Logix 5000 Controller Messages

1756 ControlLogix, 1756 GuardLogix, 1769 CompactLogix, 1769 Compact GuardLogix, 1789 SoftLogix, 5069 CompactLogix, Emulate 5570

Rockwell Automation

Programming Manual

Original Instructions

Important User Information

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The following icon may appear in the text of this document.

Identifies information that is useful and can help to make a process easier to do or easier to understand.

Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Summary of changes

This manual includes new and updated information. Use these reference tables to locate changed information.

Grammatical and editorial style changes are not included in this summary.

Global changes

Subject	Reason
Throughout	Removed references to the CompactLogix 5480 controller. The Logix Designer application versions 38 and later do not support it.

New or changed information

Subject	Reason
Preface on page 5	Corrected the link to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.
Additional resources on page 5	Added a reference to Logix 5000 Controllers Design Considerations, publication 1756-RM094.
Introduction to Controller Messages on page 7	Added a reference to Logix 5000 Controllers Design Considerations, publication 1756-RM094.
Cache list on page 9	Added an important note regarding messages that execute at a high frequency.
Guidelines on page 11	Added a guideline recommending read messages instead of write messages.

New or enhanced features

Subject	Reason
Throughout	Added references to the ControlLogix 5590 controller. The Logix Designer application versions 38 and later support it.

Table of Contents

Controller messages....7

Introduction to Controller Messages....7

Supported data types....7

Message Queue....8

Cache list....9

Unconnected buffers....11

Guidelines....11

Get or set the number of unconnected buffers....13

Get the number of unconnected buffers....13

Set the number of unconnected buffers....14

Convert between INTs and DINTs....16

Manage multiple messages....18

Introduction....18

Message manager logic....18

Send a message to multiple controllers....21

Introduction....21

Configure the I/O configuration....22

Define your source and destination elements....22

Create the MESSAGE_ CONFIGURATION data type....23

Create the configuration array....24

Get the size of the local array....25

Load the message properties for a controller....26

Configure the message....26

Step to the next controller....27

Preface

This manual shows how to program message (MSG) instructions to and from Logix 5000™ controllers. This manual is one of a set of related manuals that show common procedures for programming and operating Logix 5000 controllers.

For a complete list of common procedures manuals, refer to the Logix 5000 Controllers Common Procedures Programming Manual, publication 1756-PM001.

The term Logix 5000 controller refers to any controller based on the Logix 5000 operating system. Rockwell Automation recognizes that some of the terms that are currently used in our industry and in this publication are not in alignment with the movement toward inclusive language in technology. We are proactively collaborating with industry peers to find alternatives to such terms and making changes to our products and content. Please excuse the use of such terms in our content while we implement these changes.

Studio 5000 environment

The Studio 5000 Automation Engineering & Design Environment® combines engineering and design elements into a common environment. The first element is the Studio 5000 Logix Designer® application. The Logix Designer application is the rebranding of RSLogix 5000® software and will continue to be the product to program Logix 5000™ 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. The Studio 5000 environment is the one place for design engineers to develop all elements of their control system.

Additional resources

These documents contain additional information concerning related Rockwell Automation products.

Resource	Description
Industrial Automation Wiring and Grounding Guidelines, publication, 1770-4.1	Provides general guidelines for installing a Rockwell Automation industrial system.
Logix 5000™ Controllers Design Considerations, publication 1756-RM094.	Provides information to help design and plan Logix 5000 systems.

Resource	Description
Rockwell Automation product certifications	Provides declarations of conformity, certificates, and other certification details.

View or download publications at https://www.rockwellautomation.com/en-us/support/documentation/literature-library.html. To order paper copies of technical documentation, contact a local Rockwell Automation distributor or sales representative.

Legal notices

Rockwell Automation publishes legal notices, such as privacy policies, license agreements, trademark disclosures, and other terms and conditions on the Legal Notices page of the Rockwell Automation website.

Software and Cloud Services Agreement

Review the Rockwell Automation Software and Cloud Services Agreement here.

Open Source Software Licenses

The software included in this product contains copyrighted software that is licensed under one or more open source licenses.

You can view a full list of all open source software used in this product and their corresponding licenses at this URL:

Studio 5000 Logix Designer Open Source Attribution List

You may obtain Corresponding Source code for open source packages included in this product from their respective project web site(s). Alternatively, you may obtain complete Corresponding Source code by contacting Rockwell Automation via the **Contact** form on the Rockwell Automation website: http://www.rockwellautomation.com/global/about-us/contact/contact.page. Please include "Open Source" as part of the request text.

Controller messages

Introduction to Controller Messages

This section describes how to transfer (send or receive) data between controllers by executing a message (MSG) instruction. It explains cache connections and buffers so you can correctly program the controller.

For a comparison of Produced/Consumed tags, see Compare Messages and Produced/Consumed Tags in Logix 5000 Controllers Design Considerations, publication 1756-RM094.

Supported data types

The following data types are supported when sending CIP messages.

