Programming Manual

AB QUALITY *Allen-Bradley*

Logix5000 Controllers Produced and Consumed Tags

Catalog Numbers 1756 ControlLogix, 1756 GuardLogix, 1768 Compact GuardLogix, 1769 CompactLogix, 1789 SoftLogix, PowerFlex with DriveLogix

AB *Allen-Bradley* · *Rockwell Software*

Rockwell Automation

Important User Information

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

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

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

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

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

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

⚠	**WARNING:** Identifies information about practices or circumstances that can cause an explosion in a hazardous environment, which may lead to personal injury or death, property damage, or economic loss.
⚠	**ATTENTION:** Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss. Attentions help you identify a hazard, avoid a hazard, and recognize the consequence
⚡	**SHOCK HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that dangerous voltage may be present.
♨	**BURN HAZARD:** Labels may be on or inside the equipment, for example, a drive or motor, to alert people that surfaces may reach dangerous temperatures.
IMPORTANT	Identifies information that is critical for successful application and understanding of the product.

This manual contains new and updated information.

> **IMPORTANT** RSLogix 5000 programming software is now known as Studio 5000™ Logix Designer application, a component of Studio 5000 Engineering and Design Environment.

The following controllers are no longer supported in the Logix Designer application, version 21.

Catalog Number	Description
1756-L61	ControlLogix 5561 Controller
1756-L61S	ControlLogix 5561S Controller
1756-L62	ControlLogix 5562 Controller
1756-L62S	ControlLogix 5562S Controller
1756-L63	ControlLogix 5563 Controller
1756-L63S	ControlLogix 5563S Controller
1756-L64	ControlLogix 5564 Controller
1756-L65	ControlLogix 5565 Controller
1768-L43	CompactLogix 5343 Controller
1768-L43S	CompactLogix 5343S Controller
1768-L45	CompactLogix 5345 Controller
1768-L45S	CompactLogix 5345S Controller
1769-L23E-QBF1	CompactLogix 5323E-QB1 Controller
1769-L23E-QBFC1	CompactLogix 5323E-QBFC1 Controller
1769-L23-QBFC1	CompactLogix 5323-QBFC1 Controller
1769-L31	CompactLogix 5331 Controller
1769-L32C	CompactLogix 5332C Controller
1769-L32E	CompactLogix 5332E Controller
1769-L35CR	CompactLogix 5335CR Controller
1769-L35E	CompactLogix 5335E Controller

Changes throughout this revision are marked by change bars, as shown in the margin of this page.

Notes:

Table of Contents

Studio 5000 Engineering and Design Environment and Logix Designer Application

The Studio 5000™ Engineering and Design Environment combines engineering and design elements into a common environment. The first element in the Studio 5000 environment is the Logix Designer application. The Logix Designer application is the rebranding of RSLogix™ 5000 software and will continue to be the product to program Logix5000™ controllers for discrete, process, batch, motion, safety, and drive-based solutions.

The Studio 5000 environment is the foundation for the future of Rockwell Automation® engineering design tools and capabilities. It is the one place for design engineers to develop all the elements of their control system.

In This Manual

This manual shows how to configure and use produced and consumed tags. This manual is one of a set of related manuals that show common procedures for programming and operating Logix5000 controllers. For a complete list of common procedures manuals, see the Logix5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term *Logix5000 controller* refers to any controller that is based on the Logix5000 operating system, such as:

- CompactLogix controllers
- ControlLogix controllers
- DriveLogix controllers
- FlexLogix controllers
- SoftLogix5800 controllers

Notes:

Produce and Consume a Tag

Introduction

A Logix5000 controller lets you produce (broadcast) and consume (receive) system-shared tags.
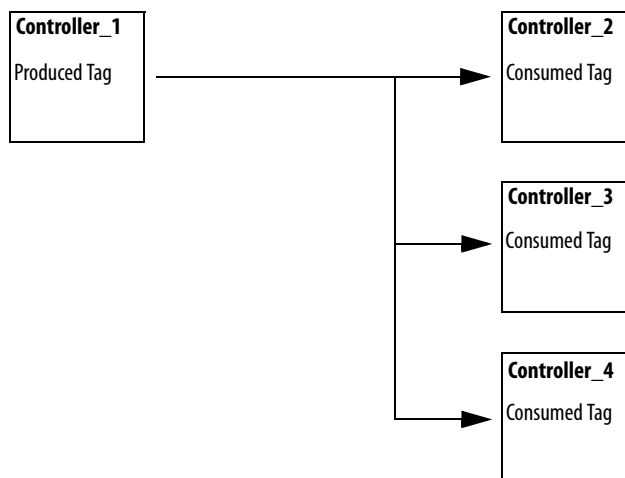
Term	Definition
Produced tag	A tag that a controller makes available for use by other controllers. Multiple controllers can simultaneously consume (receive) the data. A produced tag sends its data to one or more consumed tags (consumers) without using logic.
Consumed tag	A tag that receives the data of a produced tag. The data type of the consumed tag must match the data type (including any array dimensions) of the produced tag. The RPI of the consumed tag determines the period at which the data updates.

Controllers and Networks that Support Produced/Consumed Tags

These combinations support produced and consumed tags.

This controller	Can produce and consume tags over this network		
	Backplane	**ControlNet**	**EtherNet/IP**
SLC 500		X	
PLC-5		X	
1769 CompactLogix		X	X
1768 CompactLogix			X
ControlLogix	X	X	X
DriveLogix		X	X
FlexLogix		X	X
SoftLogix5800	X	X	X

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

Connection Requirements of a Produced or Consumed Tag

Produced and consumed tags each require connections. 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 communication and I/O.

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

Each produced or consumed tag uses these connections.

This Type of Tag	Uses This Many Connections
Produced tag	*number_of_consumers* +1
Consumed tag	1

EXAMPLE	Connection Requirements of a Produced or Consumed Tag
	•A FlexLogix controller producing a tag for 5 controllers (consumers) uses 6 connections.
	•A ControlLogix controller producing 4 tags for 1 controller uses 8 connections:
	– each tag uses 2 connections (1 consumer + 1 = 2).
	– 2 connections per tag x 4 tags = 8 connections.
	•Consuming 4 tags from a controller uses 4 connections (1 connection per tag x 4 tags = 4 connections).

Organize Tags for Produced or Consumed Data

As you organize your tags for produced or consumed data (shared data), follow these guidelines.

Guideline	Details		
Create the tags at the controller scope.	You can share only controller-scoped tags.		
Use one of these data types: • DINT • REAL • Array of DINTs or REALs • User-defined	• To share other data types, create a user-defined data type that contains the required data.		
	• Use the same data type for the produced tag and corresponding consumed tag or tags.		
To share tags with a PLC-5C controller, use a user-defined data type.	**To**	**This**	**Then**
	Produce	Integers	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.)
		Only one REAL value	Use the REAL data type.
		More than one REAL value	Create a user-defined data type that contains an array of REALs.
	Consume	Integers	Create a user-defined data type that contains these members.

To share tags with a PLC-5C controller — Consume — Integers:

Data Type	Description
DINT	Status
INT[x], where x is the output size of the data from the PLC-5C controller. (If you are consuming only one INT, omit x.)	Data produced by a PLC-5C controller

Guideline	Details
Limit the size of the tag to 500 bytes.	• If you must transfer more than 500 bytes, create logic to transfer the data in packets.
	• If you produce the tag over a ControlNet network, the tag may need to be less than 500 bytes. See Adjust for Bandwidth Limitations on page 11.
Use the highest permissible RPI for your application.	If the controller consumes the tag over a ControlNet network, use a binary multiple of the ControlNet network update time (NUT). For example, if the NUT is 5 ms, use an RPI of 5, 10, 20, 40 ms, and so forth.
Combine data that goes to the same controller.	If you are producing several tags for the same controller, group the data: • into one or more user-defined data types. (This uses less connections than producing each tag separately.) • according to similar update intervals. (To conserve network bandwidth, use a greater RPI for less critical data.) For example, you could create one tag for data that is critical and another tag for data that is not as critical.

