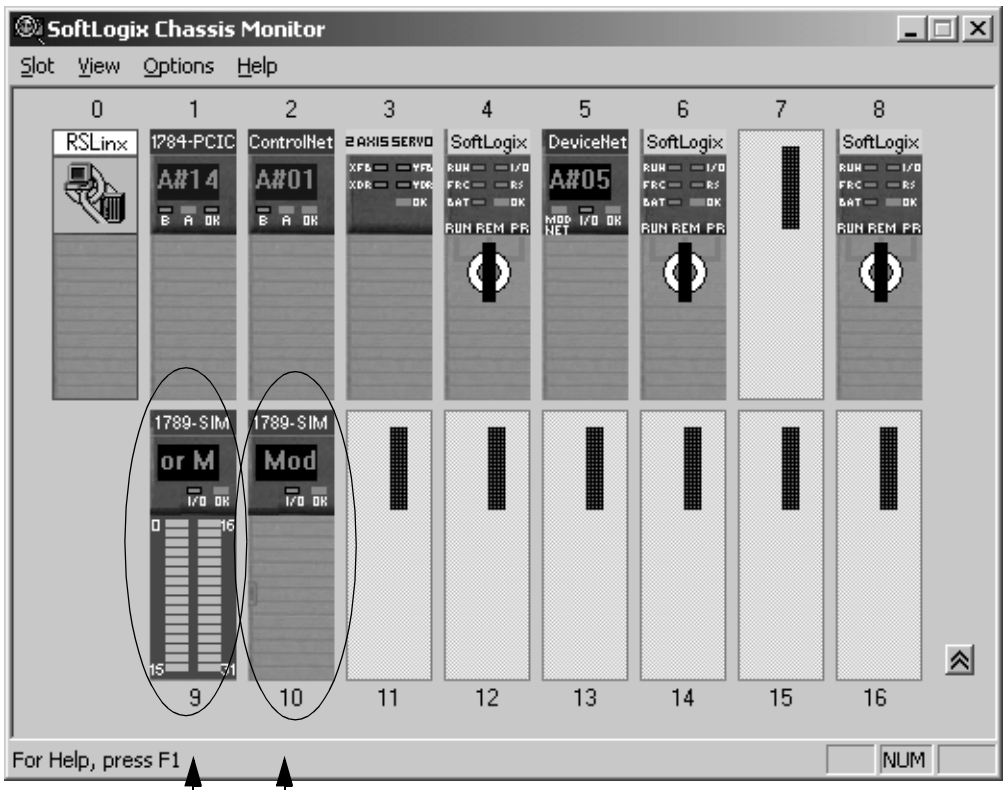


Enter the label. The text you enter here scrolls across the front of the module in the chassis monitor. If you do not enter a label, the default label is "Simulator Module."

Click Finish

You can specify any slot number greater than 0 for the 1789-SIM module. RSLinx software resides in slot 0.

The chassis monitor shows the 1789-SIM module as a virtual module in the SoftLogix chassis. Note that the door of the 1789-SIM module opens to display the output bits. Left-click to open or close the module door.



The module door is open.

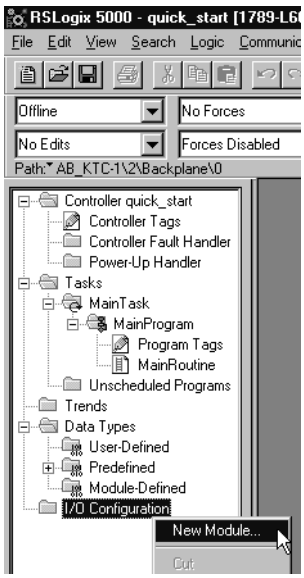
The module door is closed.

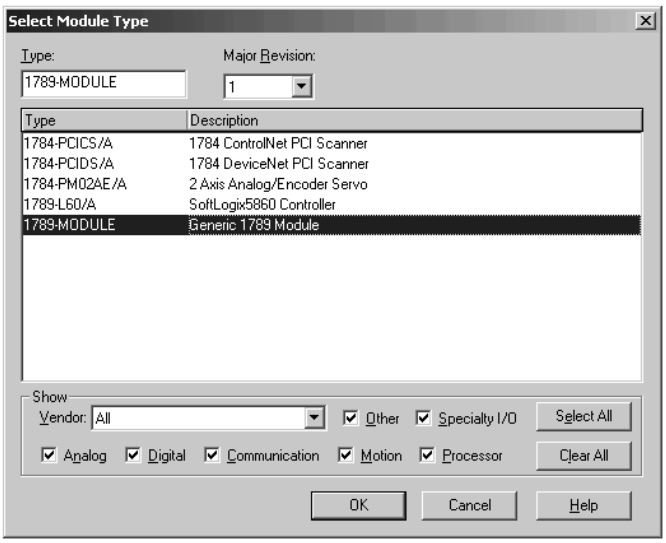
Step 2: Configure the 1789-SIM module as part of the project

Use RSLogix 5000 programming software to map the 1789-SIM module as part of the SoftLogix project.