- SINT
- INT
- DINT
- LINT
- REAL

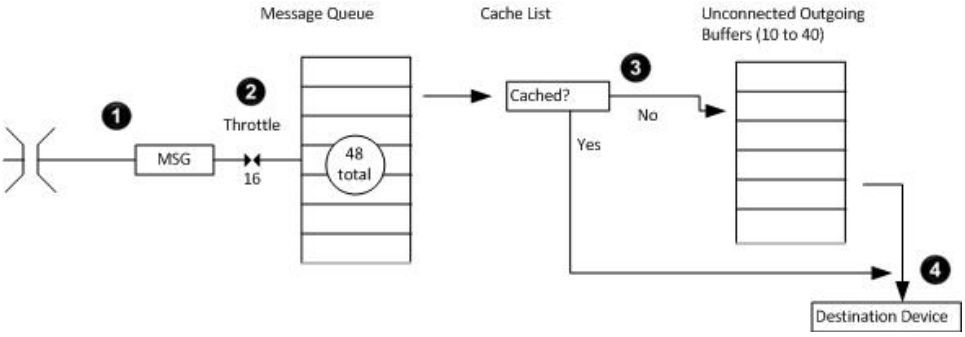
In addition, you can send a message with any structure type that is predefined, module-defined, or user-defined.

For more information, see Convert between INTs and DINTs on page 16.

For complete details on programming a message instruction, see the LOGIX 5000 Controllers General Instruction Reference Manual, publication 1756-RM003.

Example:	**Execute a message (MSG) instruction** If count_send = 1 and count_msg.EN = 0 (MSG instruction is not enabled) then execute a MSG instruction that sends data to another controller.

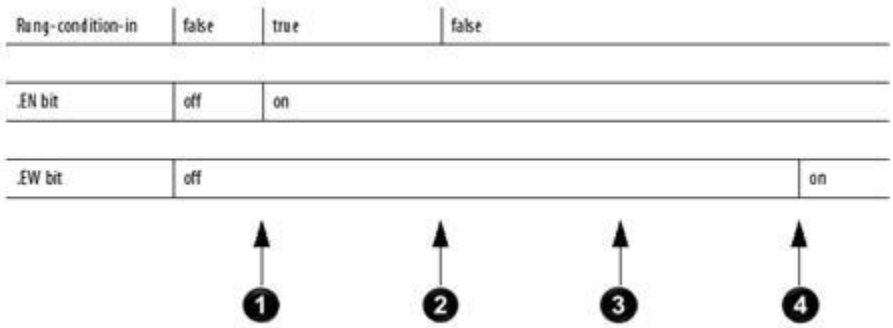
This diagram shows how the controller processes MSG instructions.



Description		
1	The controller scans the MSG instruction and its rung-condition-in goes true. The message passes to a throttle that has 16 positions. If the throttle is full, the message remains enabled but is held until another controller scan.	
2	The System-overhead time slice executes and the message is pulled from the throttle to the message queue.	
3	If the MSG instruction	Then the MSG instruction
	Does not use a connection or the connection was not previously cached	Uses an unconnected buffer to establish communication with the destination device.
	Uses a connection and the connection is cached	Does not use an unconnected buffer.
4	Communication occurs with the destination device.	

Message Queue

The message queue holds up to 48 MSG instructions, including those that you configure as a block-transfer read or block-transfer write. When the queue is full, an instruction tries to enter the queue on each subsequent scan of the instruction, as shown in the following illustration.



Description

1	The controller scans the MSG instruction. The rung-condition-in for the MSG instruction is true. The EN bit is set. The MSG instruction attempts to enter the queue but 16 throttle positions exist. If all 16 are filled and a 17th message is executed, the message is enabled. The EW bit remains cleared.
2 & 3	The controller scans the MSG instruction. The rung-condition-in for the MSG instruction is false. The EN bit remains set. The MSG instruction attempts to pass through the throttle, but no open positions exist yet. The EW bit remains cleared.
4	The controller scans the MSG instruction. The MSG instruction attempts to enter the queue. This time the throttle position is open and the message can pass to the message queue. The EW bit is set.

Cache list

Depending on how you configure a MSG instruction, it may use a connection to send or receive data.

IMPORTANT: For messages that execute at a high frequency to the same device, configure the messages for connected and cached.

This type of message	And this communication method	Uses a connection
CIP data table read or write	–	Your option(1)
PLC-2, PLC-3, PLC-5, or SLC (all types)	CIP CIP with Source ID	No
	DH+	Yes
CIP generic	–	Your option(2)
Block-transfer read or write	–	Yes

1 2

1. CIP data table read or write messages can be connected or unconnected. However for most applications, it is recommended you leave CIP data table read or write messages connected.
2. CIP generic messages can be connected or unconnected. However for most applications, it is recommended you leave CIP generic messages unconnected, unless you want to use the Large Connection option.

If a MSG instruction uses a connection, you have the option to leave the connection open (cache) or close the connection when the message is done transmitting.

If you	Then
Cache the connection	The connection stays open after the MSG instruction is done. This optimizes execution time. Opening a connection each time the message executes increases execution time.
Do not cache the connection	The connection closes after the MSG instruction is done. This frees up that connection for other uses.

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

If you have this software version and firmware revision	Then you can cache
11.x or earlier	• Block transfer messages for up to 16 connections. • Other types of messages for up to 16 connections.
12.x or later	Up to 32 connections.

If several messages go to the same device, the messages may be able to share a connection.