Adjust for Bandwidth Limitations

When you share a tag over a ControlNet network, the tag must fit within the bandwidth of the 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 time (NUT).
- Since a ControlNet network can pass only 500 bytes in one NUT, the data of each connection must be less then 500 bytes to fit into the NUT.

Depending on the size of your system, you may not have enough bandwidth on your ControlNet network for a tag of 500 bytes. If a tag is too large for your ControlNet network, make one or more of these adjustments.

Adjustment	Description	
Reduce your network update time (NUT).	At a faster NUT, less connections have to share an update slot.	
Increase the requested packet interval (RPI) of your connections.	At higher RPIs, connections can take turns sending data during an update slot.	
For a ControlNet bridge module (CNB) 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 CNB 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, timestamp, or schedule 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.	1. 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. 2. Assign a different RPI to each tag.	
Create logic to transfer the data in smaller sections (packets).	See Produce a Large Array on page 29.	

Produce a Tag

Follow these steps to create a producer tag.

1. Open the Logix Designer application and select a controller.

> **IMPORTANT**   A controller can have both producing and consuming tags, but a producer cannot consume its own data. The local controller is the consumer, and the remote controller is the producer.

2. In the Controller Organizer, right-click the Controller Tags folder and choose Edit Tags.

   You can produce only controller-scoped tags.

3. In the Controller Tags window, right-click the tag that you want to be a producer, and then choose Edit (name of tag) Properties.

   The Tag Properties dialog box appears.
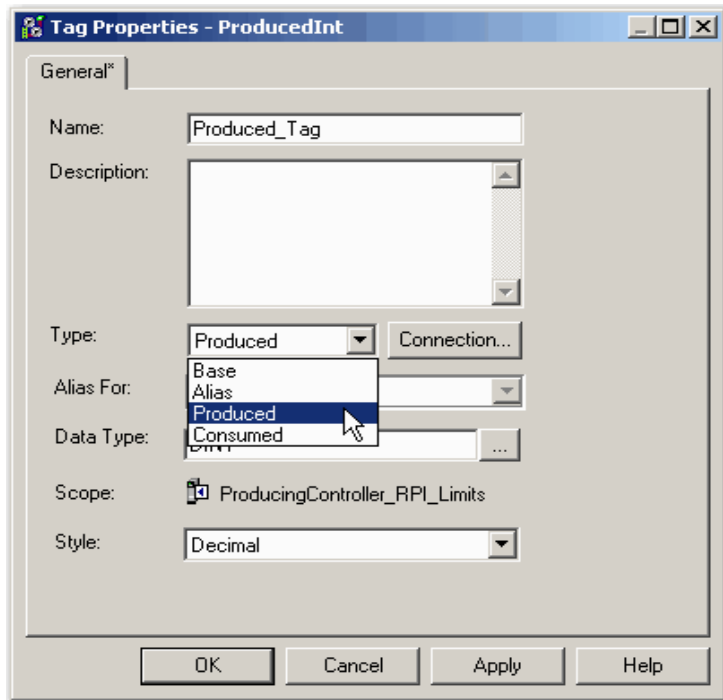
4. In the Name box, type the name of the produced tag.

5. From the Type pull-down menu, choose Produced.

   The Connection button becomes active when either Produced or Consumed is chosen from the pull-down menu.

6. Click Connection (at the right-hand side of the Type pull-down menu).

   The Produced Tag Connection dialog box appears to let you enter the number of controllers that will consume (receive) the tag.
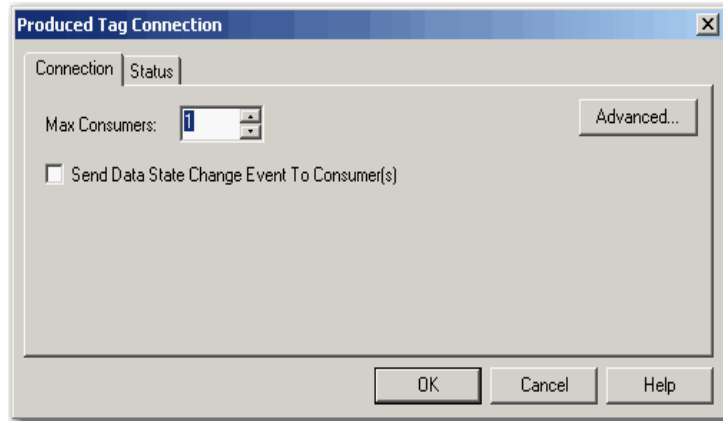
7. In the Max Consumers box, enter a value.

IMPORTANT	The Advanced button on the right-hand side of the Produced Tag Connection dialog box appears only for controllers using version 18 and later of the application. This button accesses the Advanced Options dialog box to set the RPI range limitation, negotiated default, or unicast connection. See page 17 for details and procedures.

8. Click OK.

Create a Consumer Tag

Follow these steps to create a consumer tag.

1. Open the Logix Designer application and select a controller.

| IMPORTANT	A controller can have both producing and consuming tags, but a producer cannot consume its own data. The local controller is the consumer, and the remote controller is the producer.

2. In the Controller Organizer I/O Configuration folder, add the controller that is producing the data (the other Logix5000 controller or PLC-5C controller).

3. In the Controller Organizer, right-click the Controller Tags folder and choose Edit Tags.

You can produce only controller-scoped tags.

4. In the Controller Tags window, right-click the tag that will consume the data, and then choose Edit (name of tag) Properties.

   The Tag Properties dialog box appears.
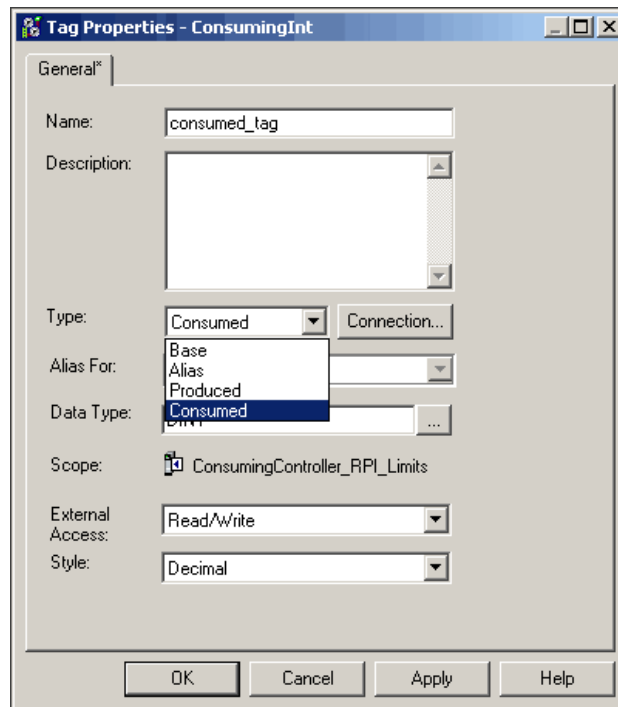
5. In the Name box, type the name of the consumed tag.

6. From the Type pull-down menu, choose Consumed.

   The Connection button becomes active when either Consumed or Produced is chosen from the pull-down menu.