1. In RSLogix 5000 programming software, select the I/O Configuration folder.

2 Right-click to select New Module and add a generic 1789 module.



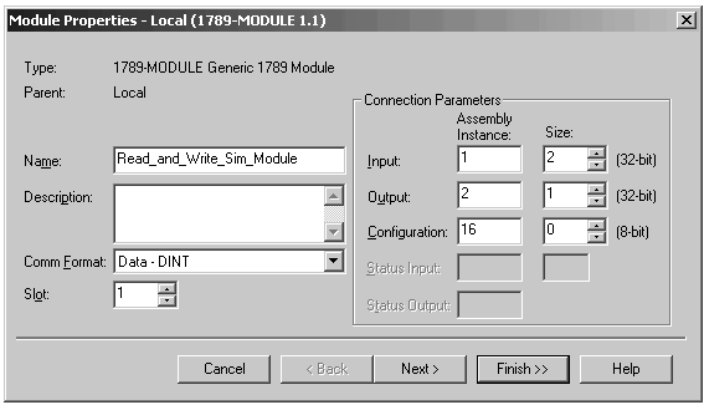


Click OK

3. Specify the connection parameters.

For this connection:	Set the connection parameters to:	
read and write This connection lets the controller read inputs and write outputs	Input Assembly Instance	1
	Input Size	2
	Output Assembly Instance	2
	Output Size	1
	Configuration Assembly Instance	16
	Configuration Size	0
listen only This connection lets the controller read inputs, but not write outputs	Input Assembly Instance	1
	Input Size	2
	Output Assembly Instance[1]	3
	Output Size	1
	Configuration Assembly Instance	16
	Configuration Size	0

(1) This is the only parameter setting that is different from the read and write connection.

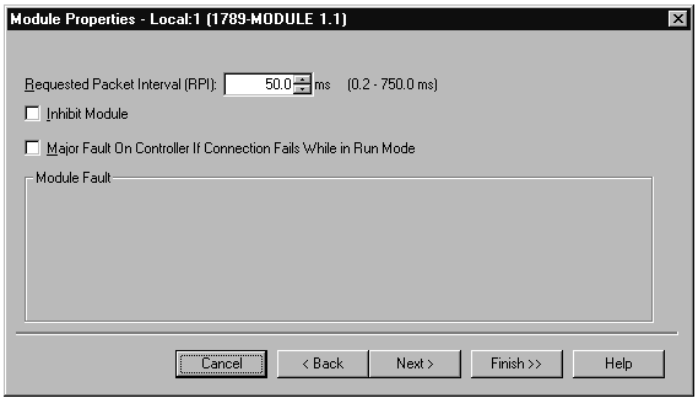


Click OK

continued

4. Specify the requested packet interval (RPI).

You must enter at least 50.0ms for the RPI. The connection will fail if the RPI is less than 50.0 ms. The default RPI is 5.0ms



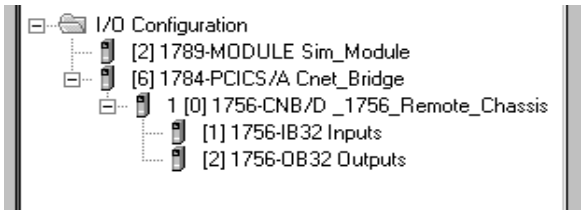
Click Finish

IMPORTANT You must specify an RPI of at least 50.0 ms for each 1789-SIM module or the connection to the module will fail. Because this module uses the generic module profile, the default RPI is 5.0 ms.

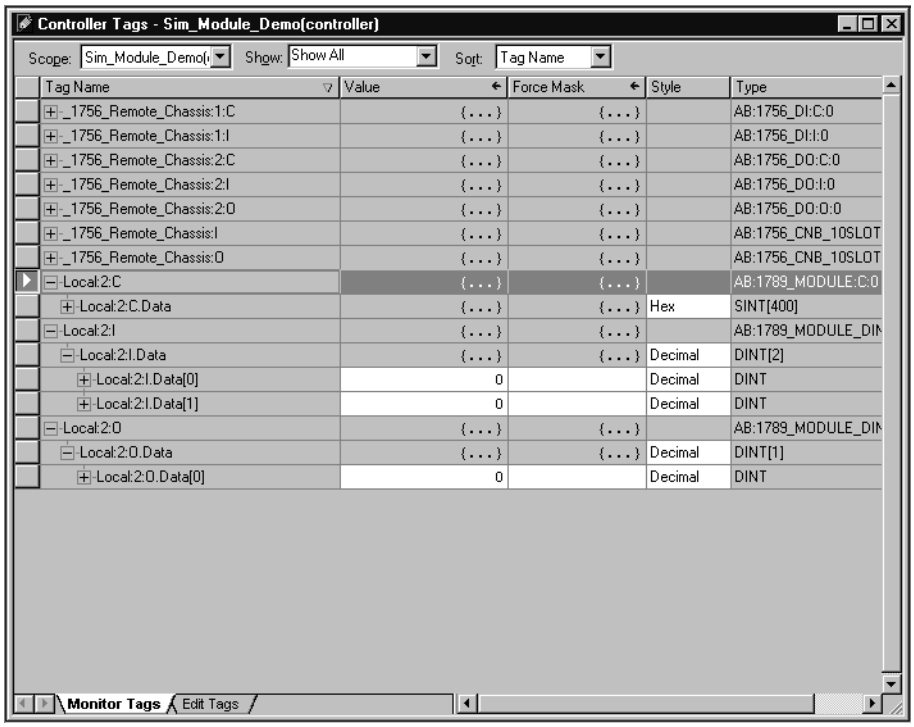
Mapping I/O Data to the 1789-SIM Module

When you add a 1789-SIM module to an RSlogix 5000 project, the programming software automatically assigns input and output data structures for the module. For example, this I/O configuration generates these I/O data structures:

The 1789-SIM module is in slot 2.