If the MSG instructions are to	And they are	Then
Different devices	—	Each MSG instruction uses 1 connection.
The same device, cached, and not a large connection	Enabled simultaneously (same scan)	Each MSG instruction uses 1 connection and 1 cached buffer.
	Not enabled simultaneously	All MSG instructions use 1 connection and 1 cached buffer. They share the connection and the buffer.
The same device, cached, and a large connection	Enabled simultaneously (same scan)	Each MSG instruction uses 1 connection and 1 cached buffer.
	Not enabled simultaneously	All MSG instructions use 1 connection and 1 cached buffer. They share the connection and the buffer.

Share a connection

- If the controller alternates between sending a block-transfer read message and a block-transfer write message to the same module, then together the messages count as one connection. Caching both messages counts as one on the cached buffer.
- If the controller sends 10 cached connected messages to the same bridge module (for example, 1756-EN2T) where 7 utilize a standard connection (large connection unchecked) and 3 utilize a large connection, then the 7 standard connection messages all utilize one cached connection. The 3 large connection messages all utilize another cached connection. In total, the 10 messages use 2 cached connections.

Unconnected buffers

To establish a connection or process unconnected messages, the controller uses an unconnected buffer.

Information on unconnected buffers applies only to CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers.

Term	Definition
Unconnected buffer	An allocation of memory that the controller uses to process unconnected communication. The controller performs unconnected communication when it: • Establishes a connection with a device, including an I/O module. • Executes a MSG instruction that does not use a connection.
	The controller can have 10 to 40 unconnected buffers. • The default number is 10. • To increase the number of unconnected buffers, execute a MSG instruction that reconfigures the number of unconnected buffers. • Each unconnected buffer uses 1.2 KB of memory. • If all unconnected buffers are in use when an instruction leaves the message queue, an error occurs and data does not transfer.

If a MSG instruction uses a connection, the instruction uses an unconnected buffer when it first executes to establish a connection. If you configure the instruction to cache the connection, it no longer requires an unconnected buffer once the connection is established.

Guidelines

As you plan and program your MSG instructions, follow these guidelines.

Guideline	Details

When possible, program read messages instead of write messages.	Read messages support simpler troubleshooting. They execute in the same controller so the information flow configuration is visible. However, write messages change data in another controller. Determining the source of the data change is dependent on code commenting quality.
For each MSG instruction, create a control tag.	• Data type = MESSAGE • Scope = controller • The tag cannot be part of an array or a user-defined data type.
Keep the source and destination data at the controller scope.	A MSG instruction can access only tags that are in the **Controller Tags** folder (controller scope) for CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers. In versions 31.00 and later, A MSG instruction can access tags that are in the **Controller Tags** folder (controller scope) or a Program Local scope tag (program scope) for CompactLogix 5380, ControlLogix 5580 and 5590, Compact GuardLogix 5380, and GuardLogix 5580 controllers. **Tip:** Tags referenced in the remote controller must be controller scoped.
If your message is to a device that uses 16-bit integers, such as a PLC-5 or SLC 500 controller, and it transfers integers (not REALs), use a buffer of INTs in the message and DINTs throughout the project.	Logix 5000 controllers execute more efficiently and use less memory when working with 32-bit integers (DINTs). See Convert between INTs and DINTs on page 16.
Cache the connection for those MSG instructions that execute most frequently, up to the maximum number permissible for your controller revision.	Execution time is optimized when the controller does not open a connection each time the message executes.
If you want to enable more than 16 MSGs at one time, use a management strategy to ensure some MSG instructions are not delayed entering the queue.	To guarantee the execution of each message, use one of these options: • Enable each message in sequence. • Enable the messages in groups. • Program a message to communicate with multiple devices. • Program logic to coordinate the execution of messages.

Keep the number of unconnected and uncached MSGs less than the number of unconnected buffers.	The controller can have 10 to 40 unconnected buffers. The default number is 10. • If all unconnected buffers are in use when an instruction leaves the message queue, an error occurs, the data is not transferred. • You can increase the number of unconnected buffers (up to 40), provided you continue to adhere to the previous guideline. • To increase the number of unconnected buffers, see "Get or set the number of unconnected buffers on page 13".

Get or set the number of unconnected buffers

To determine or change the number of unconnected buffers, use an MSG instruction.

- The range is 10 to 40 unconnected buffers.
- The default number is 10.
- Each unconnected buffers uses 1.2 KB of memory.

NOTE: Information on unconnected buffers applies only to CompactLogix 5370, ControlLogix 5570, Compact GuardLogix 5370, and GuardLogix 5570 controllers.

Get the number of unconnected buffers

To determine the number of unconnected buffers that are currently available, configure a Message (MSG) instruction as follows.

On this tab	For this item	Type or choose		
Configuration	Message Type	CIP Generic		
	Service Type	Custom		
	Service Code	3		
	Class	304		
	Instance	1		
	Attribute	0		
	Source Element	*source_array* where data type = SINT[4]		
			In this element	Enter
			source_array[0]	1
			source_array[1]	0

			source_array[2]	17
			source_array[3]	0
	Source Length (bytes)	4 (Write 4 SINTs.)		
	Destination Element	*destination_array* where data type = SINT[10] (Leave all values = 0.)		
		destination_array[6] = current number of unconnected buffers		
Communication	Path	THIS or for earlier Logix5000 controllers: 1, *slot_number_of_controller*		

Set the number of unconnected buffers

As a starting value, set the number of unconnected buffers equal to the number of unconnected and uncached messages enabled at one time plus 5. The additional 5 buffers provide a cushion in case you underestimate the number of messages that are enabled at once.