7. Specify the data type.

If the producing controller is	Then the data type should be	
Logix5000 controller	Same data type as the produced tag.	
PLC-5C controller	User-defined data type with these members.	
	Data type	**Description**
	DINT	Status
	INT[x], where x is the output size of the data from the PLC-5C controller. (If you are consuming only one INT, omit x.)	Data produced by a PLC-5C controller

8. Click Connection to define the consumed tag.
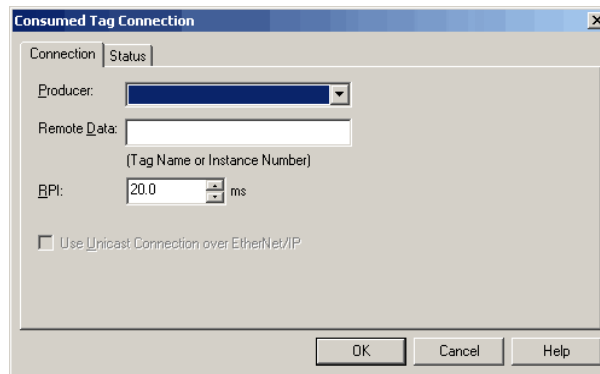
9. Do the following on the Consumed Tag Connection dialog box:

- Select the controller that produces the data.

  If the drop down is empty, you must first add a remote controller to the Controller Organizer I/O Configuration folder. This is the controller that is producing the data. If you have only one controller configured, it cannot be both the consumer and the producer. The producing controller cannot consume its own data.

- Type the name or instance number of the remote, produced data.

If the producing controller is	Then type or select
Logix5000 controller	Tag name of the produced tag
PLC-5C controller	Message number from the ControlNet configuration of the PLC-5C controller

- Type or select the requested packet interval (RPI) for the connection.

IMPORTANT	Consuming controllers have additional setup to allow its consuming tags to use an RPI provided by a producing controller. See page 19 for details.

10. Click OK.

11. If you consume the tag over a ControlNet network, use RSNetWorx for ControlNet software to schedule the network.

Additional Steps for a PLC-5C Controller

If you are sharing data with a PLC-5C controller, do the following.

Action	Details		
In the ControlNet configuration of the PLC-5C controller, schedule a message.	**If the PLC-5C controller**	**This**	**Then in RSNetWorx software**
	Produces	Integers	In the ControlNet configuration of the PLC-5C controller, insert a Send Scheduled Message.

	Consumes	Integers	Do these steps in the ControlNet configuration of the PLC-5C controller.
			  A. Insert a Receive Scheduled Message.
			  B. In the Message size, enter the number of integers in the produced tag.

		REALs	Do these steps in the ControlNet configuration of the PLC-5C controller.
			  A. Insert a Receive Scheduled Message.
			  B. In the Message size, enter two times the number of REALs in the produced tag. For example, if the produced tag contains 10 REALs, enter 20 for the Message size.

If the PLC-5C controller consumes REALs, reconstruct the values.	When you produce REALs (32-bit floating-point values) for a PLC-5C controller, the PLC-5C controller stores the data in consecutive 16-bit integers. • The first integer contains the upper (leftmost) bits of the value. • The second integer contains the lower (rightmost) bits of the value. • This pattern continues for each floating-point value.		

This example shows how to reconstruct a REAL (floating-point value) in the PLC-5C controller.

EXAMPLE   Reconstruct a floating point value.

The two MOV instructions reverse the order of the integers as the integers move to a new location. Because the destination of the COP instruction is a floating-point address, it takes two consecutive integers, for a total of 32 bits, and converts them to a single floating-point value.
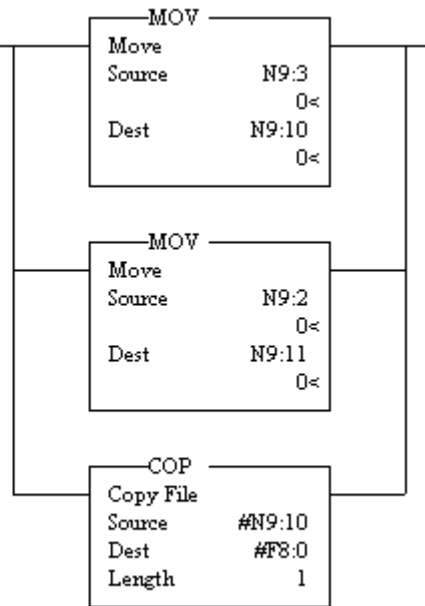
RPI Limitations and Negotiated Default

Controllers can be programmed to enhance the requested packet interval (RPI) acceptance between producing and consuming tags within specified limitations for multicast connections using version 18 and later of the application. A producing controller verifies that the RPI of incoming connections are within the produced tag settings. If the consuming tag's RPI falls outside the configured range, a producing controller will reject the incoming RPI and may provide an RPI (default) to the consuming controller.

This optional default RPI applies only if the consuming controller is set up to allow its consumed tags to use an RPI provided by the producer. You can choose to not use the default value. However, you will have to manually change a rejected RPI to make the value within the limitation settings for the produced tag.

IMPORTANT	1769-L2*x* and 1769-L3*x* controllers have additional RPI information. See <u>page 25</u> for details.

When using version 17 and earlier of the application, an incoming RPI of a consuming controller will be accepted by the producer. No default RPI is offered by the producing controller. The packet intervals of existing connections from other consumers could be sped up on multicast connections.

The RPI limitations (the range set on the produced tag) and the default process applies only to multicast connections on unscheduled networks. This feature is unavailable for controllers on the ControlNet network or unicast connections. Scheduled bandwidths on the ControlNet network transmit critical data at pre-determined intervals of time.

IMPORTANT	RPI limits and default values are not supported for safety tags and AXIS tags. AXIS produced tags are produced at a rate equal to the Coarse Update Period of the associated motion group. Safety produced tags are produced at a rate specified by the Safety Task Period.

Set RPI Limits, Default for Producer Tag

Follow these steps to establish RPI limitations (range) and default values if you are setting up a producing tag for multicast connections.

See <u>page 19</u> if you are setting up a unicast connection.

1. On the Produced Tag Connections dialog box, click Advanced.

   <u>Refer to Produce a Tag on page 12</u> to see an example of the dialog box.

The Advanced Options dialog box appears when you click Advanced. There are two connection options—multicast and unicast—on the dialog box with different functionality.
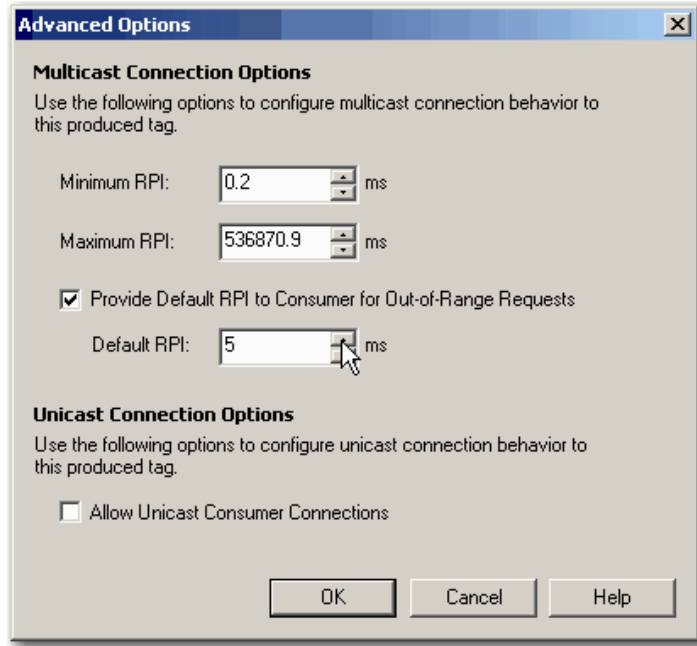
IMPORTANT	A unicast connection between a consumer and producer controller is the default on the Advanced Options dialog box. Multicast connections can be changed to a produced tag whether or not the box is checked. The checkbox dictates only whether this produced tag can accept a unicast connection. The consuming controller determines if the connection is unicast.