The programming software assigns these controller-scoped tags to the 1789-SIM module in slot 2.

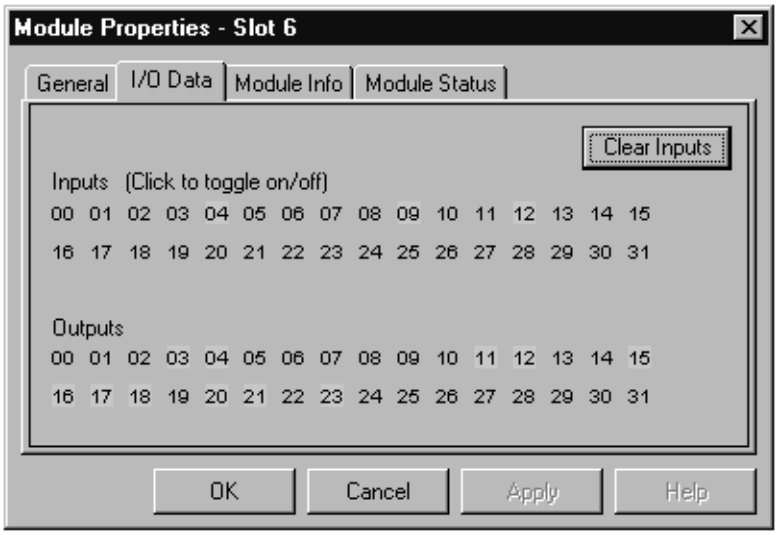


Toggling Inputs and Monitoring Outputs

Once the 1789-SIM module is installed in the chassis monitor, you can monitor the module.

1. In the chassis monitor, right-click the 1789-SIM module and select Properties.

2 Select the I/O Data tab.



Select the I/O Data tab of the module properties. Left mouse click a specific input bit to toggle it on or off.

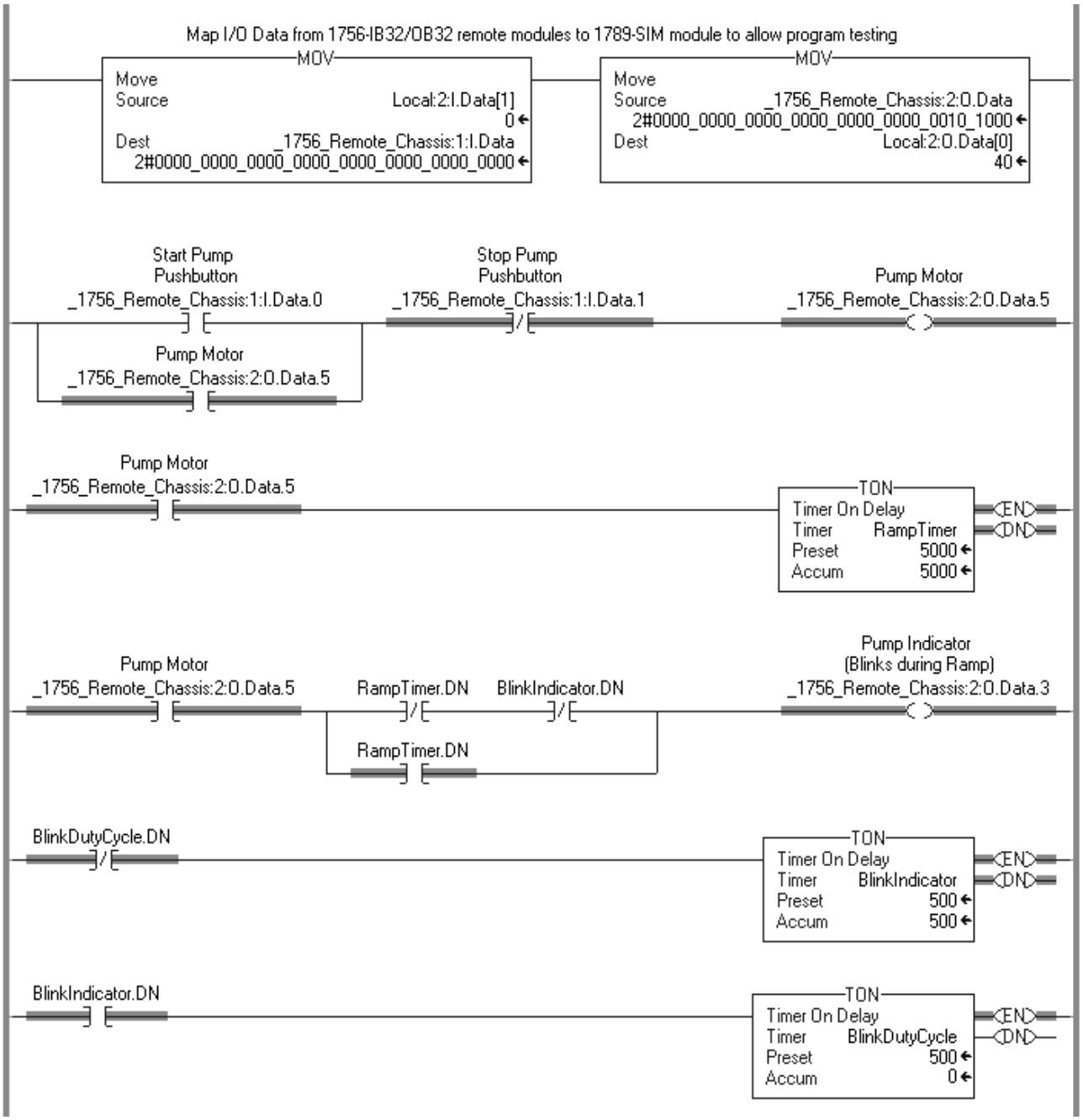
This tab also shows the state of the output bits. This is the same state that is displayed when you open the module door from the chassis monitor.