To change the number of unconnected buffers of the controller, configure a Message (MSG) instruction as follows.

On this tab	For this item	Type or select		
Configuration	Message Type	CIP Generic		
	Service Type	Custom		
	Service Code	4		
	Class	304		
	Instance	1		
	Attribute	0		
	Source Element	*source_array* where data type = SINT[8]		
			In this element	Enter
			source_array[0]	1
			source_array[1]	0
			source_array[2]	17
			source_array[3]	0
			source_array[4]	Number of unconnected buffers that you want.
			source_array[5]	0

			source_array[6]	0
			source_array[7]	0
	Source Length (bytes)	8 (Write 8 SINTs.)		
	Destination Element	*destination_array* where data type = SINT[6] (Leave all the values = 0.)		
Communication	Path	THIS or for earlier Logix 5000 controllers: 1, *slot_number_of_controller*		

Set the number of unconnected buffers

If S:FS = 1 (first scan)

then set the number of unconnected buffers for the controller.

Source_Array[0] = 1

Source_Array[0] = 1

Source_Array[1] = 0

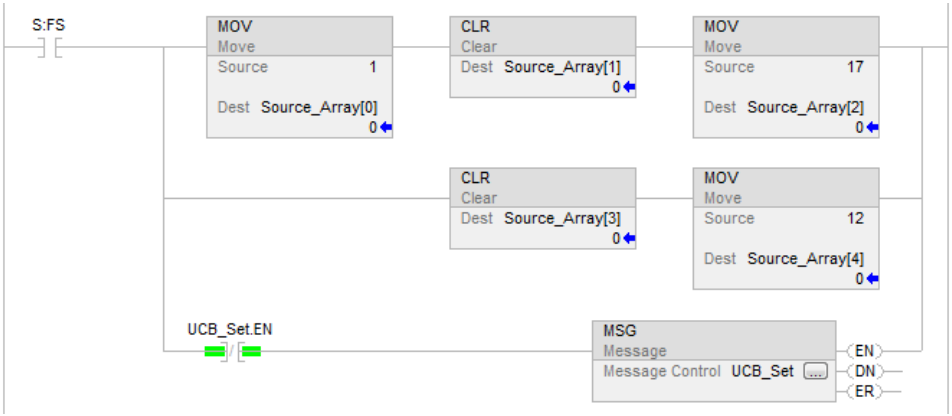
Source_Array[2] = 17

Source_Array[3] = 0

Source_Array[4] = 12 (The number of unconnected buffers that you want. In this example, we want 12 buffers.)

If UCB_Set.EN = 0 (MSG instruction is not already enabled)

then MSG instruction sets the number of unconnected buffers = Source_Array[4].



Tag Name	Type	Description
UCB_Set	MESSAGE	Control tag for the MSG instruction.
Source_Array	SINT[8]	Source values for the MSG instruction, including the number of unconnected buffers that you want.

Convert between INTs and DINTs

In the Logix 5000 controller, use the DINT data type for integers whenever possible. Logix 5000 controllers execute more efficiently and use less memory when working with 32-bit integers (DINTs).

If your message is to a device that uses 16-bit integers, such as a PLC-5 or SLC 500 controller, and it transfers integers (not REALs), use a buffer of INTs in the message and DINTs throughout the project. This increases the efficiency of your project.

Read 16-bit integers

	❶		❷	
Data from the device		**Buffer of INTs**		**DINTs for use in the project**
Word 1	→	INT_Buffer[0]	→	DINT_Array[0]
Word 2	→	INT_Buffer[1]	→	DINT_Array[1]
Word 3	→	INT_Buffer[2]	→	DINT_Array[2]

Description	
❶	The Message (MSG) instruction reads 16-bit integers (INTs) from the device and stores them in a temporary array of INTs.
❷	A File Arith/Logical (FAL) instruction converts the INTs to DINTs for use by other instructions in your project.

Write 16-bit integers

	❶		❷	
DINTs from the project		**Buffer of INTs**		**Data for the device**
DINT_Array[0]	→	INT_Buffer[0]	→	Word 1
DINT_Array[1]	→	INT_Buffer[1]	→	Word 2
DINT_Array[2]	→	INT_Buffer[2]	→	Word 3

	Description
1	An FAL instruction converts the DINTs from the Logix 5000 controller to INTs.
2	The MSG instruction writes the INTs from the temporary array to the device.

Read integer values from a PLC-5 controller

If Condition_1 = 1

and Msg_1.EN = 0 (MSG instruction is not enabled)

then read 3 integers from the PLC-5 controller and store them in INT_Buffer (3 INTs).

NOTE: If Msg_1.DN =1 (MSG instruction has read the data)

then reset the FAL instruction.

The FAL instruction sets DINT_Array = INT_Buffer. This converts the values to 32-bit integers (DINTs).