2. Complete the Advanced Options dialog box.

Multicast Connection Options	
Minimum RPI	Use the default of 0.2 ms.
Maximum RPI	Use the default of 536870.9 ms.
Provide Default RPI to Consumer for Out-of-Range Requests	Check the box to allow a producing tag to provide an RPI to a consuming tag whenever a consuming tag has an RPI that is outside the range of the producing tag. The consumer then connects with the RPI provided by the producer if set up for this functionality. If the box is blank, the producing controller will not return to the consumer an RPI value that is within the producer's RPI range.
Default RPI	Type a value that the producer will provide to the consumer when the consumer's RPI falls outside the range of the producer's RPI settings. The above box must be checked for the producer to provide the default.
Unicast Connection Options	See .

3. Click OK.

The Produced Tag Connections dialog box reappears.

4. Click OK.

IMPORTANT	The RPI Limits configuration, which is disabled when online, must be within: • minimum RPI: 0.196 ms. • maximum RPI: 536870.911 ms. • default RPI: configurable within the minimum/maximum range. • 1769-L2x and 1769-L3x controllers: see <u>page 25</u> for additional instructions. The RPI limits apply only to multicast connections on unscheduled networks. This functionality is not supported on the ControlNet network.

Unicast Connection Options

Unicast connections are point-to-point connections between a producing controller and consuming controller; no multiple connections. You do not have to enter a minimum or maximum RPI range or default value for this type of connection.

The Allow Unicast Consumer Connections box at the bottom of the Advanced Options dialog box defaults with a check mark. Click OK to complete setting up the producing tag.

Set Up the Consumer Tag

The Consuming controller must be set up to allow its consumed tags to accept an RPI provided by the producer. Follow these steps to access the Controller Properties dialog box.

1. In the Controller Organizer, right-click a controller that has been set up with a consuming tag and choose Properties.
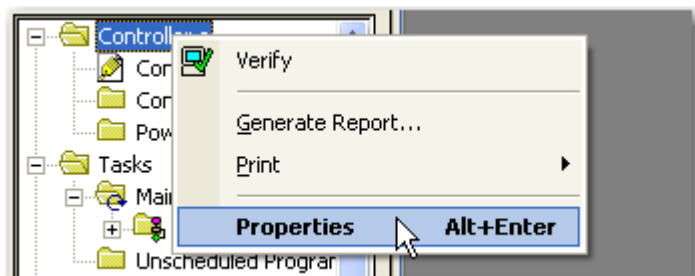
The Controller Properties dialog box appears with the General tab as the default.

2. Click the Advanced tab.
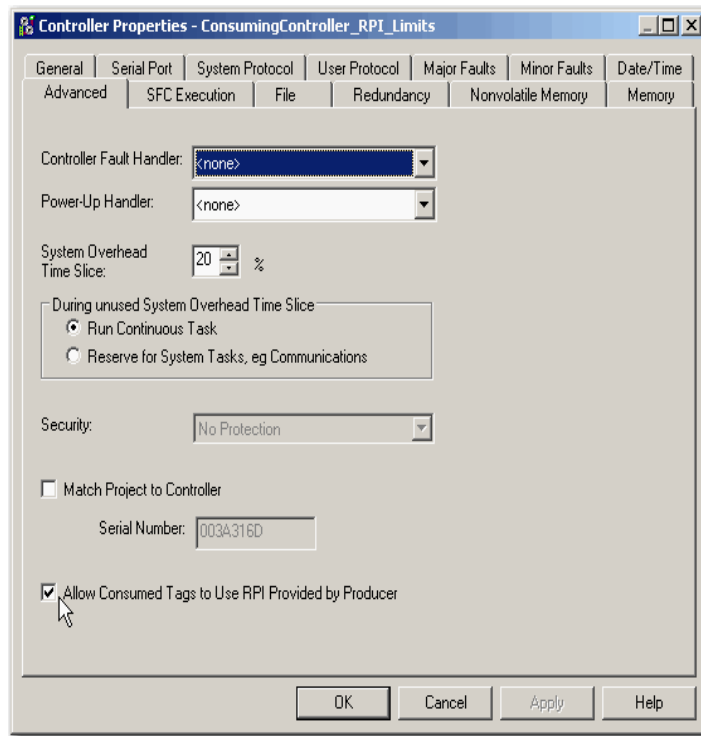
3. Check Allow Consumed Tags to Use RPI Provided by Producer.

4. Click OK.

IMPORTANT	The checkbox that activates 'RPI Provided by Producer' is disabled online. This box must be checked for the RPI negotiation to succeed. If the box is not checked, the consuming controller will not accept the RPI provided by the producer if the consuming controller requests an RPI that is out of the producer's configured RPI range. As a result, an out-of-range RPI request will result in a failed connection.

Verify Consumed Tag Acceptance

When a consuming controller accepts a default RPI by a producing controller, you can verify the acceptance and the RPI value on the Consumed Tag Connection dialog box. You must be online to complete the following procedures.

1. Open the Logix Designer application and select a controller that is set up with a consuming tag.

2. In the Controller Organizer, right-click the Controller Tags folder and choose Edit Tags.

3. In the Controller Tags window, right-click a consuming tag and choose Edit (name of tag) Properties.

The Tag Properties dialog box appears.
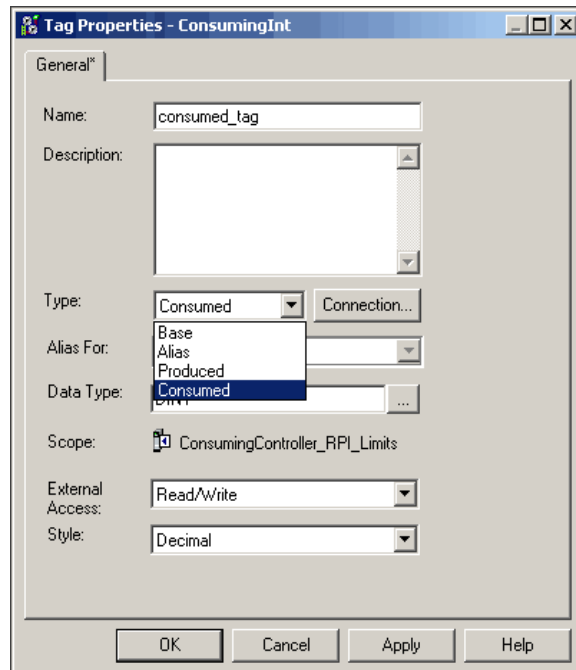
4. Click Connection (at the right-hand side of the Type pull-down menu).

The Consumed Tag Connection dialog box appears with a flag to the right of the RPI field to indicate the consuming controller has accepted an RPI provider by the producer. The time interval of the RPI also is indicated.
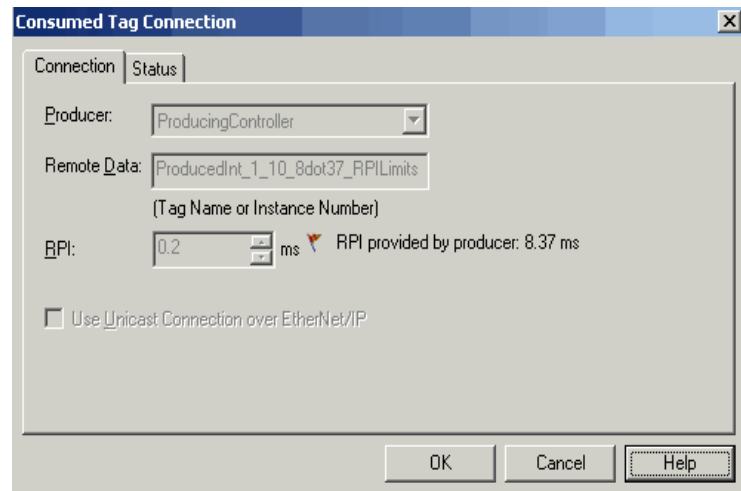
The RPI that had been requested by the consuming controller is dimmed. This lets you view the time interval the consuming controller had requested and the RPI provided by the producing controller.

5. Click Cancel to exit.

Produced and Consumed RPI Scenarios

The following scenarios explain how producing and consuming tags exchange RPI for controllers.