Outputs remain in last state when the controller is in program mode (this is not user configurable). If the I/O connection is broken to the module, all of the outputs will reset to OFF.

Example: Moving Application Data into the 1789-SIM Tags

The following example uses MOV instructions to copy:

- the input data from the 1789-SIM module into the application's input tags
- the application's output tags into the output data for the 1789-SIM module



Windows NT/2000 Considerations

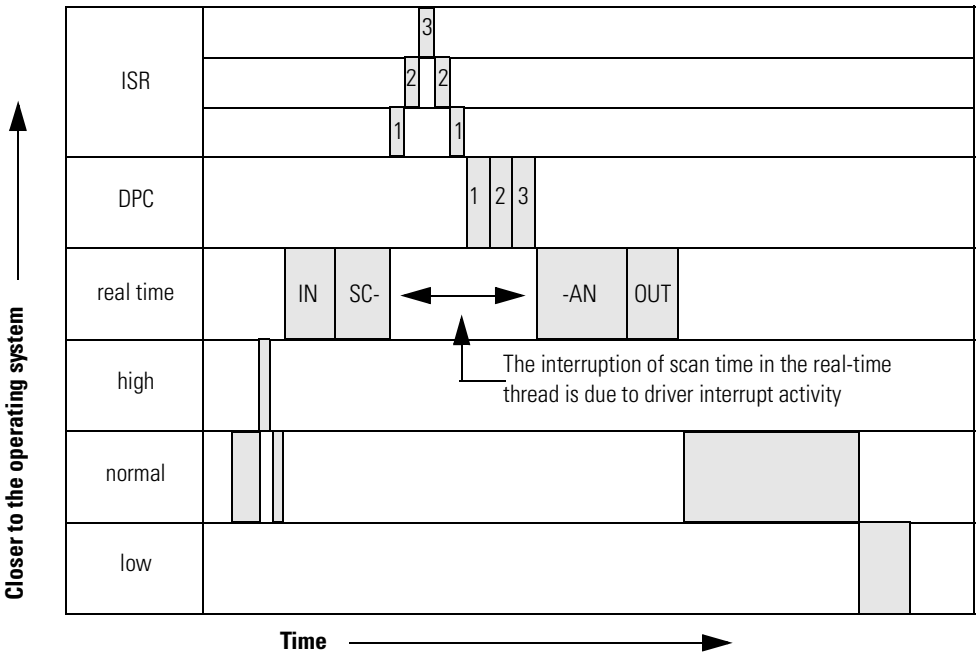
Using This Appendix

Windows NT Objects

There are three objects that execute within Windows that get CPU resources based on Window's multitasking/multithreading algorithms.

Windows Object	Description:
interrupt service routine (ISR)	An interrupt service routine is a software routine that is primarily executed in response to hardware and software interrupts. ISRs always execute immediately and run in the kernel-mode layer of Windows. Each ISR executes at one of 32 levels. The hardware interrupts that occur in a computer get mapped to 16 of the 32 possible levels. The important point about ISRs is that they are written by the vendors of hardware and software products and not Microsoft. Microsoft can recommend how to write ISRs, but there is no guarantee about the how well other vendors write them. For example, once an ISR starts to execute, it can raise its level to a higher priority so that the Windows scheduler won't swap it out, or it can even make a function call to mask all other interrupts until it is finished. Due to the variations in code writing, ISRs can cause swings in system responsiveness and determinism in a soft controller.
deferred procedure call (DPC)	A deferred procedure call is a software routine that is queued by an ISR to perform less time-critical processing. DPCs also execute in the kernel-mode layer of Windows, and therefore can prevent user mode applications from running. DPCs do not have priority levels, but execute in a FIFO (first-in-first-out) order as queued by their associated ISRs. Poorly written DPCs could unnecessarily keep the SoftLogix controller from running. Disk drivers, network drivers, and video drivers can be major sources of long DPCs, causing variation in SoftLogix performance. Do not use CD writing (recording) devices and fancy screen savers on the SoftLogix controller because they have shown significant jumps in CPU utilization.
dispatched threads of execution	Threads are the primary execution pieces of software that Windows switches between. Some are associated with larger application programs and background processes, while others are created by the kernel and device drivers. Threads are individual code segments spawned from processes that can be in one of four priority levels; low, medium, high and real-time. Threads that are spawned from a real-time process, like the SoftLogix controller, execute to the point of blocking, yielding, or completing. The dwell component of the SoftLogix controller allows the continuous task of the controller to give time to other lower priority threads that need to execute.