Write integer values to a PLC-5 controller

If Condition_2 = 1

then reset the FAL instruction.

The FAL instruction sets INT_Buffer = DINT_Array. This converts the values to 16-bit integers (INTs).

NOTE: If Control_2.DN = 1 (FAL instruction has converted the DINTs to INTs)

and Msg_2.EN = 0 (MSG instruction is not enabled)

then write the integers in INT_Buffer (3 INTs) to the PLC-5 controller.

Manage multiple messages

Introduction

You can use ladder logic to send groups of message (MSG) instructions in sequence.

- To be processed, each MSG instruction must enter the message queue.
- The queue holds 48 MSGs.
- If more than 16 MSGs are enabled at one time, the message throttle prevents some of the messages from entering the message queue. If this occurs, the MSG is held until room exists on the queue for the controller to process the MSG. On each subsequent scan of the MSG, it checks the queue to see if room exists.

The message manager logic lets you control the number of MSGs that are enabled at one time and enable subsequent MSGs in sequence. In this way, MSGs enter and exit the queue in order and do not need to wait for room on the queue to become available.

Message manager logic

The message manager logic sends three groups of MSGs. Use as many groups as needed to include all your MSGs.

The Msg_Group tag controls the enabling of each MSG.

- The tag uses the DINT data type.
- Each bit of the tag corresponds to a group of MSGs. For example, Msg_Group.0 enables and disables the first group of MSGs (group 0).

Message manner logic

To make the example easier to follow, each group contains only two MSGs. In your project, use more MSGs in each group, such as five.

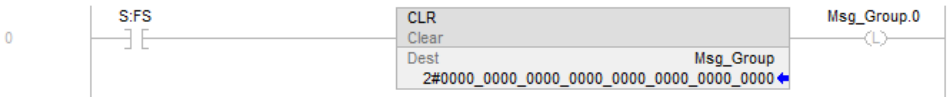
Initialize the logic

If S:FS = 1 (first scan)

then initialize the MSGs:

Msg_Group = 0, which disables all MSGs.

Msg_Group.0 =1, which enables the first group of MSGs.



NOTE: Restart the sequence

If the MSGs in group 2 (last group) are currently enabled (Msg_Group.2 = 1)

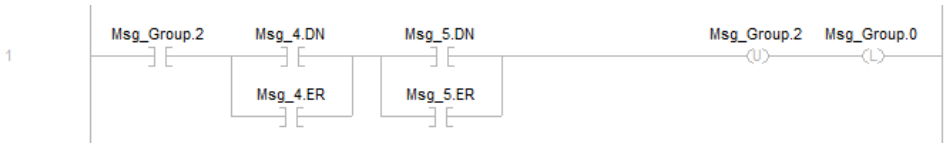
and Msg_4 is in the state of done or error

and Msg_5 is in the state of done or error

then restart the sequence of MSGs with the first group:

Msg_Group.2 = 0. This disables the last group of MSGs.

Msg_Group.0 = 1. This enables the first group of MSGs.



NOTE: Send the first group of MSGs

If Msg_Group.0 changes from 0 -> 1 then

send Msg_0.

send Msg_1.

Because a MSG instruction is a transitional instruction, it executes only when its rung-condition-in changes from false to true.

NOTE: Enable the second group of MSGs

If the MSGs in group 0 are currently enabled (Msg_Group.0 = 1)

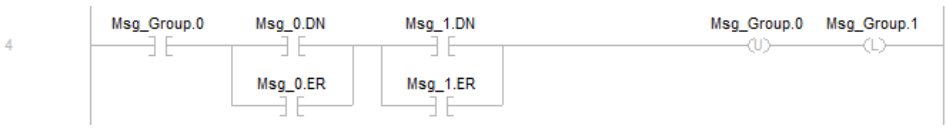
and Msg_0 is in the state of done or error

and Msg_1 is in the state of done or error

then:

Msg_Group.0 = 0. This disables the current group of MSGs.

Msg_Group.1 = 1. This enables the next group of MSGs.



NOTE: Send the second group of MSGs

If Msg_Group.1 changes from 0 -> 1 then

send Msg_2.

send Msg_3.

NOTE: Enable the next group of MSGs

If the MSGs in group 1 are currently enabled (Msg_Group.1 = 1)

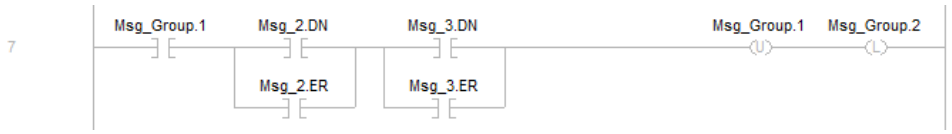
and Msg_2 is in the state of done or error

and Msg_3 is in the state of done or error

then:

Msg_Group.1 = 0. This disables the current group of MSGs.

Msg_Group.2 = 1. This enables the next group of MSGs.



NOTE: Send the next group of MSGs

If Msg_Group.1 changes from 0 -> 1 then

send Msg_2.

send Msg_3.