Scenario 1

The RPI is within range of the producing controller's RPI Limits.
There is no default RPI that is set up for the producing controller.

Produced Tag
Minimum RPI: 2 ms
Maximum RPI: 8 ms
Default RPI: - ms

Actual Packet Interval (API) = 3 ms

RPI = 3 ms
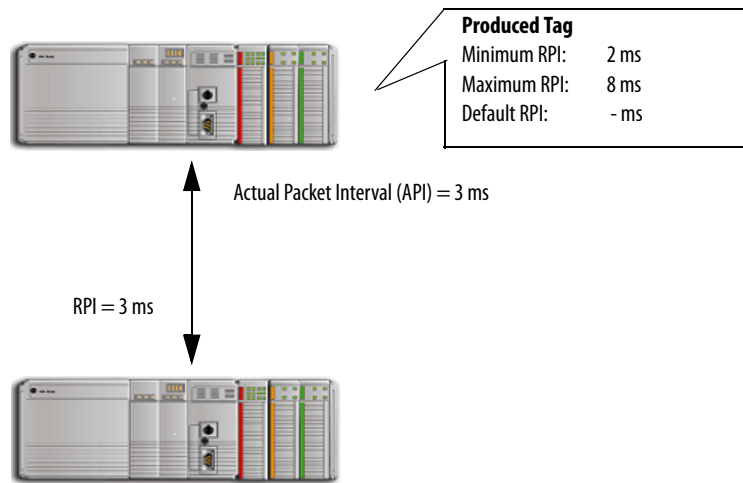
Version 17 and earlier: The consuming controller is brought online with an RPI configured at 3 ms. The producer accepts the RPI requested by the consuming tag. All controllers on the multicast connection will be increased to 3 ms for the API, which is the interval the data is actually being produced.

Version 18 and later: The consuming controller is brought online with an RPI configured at 3 ms. The producing controller verifies the requested RPI by the consumer is within the RPI limits set up for the producer. Since the request is within the range, the producing controller accepts the RPI of the consumer.

> **TIP**   If you use all the default settings of version 18 and later, you have the same behavior as version 17 and earlier.

Scenario 2

The RPI is outside the range of the producing controller's RPI Limits. There is no default RPI that is set up for the producing controller.

Produced Tag
Minimum RPI: 2 ms
Maximum RPI: 8 ms
Default RPI: - ms

Error: 0111

RPI = 1 ms
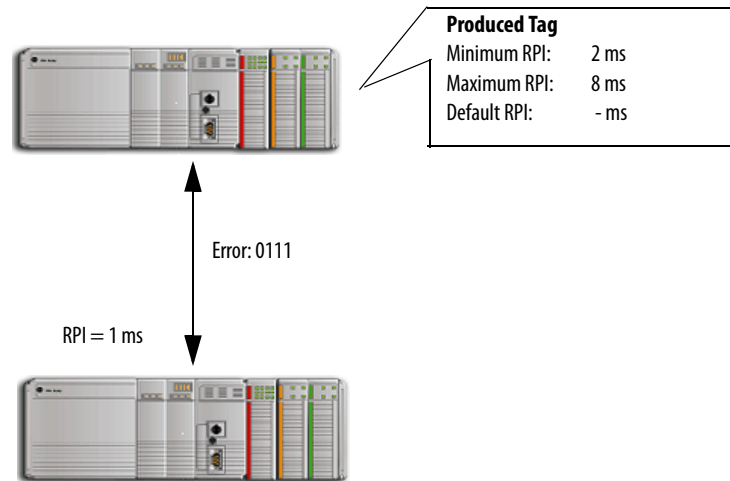
Version 17 and earlier: The RPI requested by the consumed tag is not within the range of the producer. The connection is rejected and a 0111 error is reported that the connection failed. You must re-enter an RPI that is within the range configured on the produced tag. If a version 17 consumer is trying to connect with a version 18 producer and the RPI is outside the range of the producer and there is no default set up, a 0111 error reports the connection failed.

> **TIP**   See RPI I/O Faults on page 27 for an explanation of the error messages.

Version 18 and later: The RPI requested by the consumed tag is not within the RPI limitation range of the producer. The producer is not set up to provide a default RPI, so a 0111 error message is reported that the connection failed. You must re-enter an RPI that is within the range configured for the producer because there is no RPI default negotiation with the consumer.

Scenario 3

The RPI is outside the range of the producing controller's RPI Limits. A default RPI is set up for the producing controller. But, the network path contains a bridge that blocks the negotiated RPI to prevent an interruption to other controllers on the network.

Produced Tag
Minimum RPI: 2 ms
Maximum RPI: 8 ms
Default RPI: 5 ms

Error: 0112, Default RPI = 5 ms

RPI = 1 ms

Negotiation Disabled

Version 17 and earlier: The RPI requested by the consumed tag is not within the range of the producer. The connection is rejected and a 0112 error is reported that the connection failed. You must re-enter an RPI that is within the range configured on the produced tag. If a version 17 consumer is trying to connect with a version 18 producer and the RPI is outside the range of the producer, a 0112 error reports the connection failed.

> **TIP**   See RPI I/O Faults on page 27 for an explanation of the error messages.

Version 18 and later: The RPI requested by the consumed tag is not within the RPI limitation range of the producer. The producer is configured to provide an RPI default. The producer sends the default RPI to the consumer, but the consumer controller is not set up to accept a default RPI from the producer. The negotiated RPI is disabled to prevent an interruption to the multicast connection. A 0112 error message is reported and you must re-enter an RPI within the range of the producer's limits.

Scenario 4

The RPI is outside the range of the producing controller's RPI Limits. A default RPI is set up for the producing controller. The consumer is set up to accept the RPI provided by the producer.
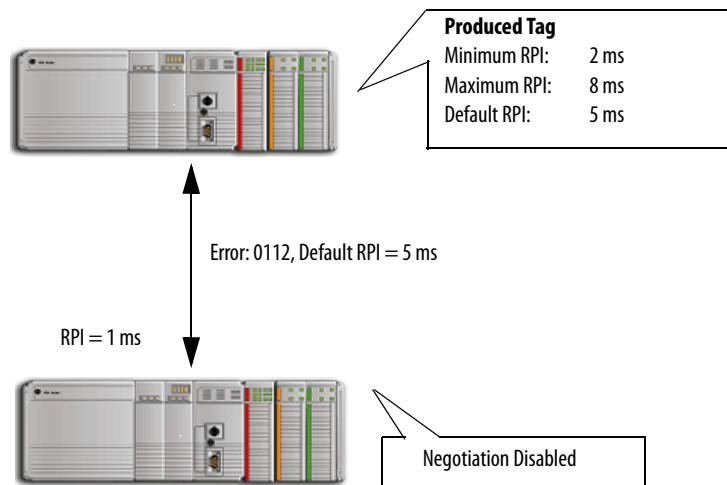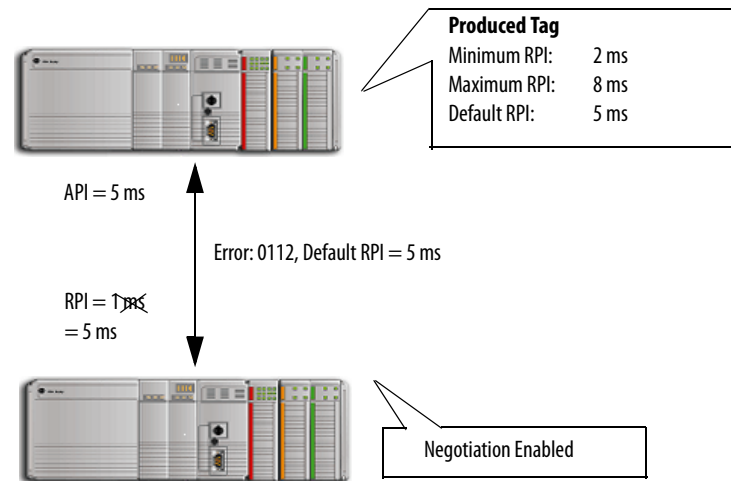

Vrsion 17 and earlier: The RPI requested by the consumed tag is not within the range of the producer. The connection is rejected and a 0112 error is reported that the connection failed. You must re-enter an RPI that is within the range configured on the produced tag. If a version 17 consumer is trying to connect with a version 18 producer and the RPI is outside the range of the producer, a 0112 error reports the connection failed.