The following diagram shows the relationship between these objects and shows how one object has to stop running if another with higher priority wants to execute. The SoftLogix controller executes as a real-time priority process, and thus waits for all ISRs and DPCs to complete before executing.



Other Considerations

Consideration:	Description:
Multiple CPUs in one computer	Multiple CPUs in the PC can greatly improve performance. The Windows scheduler algorithm automatically uses both CPUs to execute whatever needs to be executed. Any code that needs to execute will move to whichever CPU is available, unless the current process specifically requests a certain CPU. The CPU Affinity parameter of the SoftLogix controller has you specify a which CPU to execute on, allowing you to customize your system for more determinism.
Blue screen events	Blue screen events are the result of the Pentium processor generating self-diagnostic events that fall in the category of fault or abort. Usually there is code to recover from fault interrupts, such as Page Faults, and these do not cause Windows to stop. But there are occasions in the case of hardware failures that generate a NMI (non-maskable interrupt) or a parity error that are considered faults that cannot be ignored and therefore Windows does the proper thing and immediately stops. This is similar to a PLC-5 processor 'red lighting' when it detects an internal memory or hardware error. Blue screens that occur during system integration of new third-party hardware indicate a poorly written driver that is corrupting Windows kernel memory. The I/O controlled by the SoftLogix controller will always go to a fail safe mode of operation after a blue screen event.

Consideration:	Description:
Microsoft service pack	Microsoft service pack is the name Microsoft gives to an operating system upgrade. It is always recommended to apply the latest service pack after installing third party software, especially networking drivers and the addition of network protocols. Whenever you receive errors that seem low-level or don't make sense, reapply the latest Microsoft service pack and reboot.
Third party peripheral devices	Third party peripheral devices, such as network cards and IDE devices should be installed directly in the computer's master PCI bus. IDE devices should use PCI bus mastering. Bus mastering is the capability of writing directly to the computer's memory without having to use the Pentium chip to move the data. You can use Microsoft's DMACHECK.EXE utility to see whether a third-party card truly uses PCI bus mastering. You might also need to turn bus mastering on within the BIOS setup of your computer. See the documentation for your computer for more information about bus mastering.
TestTime utility	The SoftLogix controller ships with a TestTime utility that you can use to determine the responsiveness of your system to CPU interrupts. The utility measures how long it takes to respond to a software interrupt that is generated every 2 ms. It measures the average, max, number of occurrences and standard deviation of how quickly your PC responds. If you find significant delays, focus on any peripheral devices on the computer and their associated drivers. The best way to use the utility is to run it on a new system with no software or third-party hardware installed to get a baseline measurement. Then rerun the TestTime utility each time you install a piece of hardware or a software package to determine if there is a problem.
Screen savers	Disable screen savers on your computer. OpenGL screen savers have been known to generate excessive loading on a PC. This can cause fluctuations in SoftLogix control because video-driver operations interrupt the SoftLogix real-time execution priority under Windows. OpenGL is a standard for generating 2-dimensional and 3-dimensional graphics in current screen savers and animated games.

Running a SoftLogix Controller on Windows

The SoftLogix controller executes as a service (background program) that starts when Windows boots and then runs at real-time priority within Windows. Most other applications, such as word processors and spreadsheets, run at normal priority. Because SoftLogix runs at a real-time level, it is guaranteed to get as many CPU cycles as it needs before allowing the CPU to execute other application programs. Only DPCs and ISRs run before a SoftLogix controller.

Selecting a dwell time setting

Every SoftLogix controller has a main task that can be configured to run continuously or periodically. If set for continuous, the main task would use all of the Windows CPU cycles, if it were able, running as a real-time priority process. But the dwell time configuration of the SoftLogix controller is a value in milliseconds which is directly added to the end of every scan of a continuous program task. The dwell time is a period of time that counts off in real-time after the SoftLogix controller's continuous task. This time is like a sleep time for the SoftLogix controller so that Windows can execute lower priority threads.

If a SoftLogix controller's periodic task is set to run, it runs during the dwell time, but the time spent executing the periodic task is not added to the dwell time. The dwell time counts in the background in real-time and the end of the dwell marks the continuous task as ready to run. The continuous task will run as long as no other periodic tasks are already executing or are ready to execute.

A dwell time of 0 ms does leave some dynamic amount of time of dwell that is less than 1 ms to prevent you from completely using all CPU cycles, and thus locking up your computer. When the continuous task enters the dwell time, it makes a function call to Windows to "SwitchToThread," which is a function that lets the next thread that needs to run go ahead and execute.

If multiple SoftLogix controllers in the same virtual chassis are set for a dwell time of 0 ms, the controllers will starve other applications that are running at normal priority. The effect is sluggish mouse control and slow response time by other Windows applications. And if you run this configuration on a slower computer, you may even lock yourself out of being able to do anything in Windows.