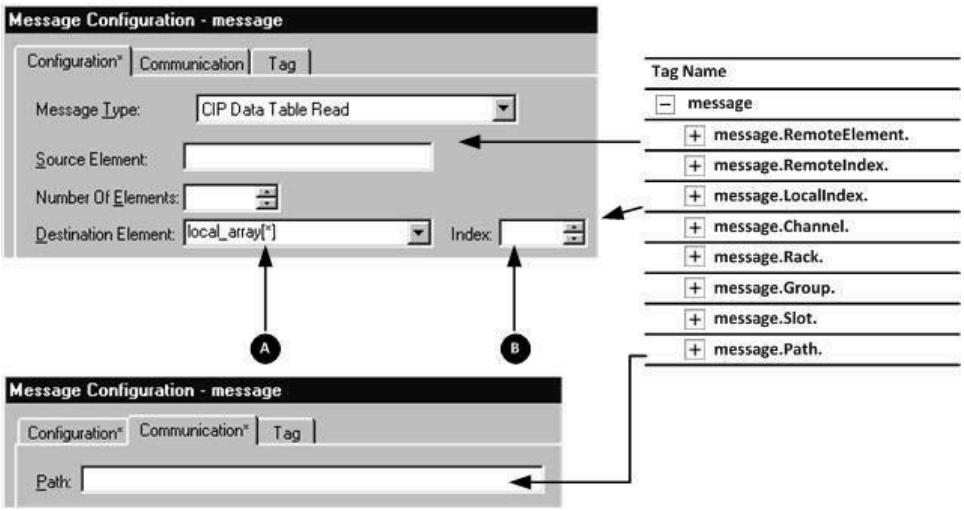
Send a message to multiple controllers

Introduction

You can program one message instruction to communicate with multiple controllers. To reconfigure a MSG instruction during runtime, write new values to the members of the MESSAGE data type.

IMPORTANT: In the MESSAGE data type, the RemoteElement member stores the tag name or address of the data in the controller that receives the message.

	If the message	Then the RemoteElement is the	
	Reads data	Source element	
	Writes data	Destination element	



A	If you use an asterisk [*] to designate the element number of the array, the value in B provides the element number.
B	The **Index** box is available only when you use an asterisk [*] in the Source Element or Destination Element. The instruction substitutes the value of Index for the asterisk [*].

To copy the previous components from a sample project, take the following steps.

a. On the **Help** menu, click **Quick Start**.

b. On the **Quick Start** window, in the left navigation pane, expand **Controller Projects**, and click **Open Sample Project**.

In the **Open Project** dialog box, click **MSG_To_multiple_Controllers.acd**, and click **Open**.

Configure the I/O configuration

Although not required, it is recommended that you add the communication modules and remote controllers to the I/O configuration of the controller. This makes it easier to define the path to each remote controller.

For example, once you add the local communication module, the remote communication module, and the destination controller, clicking Browse lets you select the destination.

Message Path Browser

Path: peer_controller

- peer_controller
  - I/O Configuration
    - [0] 1756-CNB/x Local_CNB
      - 2 [0] 1756-CNB/x chassis_b
        - [1] 1756-L55/x peer_controller ←

Define your source and destination elements

An array stores the data that is read from or written to each remote controller. Each element in the array corresponds to another remote controller.

1. Use a worksheet similar to this one to organize the tag names in the local and remote controllers.

Name of Remote Controller	Tag or Address of Data in Remote Controller	Tag in This Controller
		local_array[0]
		local_array[1]
		local_array[2]
		local_array[3]

2. Create the local_array tag, which stores the data in this controller.

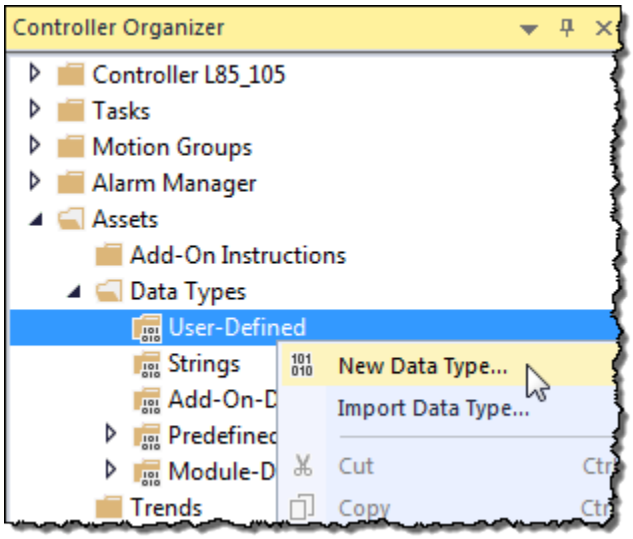
Tag Name	Type
local_array	*data_type [length]* where: *data_type* is the data type of the data that the message sends or receives, such as DINT, REAL, or STRING. *length*

Create the MESSAGE_ CONFIGURATION data type

Create a user-defined data type to store the configuration variables for the message to each controller.

- Some of the required members of the data type use a string data type.
- The default STRING data type stores 82 characters.
- If your paths or remote tag names or addresses use less than 82 characters, you have the option of creating a new string type that stores fewer characters. This lets you conserve memory.
- To create a string type, click **File > New Component > String Type**.
- If you create a string type, use it in place of the STRING data type.