Version 18 and later: The RPI requested by the consumed tag is not within the RPI limitation range of the producer. The producer is configured to provide an RPI default. The producer sends the default RPI to the consumer. The consumer is set up to accept the RPI provided by the producer. The connection is successful and the multicast connection accepts the new RPI of 5 ms.

1769-L2*x* and 1769-L3*x* RPI Limits

All controllers, with the exception of the CompactLogix 1769-L2*x* and 1769-L3*x* controllers, produce data in packet intervals ranging from 0.196…536870.911 ms. The 1769-L2*x* and 1769-L3*x* controllers produce only in packet intervals of 2, 4, 8, 16, 32, 64, 128, 256, 512, or 1024 ms.

You are not confined to entering these exact values when configuring an RPI limitation range and default RPI for the 1769-L2*x* and 1769-L3*x* controllers. However, if an unsupported packet interval is entered, then the Logix Designer application may override the unsupported packet interval with a supported

packet interval. When this occurs, the new 'effective' value is displayed separately next to the unsupported value that was entered.

Similar to other controllers, the 1769-L2*x* and 1769-L3*x* controllers will verify that the RPI of incoming connections are within the produced tag settings. If the consuming tag's RPI falls outside the configured range, a producing controller will reject the incoming RPI and then provide an RPI (default) to the consuming controller.

IMPORTANT	The producing and consuming controllers must be set up to allow the consumed tags to use an RPI provided by the producer.
	See <u>RPI Limitations and Negotiated Default</u> and <u>Set Up the Consumer Tag</u> for procedures to set up the producer and consumer tags to accept a negotiated default RPI.

For the 1769-L2*x* and 1769-L3*x* controllers, the Effective Minimum RPI, when present, is used to determine the fastest packet interval allowed by the tag. When the Effective Default RPI is present, it is the largest packet interval (slowest rate) at which negotiated connections will be produced for the tag.

The effective RPI limit values are presented with a flag on the Advanced Options dialog box to indicate that RPI limits are set to values other than what you entered.
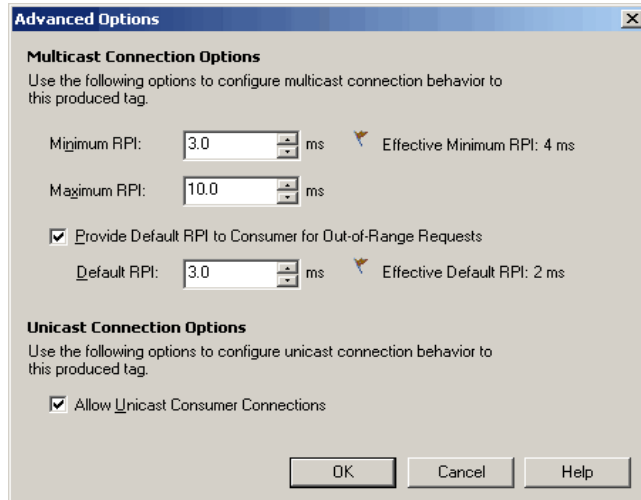
RPI I/O Faults

The table provides a description of error messages for situations where an RPI is not accepted.

Scenario Description	Software Version	I/O Fault Message
RPI of consumed tag is not within the range of the producer. Producer does not support RPI negotiation or is not configured to provide a default RPI to the consumer.	Any	(Code 0111) Requested Packet Interval (RPI) out of range.
RPI of consumed tag is not within the range of the producer. Producer is returning default RPI. Consumer does not support negotiation or is not configured to accept RPI from producer.	18 and later	(Code 0112) Requested Packet Interval (RPI) out of range.
	17 and earlier	(Code 0112) Unknown Error.
RPI of consumer is out of range, but producer provides a default RPI and consumer can accept default RPI. Network path contains bridge that does not support extended error information.	18 and later	(Code 0112) Requested Packet Interval (RPI) out of range. Producer provided RPI blocked by the network path.
RPI of consumer is out of range, but producer provides a default RPI and consumer can accept RPI provided by producer. Network path contains scheduled NUT.	18 and later	(Code 0112) Requested Packet Interval (RPI) out of range. Cannot use producer provided RPI over a path containing a scheduled network.

Notes:

Produce a Large Array

Introduction

The Logix5000 controller can send as many as 500 bytes of data over a single scheduled connection. This corresponds to 125 DINT or REAL elements of an array. To transfer an array of more than 125 DINTs or REALs, use a produced/consumed tag of 125 elements to create a packet of data. You can then use the packet to send the array piecemeal to another controller.

When you send a large array of data in smaller packets, you must be sure that the transmission of a packet is complete before the data is moved into the destination array.

- Produced data over the ControlLogix backplane is sent in 50 byte segments.
- Data transmission occurs asynchronous to a program scan.

The logic that this section includes uses an acknowledge word to make sure that each packet contains new data before the data moves to the destination array. The logic also uses an offset value to indicate the starting element of the packet within the array.

Because of the offset and acknowledge elements, each packet carries 123 elements of data from the array, as depicted below.

Figure 1 - Producer/Consumer Data Elements
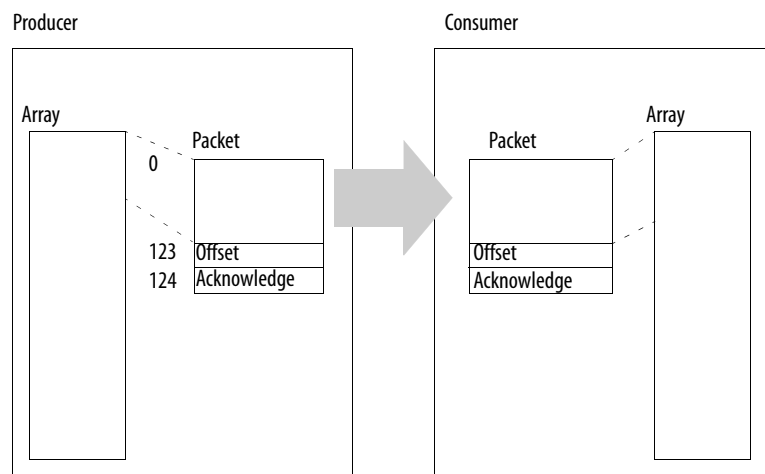
In addition, the array must contain an extra 122 elements. In other words, it must be 122 elements greater than the greatest number of elements that you want to transfer.

- These elements serve as a buffer.

- Because each packet contains the same number of elements, the buffer prevents the controller from copying beyond the boundaries of the array.

- Without the buffer, this would occur if the last packet contained fewer than 123 elements of actual data.

Produce a Large Array

Follow these steps to produce a large array.

1. In the Controller Tags folder of the controller project that produces the array, create these tags.

P	Tag Name	Type
	array_ack	DINT[2]
X	array_packet	DINT[125]

2. Convert array_ack to a consumed tag.

For	Specify
Controller	Name of the controller that is receiving the packet.
Remote Tag Name	array_ack Both controllers use the same name for this data.

3. In either the Controller Tags folder or the tags folder of the program that will contain the logic for the transfer, create these tags.

Tag Name	Type
array	DINT[x] where x equals the number of elements to transfer plus 122 elements
array_offset	DINT
array_size	DINT
array_transfer_time	DINT
array_transfer_time_max	DINT
array_transfer_timer	TIMER

4. In the array_size tag, enter the number of elements of real data.

   The value of x from step 3 minus the 122 elements of buffer.