IMPORTANT

It is possible to lock yourself out of your computer if you have multiple controllers installed in the virtual chassis and

- each controller is set for a dwell of 0
- periodic tasks are set for very low settings (short time periods)

In this state, the keyboard and mouse are not recognized by Windows because Windows is spending all of its time executing the real-time tasks of the SoftLogix controllers. If the controllers are set to start in "last state", you will never be able to move the mouse to put them in Program mode to free up CPU resources.

It is recommended that during development, set the controller to start in the Remote Program mode. This way, if you ever have controllers in Run mode and the PC locks up, you can cycle power and have the controllers come up in Program mode, giving you enough CPU time to make changes to your application to correct the problem. Then after development is complete, you can change the startup mode to start in "last state".

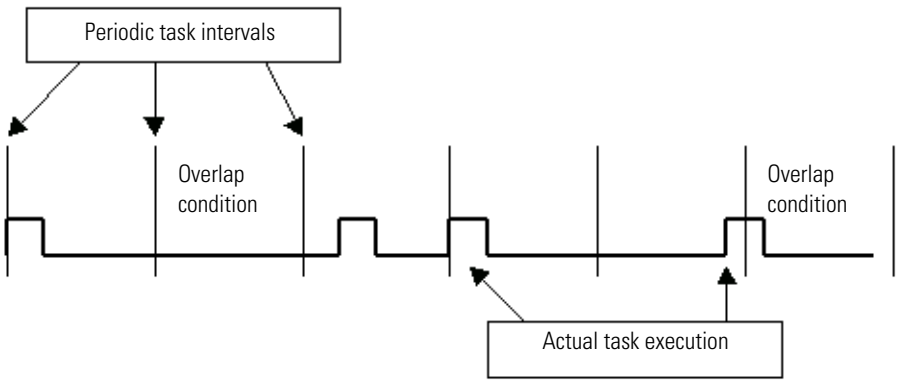
There is no window in RSLogix 5000 programming software that shows overall scan time including the dwell time component. The scan times reported in each task are values that indicate the time to scan a particular program and do not include dwell time. Use the Task Manager's Performance Monitor to gauge the effect of dwell time settings.

Using periodic tasks

Periodic tasks always attempt to execute according to their setting, and they always interrupt the continuous task. If the controller is running it's dwell time, a periodic task still interrupts the dwell time to run. If two periodic tasks attempt to run at the same time, the task that has the higher priority executes first. Be careful not to execute too many periodic tasks with short intervals as you can start to use all the bandwidth of the computer without leaving CPU cycles to operate the mouse and keyboard.

A periodic task pauses if an ISR or DPC routine needs to be executed by Windows, and then the periodic task continues when the interrupt is complete. The periodic task executes again in real time at the next preset interval. The time spent in the ISR or DPC does not get added to the time counted between periodic tasks.

A periodic task detects an overlap and sets the Overlap fault bit in the controller if a periodic tasks fails to run at all during its assigned time slot or if a periodic tasks starts later than scheduled and cannot complete before the start of the next period. The following diagram shows periodic task intervals, when a task actually starts, and what is considered an overlap condition.