To store the configuration variables for the message to each controller, expand the **Assets > Data Types** folder, right-click **User Defined**, and select **New Data Type** to create the following user-defined data type.

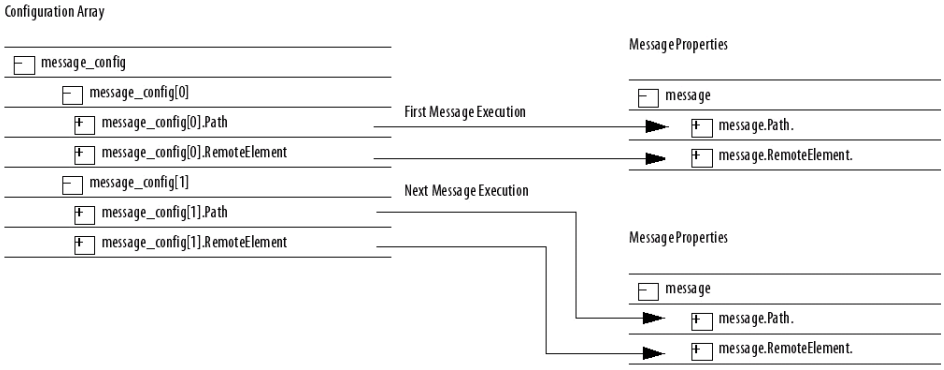


Data Type: MESSAGE_CONFIGURATION
Name: MESSAGE_CONFIGURATION

Description: Configuration properties for a message to another controller			
Members			
Name	Data Type	Style	Description
Path	STRING		
RemoteElement	STRING		

Create the configuration array

Store the configuration properties for each controller in an array. Before each execution of the MSG instruction, your logic loads new properties into the instruction. This sends the message to another controller.

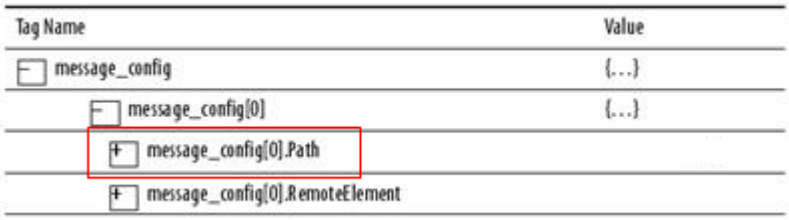


1. To store the configuration properties for the message, create the following array.

Tag Name	Type	Scope
message_config	MESSAGE_CONFIGURATION[*number*][1]	Any

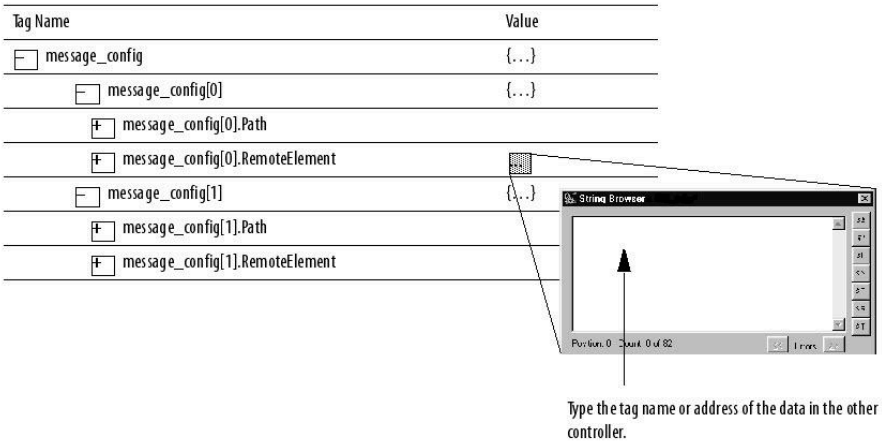
[3]

2. In the message_config array, enter the path to the first controller that receives the message.



3. Number indicates the number of controllers to send the message

3. In the message_config array, enter the tag name or address of the data in the first controller to receive the message.



4. Enter the path and remote element for each additional controller.

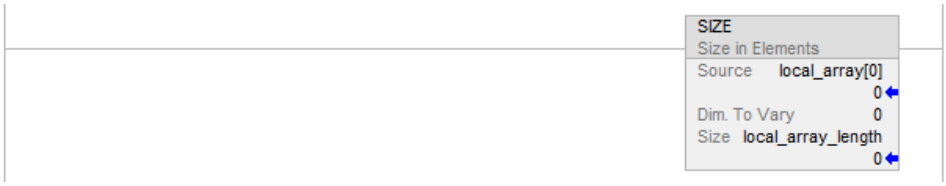
Tag Name			Value
message_config			{...}
—	message_config[0]		{...}
		— message_config[0].Path	
		— message_config[0].RemoteElement	
	— message_config[1]		{...}
		— message_config[1].Path	←
		— message_config[1].RemoteElement	←

Get the size of the local array

The SIZE instruction:

- Counts the number of elements in local_array.
- Counts the number of elements in Dimension 0 of the array. In this case, that is the only dimension.