5. Create or open a routine for the logic that will create packets of data.

6. Enter this logic.

When the offset value in array_ack[0] is not equal to the current offset value but array_ack[1] equals -999, the consumer has begun to receive a new packet, so the rung moves -999 into the last element of the packet. The consumer waits until it receives the value -999 before it copies the packet to the array. This guarantees that the consumer has new data.

```
        ┌─────NEQ──────────────────┐       ┌─────EQU──────────────────┐                              ┌─────MOV──────────────────┐
        │ Not Equal                │       │ Equal                    │                              │ Move                     │
 ───────┤ Source A        array_ack[0]├───┤ Source A  array_ack[1]    ├──────────────────────────────┤ Source              -999 ├───
        │                         0 │       │                       0 │                              │ Dest  array_packet[124]  │
        │ Source B  array_packet[123]│      │ Source B           -999 │                              │                        0 │
        │                         0 │       └──────────────────────────┘                             └──────────────────────────┘
        └──────────────────────────┘
```

When the offset value in array_ack[0] is equal to the current offset value, the consumer has copied the packet to the array; so the rung checks for more data to transfer. If the offset value plus 123 is less than the size of the array, there is more data to transfer; so the rung increases the offset by 123. Otherwise, there is no more data to transfer; so the rung resets the offset value, logs the transfer time, and resets the timer. In either case, the rung uses the new offset value to create a new packet of data, appends the new offset value to the packet, and clears the acknowledge element of the packet (packet[124]).

```
                                                                        ┌─────TON───────────────────────────┐
                                                                        │ Timer On Delay                   ─(EN)─
                                                                        │ Timer        array_transfer_timer │
 ───────────────────────────────────────────────────────────────────────┤ Preset            10000000        │
                                                                        │ Accum                    0       ─(DN)─
                                                                        └───────────────────────────────────┘

        ┌─────NEQ──────────────────┐       ┌─────EQU──────────────────┐                              ┌─────MOV──────────────────┐
        │ Not Equal                │       │ Equal                    │                              │ Move                     │
 ───────┤ Source A        array_ack[0]├───┤ Source A  array_ack[1]    ├──────────────────────────────┤ Source              -999 ├───
        │                         0 │       │                       0 │                              │ Dest  array_packet[124]  │
        │ Source B  array_packet[123]│      │ Source B           -999 │                              │                        0 │
        │                         0 │       └──────────────────────────┘                             └──────────────────────────┘
        └──────────────────────────┘

        ┌─────EQU──────────────────┐       ┌──────────CMP──────────────────────────┐     ┌─────ADD──────────────────────┐
        │ Equal                    │       │ Compare                               │     │ Add                          │
 ───────┤ Source A        array_ack[0]├───┤ Expression  (array_ack[0] + 123) < array_size├──┤ Source A  array_packet[123]  ├──
        │                         0 │       └────────────────────────────────────────┘     │                        0     │
        │ Source B  array_packet[123]│                                                      │ Source B             123     │
        │                         0 │                                                       │ Dest          array_offset   │
        └──────────────────────────┘                                                        │                        0     │
                                                                                            └──────────────────────────────┘

                                      ┌──────────CMP──────────────────────────┐             ┌─────CLR──────────────┐
                                      │ Compare                               │             │ Clear                │
                                   ───┤ Expression  (array_ack[0] + 123) >= array_size├──────┤ Dest    array_offset │
                                      └────────────────────────────────────────┘             │                    0 │
                                                                                             └──────────────────────┘

                                                                                             ┌─────MOV──────────────────────────┐
                                                                                             │ Move                             │
                                                                                             │ Source  array_transfer_timer.ACC │
                                                                                             │                               0  │
                                                                                             │ Dest          array_transfer_time│
                                                                                             │                               0  │
                                                                                             └──────────────────────────────────┘

                                                                                                    array_transfer_timer
                                                                                                        ─(RES)─

        ┌─────COP──────────────────────┐     ┌─────MOV──────────────────┐     ┌─────CLR──────────────────┐
        │ Copy File                    │     │ Move                     │     │ Clear                    │
     ───┤ Source  array[array_offset]  ├─────┤ Source        array_offset├────┤ Dest  array_packet[124]  ├──
        │ Dest       array_packet[0]   │     │                       0  │     │                        0 │
        │ Length            123        │     │ Dest  array_packet[123]  │     └──────────────────────────┘
        └──────────────────────────────┘     │                       0  │
                                             └──────────────────────────┘
```
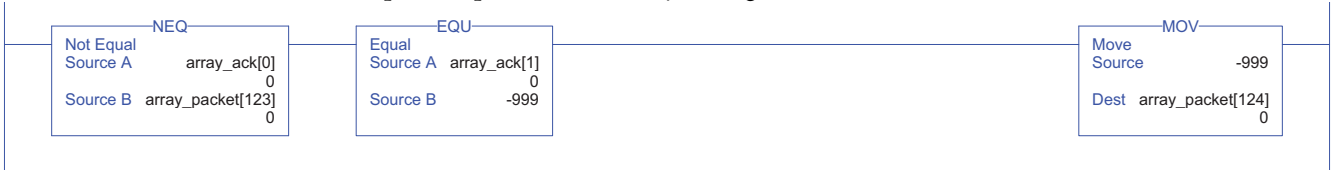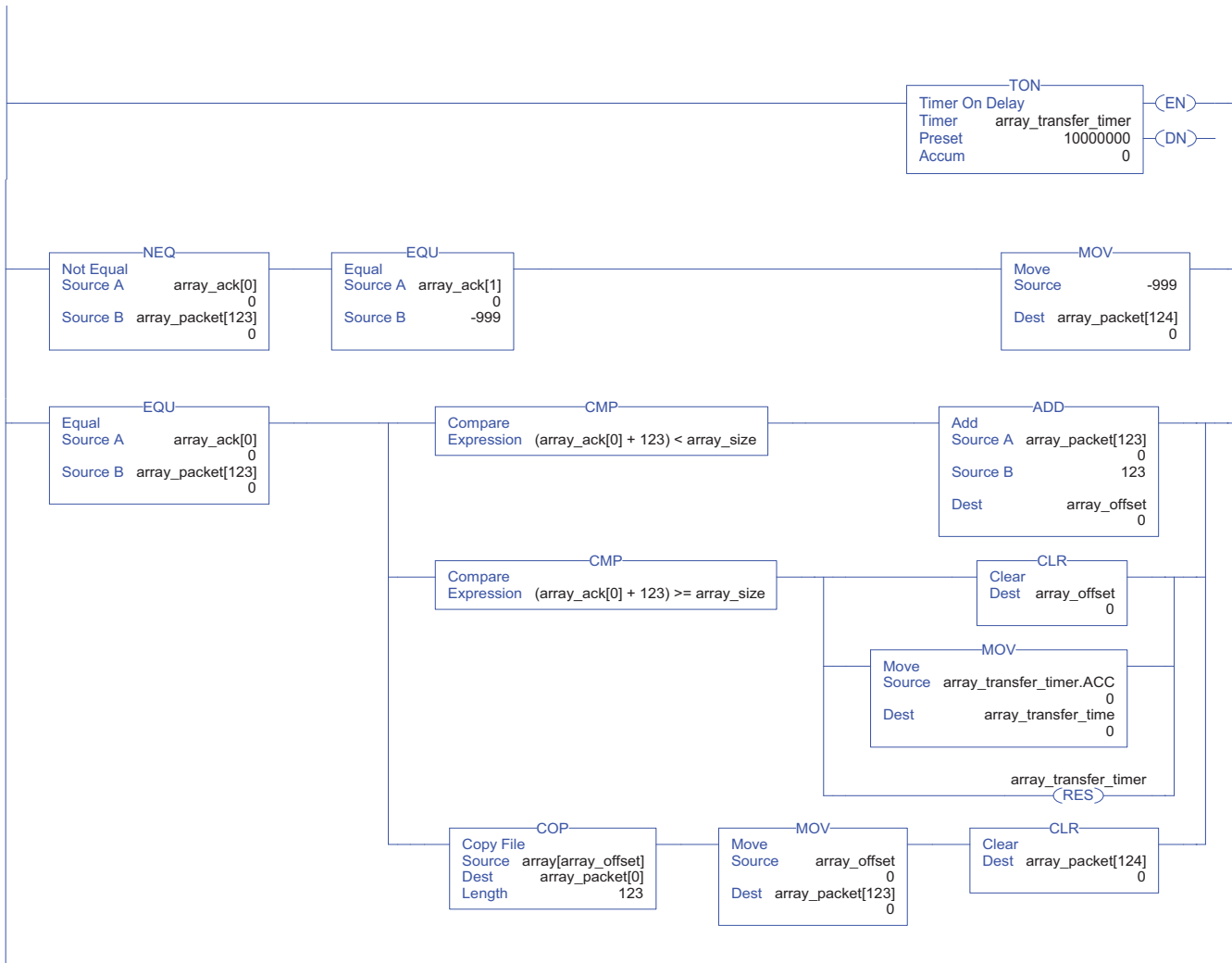
If the current transfer time is greater than the maximum transfer time, update the maximum transfer time. This maintains a record of the longest time to transfer data.

```
                    GRT                                                                                    MOV
       Greater Than (A>B)                                                                    Move
       Source A         array_transfer_time                                                 Source        array_transfer_time
                                          0                                                                                 0
       Source B   array_transfer_time_max                                                   Dest   array_transfer_time_max
                                          0                                                                                 0
```