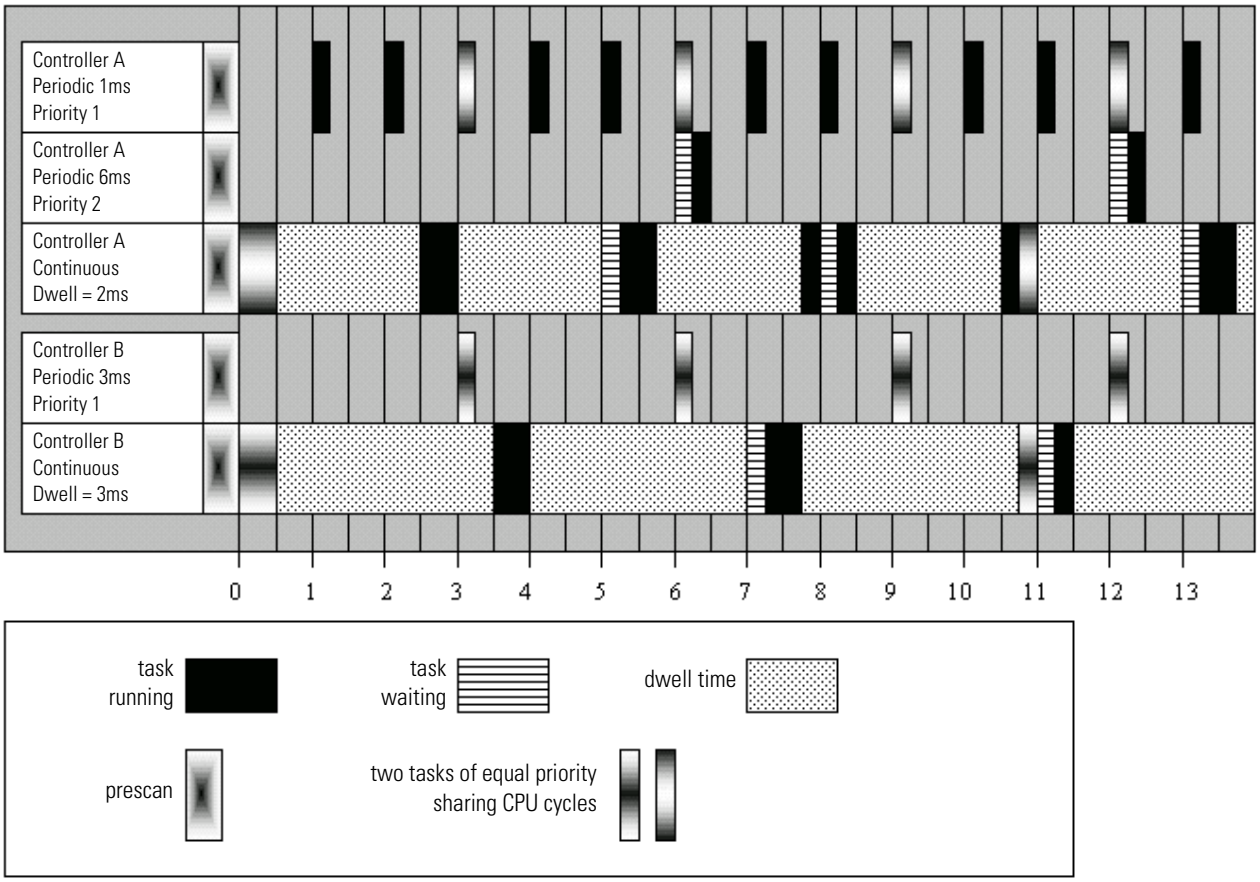


If two controllers in the same virtual chassis each have high priority periodic tasks and the tasks become active at the same time, Windows tries to switch between the tasks at whatever quantam is set within Windows. The quantam varies based on the performance boost setting for the process. With no performance boost, the quantam is 20 ms for the Windows workstation. Typically a SoftLogix controller finishes the entire scan of a periodic task before using a whole quantam.

To use the whole quantam, a thread has to be ready to execute the whole time. If a thread stops and makes any type of I/O call, (such as disk drive, DRAM memory, etc.), the thread gets switched by Windows and the CPU executes the next thread that is ready to run. This applies to the SoftLogix controller because the controller references different tags in a program scan, which are DRAM I/O operations. Therefore, Windows switches back and forth many times between two periodic tasks that are executing at the same time and at the same priority level, with the switching happening in the microsecond range.

The following diagram shows the timing of task execution between two SoftLogix controllers in the same virtual chassis. Each controller has periodic tasks and a continuous task. The example periodic tasks are short and only take 0.25 ms to execute. The example continuous tasks take 0.5 ms to execute. Anytime two periodic tasks need to execute at the same time and they each have the same priority, they share CPU cycles as Windows constantly switches between.

The beginning of the diagram shows what happens when the controller goes from Program mode to Run mode, which involves a prescan of all tasks. Then Run mode begins. The real time starts counting as shown at the bottom of the diagram.



Selecting the system overhead time slice

All Logix-based controllers have a configuration setting for the system overhead time slice. This function lets the controller take care of communication requests that occur from other controllers or from queued requests from within the controller's application program. The time slice switches the priority level of the continuous task with that of the background communication task, which is always running at a lower level than the continuous task.

The time slice setting is a percent value that is applied to a 100 ms background timing window. With a setting of 10% (the default), for every 100 ms of real time, there is 10 ms of time when the communications task priority is higher than the continuous task. If there is communication activity to perform, the controller does it and when completed, the controller lets the continuous task run again during that 10 ms window. For the next 90 ms, the continuous task is at its normal priority and the communications task is lower. During dwell time, if there are communication tasks ready to run, they will run during the dwell even though the communication task is not switched to a higher priority. And any periodic task that needs to run overrides both the continuous task and the communications task.

Integrating motion

The component of motion that most directly effects performance in the SoftLogix controller is the course update setting. This setting is the periodic rate for a separate SoftLogix thread in Windows that takes all motion commands that execute in the SoftLogix controller and sends them to the motion card.

In a ControlLogix system, to improve motion performance, you would typically use a dedicated ControlLogix CPU for each motion module. In a SoftLogix system, adding more controllers actually decreases system performance. It is better to put everything in one controller. Or add another Pentium processor in the computer and dedicate motion to just one of the Pentium processors.

Using multiple SoftLogix controllers

Multiple controllers in the virtual chassis, executing on a computer with only one CPU, is less efficient than one controller. With multiple controllers, Windows has to take time to swap threads in a round-robin fashion, assuming all the controllers have a continuous task and a very small dwell. If your computer has multiple CPUs, then assign multiple controllers across the multiple CPUs.

PC Hardware Considerations

The PC hardware you chose for the SoftLogix controller will have a dramatic impact on the performance of the SoftLogix control system. The recommended platform at the time of release of the SoftLogix controller is a Pentium III (450 Mhz minimum) processor with 128 Mbytes of RAM. As mentioned earlier, a dual CPU system will provide greater stability in ladder scan time and motion updates. And if a graphical MMI package is running on the same computer, 256 Mbytes of RAM is recommended.