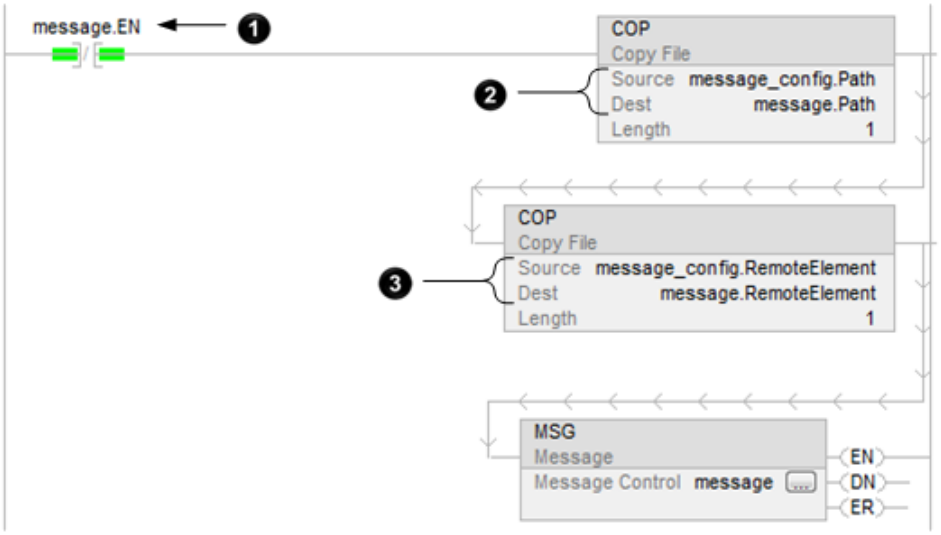
Local_array_length (DINT) stores the size (number of elements) of local_array. This value tells a subsequent rung when the message is sent to all controllers and to start with the first controller again.



Load the message properties for a controller

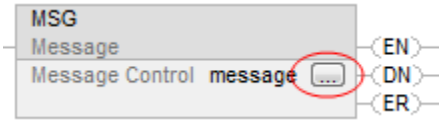
To load the message properties for a controller:

1. The XIO instruction conditions the rung to continuously send the message.
2. The first COP instruction loads the path for the message. The value of index determines which element the instruction loads from message_config. The instruction loads one element from message_config.
3. The second COP instruction loads the tag name or address of the data in the controller that receives the message. The value of index determines which element the instruction loads from message_config. The instruction loads one element from message_config.



Configure the message

The following table explains how to configure the message.



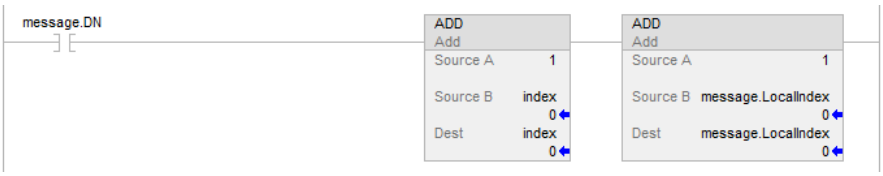
On this tab	If you want to	For this item	Type or select
Configuration	Read (receive) data from the other controllers	Message Type	The read-type that corresponds to the other controllers

		Source Element	Tag or address that contains the data in the first controller
		Number Of Elements	1
		Destination Element	local_array[*]
		Index	0
	Write (send) data to the other controllers	Message Type	The write-type that corresponds to other controllers
		Source Element	local_array[*]
		Index	0
		Number Of Elements	1
		Destination Element	Tag or address that contains the data in the first controller
Communication	–	Path	Path to the first controller
		Cache Connections	Clear the **Cache Connections** check box (more efficient since this procedure continuously changes the path of the message)

Step to the next controller

After the MSG instruction sends the message, the following actions occur.

1. The first ADD instruction increments the index. This lets the logic load the configuration properties for the next controller into the MSG instruction.
2. The second ADD instruction increments the LocalIndex member of the MSG instruction. This lets the logic load the value from the next controller into the next element of local_array.



Rockwell Automation Support

Use these resources to access support information.

Technical Support Center	Find help with how-to videos, FAQs, chat, user forums, and product notification updates.	rok.auto/support
Local Technical Support Phone Numbers	Locate the telephone number for your country.	rok.auto/phonesupport
Technical Documentation Center	Quickly access and download technical specifications, installation instructions, and user manuals.	rok.auto/techdocs
Literature Library	Find installation instructions, manuals, brochures, and technical data publications.	rok.auto/literature
Product Compatibility and Download Center (PCDC)	Get help determining how products interact, check features and capabilities, and find associated firmware.	rok.auto/pcdc

Documentation Feedback

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

Waste Electrical and Electronic Equipment (WEEE)

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

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

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

Connect with us.

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

AMERICAS: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000
EUROPE/MIDDLE EAST/AFRICA: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2663 0600
ASIA PACIFIC: Rockwell Automation SEA Pte Ltd, 2 Corporation Road, #04-05, Main Lobby, Corporation Place, Singapore 618494, Tel: (65) 6510 6608
UNITED KINGDOM: Rockwell Automation Ltd., Pitfield, Kiln Farm, Milton Keynes, MK11 3DR, United Kingdom, Tel: (44)(1908) 838-800

Publication 1756-PM012L-EN-P - September 2025
Supersedes Publication 1756-PM012K-EN-P - September 2024