7. In the Controller Tags folder of the controller project that consumes the array, create these tags.

P	Tag Name	Type
X	array_ack	DINT[2]
	array_packet	DINT[125]

8. Convert array_packet to a consumed tag.

For	Specify
Controller	Name of the controller that is sending the packet.
Remote tag name	array_packet Both controllers use the same name for this data.

9. In either the Controller Tags folder or the tags folder of the program that will contain the logic for the transfer, create these tags.

Tag Name	Type
array	DINT[x] where x equals the number of elements to transfer plus 122 elements
array_offset	DINT

10. Create or open a routine for the logic that will move the data from the packets to the destination array.

11. Enter this logic.

When the offset value in array_packet[123] is different than the offset value in array_ack[0], the controller has begun to receive a new packet of data; so the rung checks for the value of -999 in the last element of the packet.

If the last element of the packet equals -999, the controller has received an entire packet of new data and begins the copy operation.

- The offset value moves from the packet to array_offset.
- The COP instructions copy the data from the packet to the destination array, starting at the offset value.
- The offset value moves to array_ack[0], which signals that the copy is complete.
- Array_ack[1] resets to zero and waits to signal the arrival of a new packet.

If the last element of the packet is not equal to -999, the transfer of the packet to the controller may not be complete; so -999 moves to array_ack[1]. This signals the producer to return the value of -999 in the last element of the packet to verify the transmission of the packet.
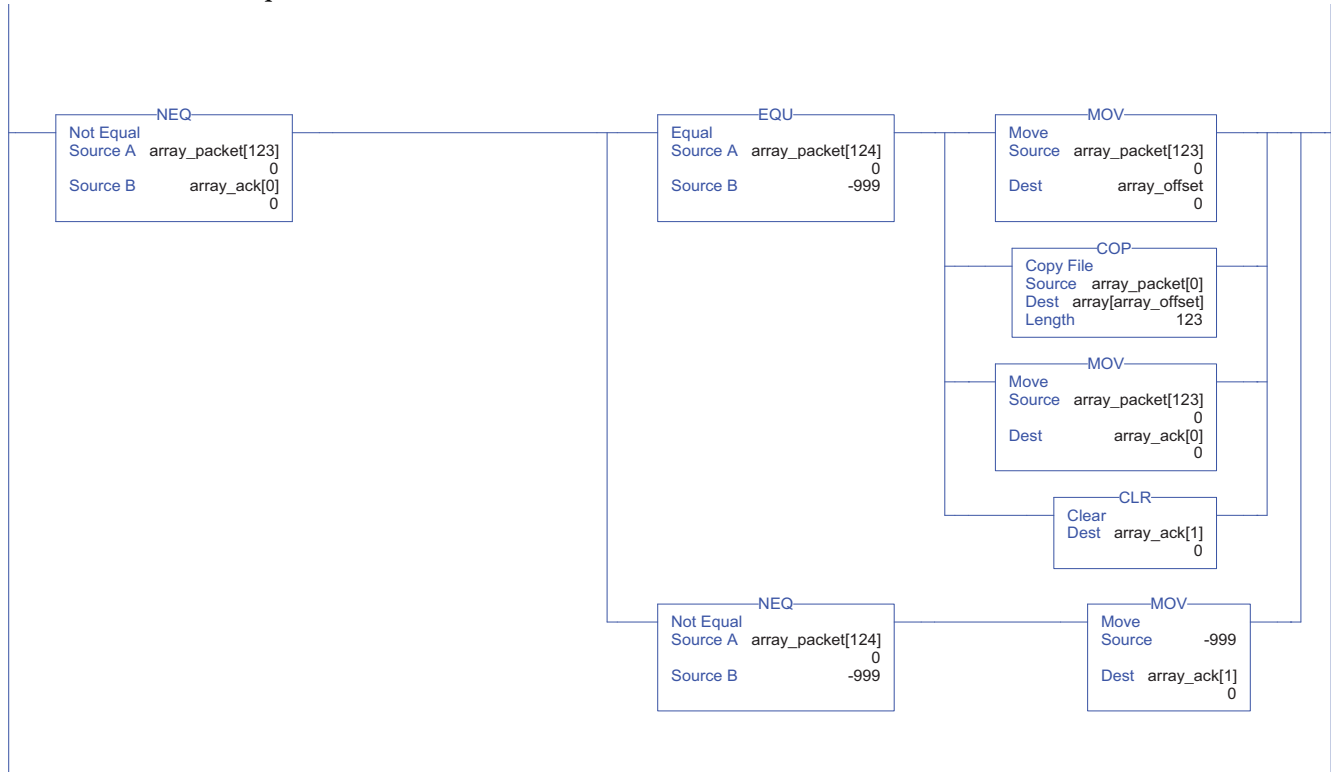
44948

Transferring a large array as smaller packets improves system performance over other methods of transferring the data.

- Fewer connections are used than if you broke the data into multiple arrays and sent each as a produced tag. For example, an array with 5000 elements would take 40 connections (5000/125=40) by using individual arrays.

- Faster transmission times are achieved than if you used a message instruction to send the entire array.

    – Messages are unscheduled and are executed only during the 'system overhead' portion of the Logix5550 execution. Therefore, messages can take a fairly long time to complete the data transfer.
    – You can improve the transfer time by increasing the system overhead time slice, but this diminishes the performance of the continuous task.

Rockwell Automation Support

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

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

Installation Assistance

If you experience a problem within the first 24 hours of installation, review the information that is contained in this manual. You can contact Customer Support for initial help in getting your product up and running.

United States or Canada	1.440.646.3434
Outside United States or Canada	Use the Worldwide Locator at http://www.rockwellautomation.com/support/americas/phone_en.html, or contact your local Rockwell Automation representative.

New Product Satisfaction Return

Rockwell Automation tests all of its products to ensure that they are fully operational when shipped from the manufacturing facility. However, if your product is not functioning and needs to be returned, follow these procedures.

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

Documentation Feedback

Your comments will help us serve your documentation needs better. If you have any suggestions on how to improve this document, complete this form, publication RA-DU002, available at http://www.rockwellautomation.com/literature/.