Other considerations include:

Consideration:	Description:
hard drives	The hard drive should be capable of bus mastering in order to reduce loading on the Pentium processor. Bus mastering allows the hard drive to initiate data transfers without using Pentium CPU cycles. To accomplish this the PC must have a motherboard that supports this technology as well as a BIOS that supports it. Then the drive itself should be capable of bus mastering. Most PC vendors will fully integrate for you this IDE bus mastering capability.
CD-ROM drives	Verify that the hard drive for your PC is on a designated IDE channel and that the CD-ROM drive is on another (secondary). Some PC vendors attempt to put the CD -ROM as slave off of the primary IDE channel and this causes performance problems for the hard drive.
redundant array of disks (RAID)	This technology uses multiple hard drives in a PC, so that any one hard drive can fail without causing Windows to crash. There are 5 different versions of RAID, each with its own method of error correction and recovery. The SoftLogix controller supports the RAID environment, which is recommended for critical applications that can't afford a crash. Sensitivity to hard drive crashes is common among PC users, but over the last 5 years, the reliability of hard drives has greatly increased. RAID technology is expensive and can be hard to implement and support. A more inexpensive option is to have another hard drive with a copy of the original hard drive image available. You can even mount the duplicate in the same PC without power or IDE connections, so that it is ready to connect if the original hard drive ever fails.
uninterruptable power supplies (UPS)	Uninterruptable power is an excellent accessory for a SoftLogix system as it prevents disruptions to the SoftLogix controller due to brown outs and power outages, which are the most common interferences to PCs. There are many UPS systems available, including some with digital outputs that can be interfaced back into the SoftLogix controller using discrete inputs so that the controller can detect a power outage and prepare for an orderly shutdown after a designated amount of time.

Notes:

Monitoring Controller LEDs

Indicator:	Color:	Description:
RUN	off	The controller is in Program or Test mode.
	green	The controller is in Run mode.
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).
	green	The controller is communicating with all the devices in its I/O configuration.
	green flashing	One or more devices in the I/O configuration of the controller are *not* responding.
	red flashing	A virtual chassis error was detected. Contact your Rockwell Automation representative or local distributor.
FRC	off	No tags contain force values. Forces are inactive (disabled).
	flashing	At least one tag contains a force value. Force values are inactive (disabled).
	green	Forces are active (enabled). Force values may or may not exist.
RS232[1]	off	No COM port was selected.
	green	The selected COM port was successfully assigned to channel 0 of the controller.
	red	There is a COM port conflict or you selected an invalid COM port number.
BAT[1]	off	Normal operation.
	amber flashing	The controller is in power-up mode.
	red	Persistent storage for the controller has failed.
OK	red flashing	Recoverable fault
	red	Non-recoverable fault. To correct: 1. Remove the controller from the virtual chassis and then re-install the controller. 2. Download the project. 3. Place the controller in Run mode. If the problem continues to occur, contact your Rockwell Automation representative or local distributor.
	green	The controller is OK.

(1) Note that these LEDs function slightly different than the same LEDs on a ControlLogix controller.

Notes:

Numerics

A

C

D

E

G

H

I

L

M

O

P

Allen-Bradley
Publication Problem Report

If you find a problem with our documentation, please complete and return this form.

Pub. Title/Type SoftLogix5800 System User Manual

Cat. No. 1789-L10, -L30, -L60 Pub. No. 1789-UM002B-EN-P Pub. Date June 2001 Part No. 957564-11

Check Problem(s) Type:	Describe Problem(s)			Internal Use Only
☐ Technical Accuracy	☐ text		☐ illustration	
☐ Completeness	☐ procedure/step	☐ illustration	☐ definition	☐ info in manual
What information is missing?	☐ example	☐ guideline	☐ feature	(accessibility)
	☐ explanation	☐ other		☐ info not in manual
☐ Clarity				
What is unclear?				
☐ Sequence				
What is not in the right order?				
☐ Other Comments				
Use back for more comments.				

Your Name _____ Location/Phone _____

~~Return to: Marke~~ting Communications, Allen-Bradley., 1 Allen-Bradley Drive, Mayfield Hts., OH 44124-6118
Phone:(440) 646-3176 FAX:(440) 646-4320

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

Rockwell Automation

1 ALLEN-BRADLEY DR
MAYFIELD HEIGHTS OH 44124-9705

Reach us now at www.rockwellautomation.com

Wherever you need us, Rockwell Automation brings together leading brands in industrial automation including Allen-Bradley controls, Reliance Electric power transmission products, Dodge mechanical power transmission components, and Rockwell Software. Rockwell Automation's unique, flexible approach to helping customers achieve a competitive advantage is supported by thousands of authorized partners, distributors and system integrators around the world.

Americas Headquarters, 1201 South Second Street, Milwaukee, WI 53204, USA, Tel: (1) 414 382-2000, Fax: (1) 414 382-4444
European Headquarters SA/NV, avenue Herrmann Debroux, 46, 1160 Brussels, Belgium, Tel: (32) 2 663 06 00, Fax: (32) 2 663 06 40
Asia Pacific Headquarters, 27/F Citicorp Centre, 18 Whitfield Road, Causeway Bay, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication 1789-UM002B-EN-P - June 2001
PN 957564-11
Supersedes Publication 1789-UM002A-EN-P October 2000