Quick Start

Allen-Bradley

Micro800 Programmable Controllers: Getting Started with CIP Client Messaging

Catalog Numbers Bulletin 2080-LC20, 2080-LC30, 2080-LC50

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.

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

About This Publication

This publication is designed to provide quickstart instructions for using CIP GENERIC and CIP Symbolic Messaging in Micro820™, Micro830®, and Micro850® programmable logic controllers (PLC). It makes use of sample programs to illustrate the basic steps that a user needs to perform to use the CIP messaging functions in Micro820, Micro830, and Micro850 controllers.

To assist in the design and installation of your system, refer to the:

- Micro820 20-pt Programmable Controllers User Manual publication 2080-UM005.
- Micro830 and Micro850 Programmable Controllers User Manual publication 2080-UM002.

The beginning of each chapter contains the following information. Read these sections carefully before beginning work in each chapter.

- **Before You Begin** – This section lists the steps that must be completed and decisions that must be made before starting that chapter. The chapters in this quick start do not have to be completed in the order in which they appear, but this section defines the minimum amount of preparation required before completing the current chapter.
- **What You Need** – This section lists the tools that are required to complete the steps in the current chapter. This includes, but is not limited to, hardware and software.
- **Follow These Steps** – This illustrates the steps in the current chapter and identifies which steps are required to complete the examples using specific networks.

Audience

To be able to use the CIP messaging feature effectively, you need to be familiar with programming in function block diagram, structured text, and ladder programming.

This quick start works hand-in-hand with:

- Micro820 20-pt Programmable Controllers User Manual publication 2080-UM005.
- Micro830 and Micro850 Programmable Controllers User Manual publication 2080-UM002.

Required Software

To complete this quick start, the following software is required:

- **Connected Components Workbench™ revision 4 and later**

  Connected Components Workbench is the main programming software for Micro800 systems. It provides a choice of IEC 61131-3 programming languages (ladder diagram, function block diagram, structured text) with user defined function block support that optimizes machine control.

  You will need the Connected Components Workbench software to write your function block programs, execute your function blocks, and see results.

Additional Resources

Resource	Description
Micro820 20-pt Programmable Controllers User Manual, publication 2080-UM005	A detailed description of how to install and use your Micro820 programmable controller and expansion I/O system.
Micro830 and Micro850 Programmable Controllers User Manual, publication 2080-UM002	A detailed description of how to install and use your Micro830 and Micro850 programmable controller and expansion I/O system.
Micro800 Programmable Controllers General Instructions, publication 2080-RM001	Information on instruction sets for developing programs for use in Micro800 control systems.
Micro800 Programmable Controller External AC Power Supply Installation Instructions, publication 2080-IN001	Information on wiring and installing the optional AC power supply.
Industrial Automation Wiring and Grounding Guidelines, publication 1770-4.1	More information on proper wiring and grounding techniques.
Connected Components Workbench Online Help	Online Help that provides a description of the different elements of the Connected Components Workbench software.

Appendix D

CIP Objects

Appendix E

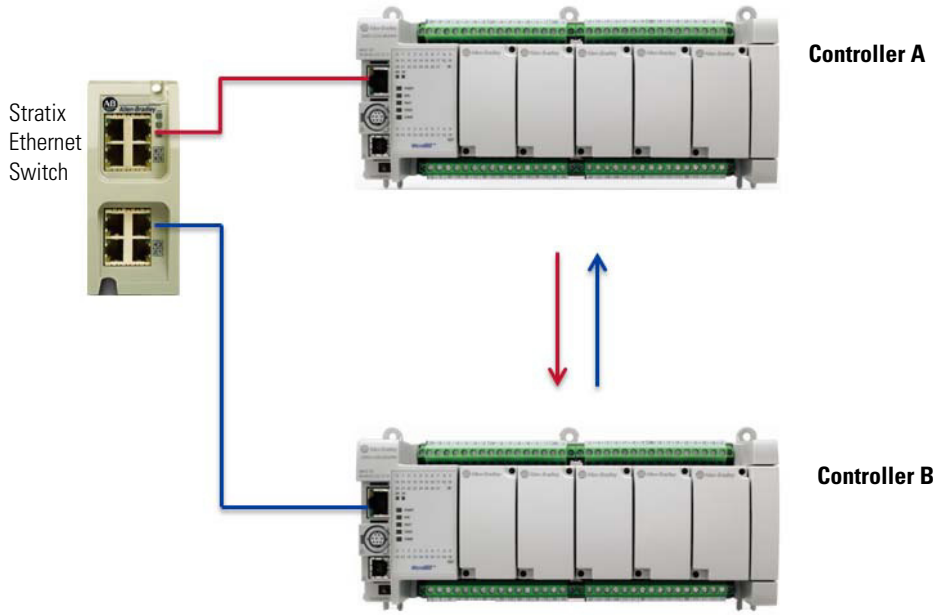
CIP Error Message Codes

Notes:

Overview

This quick start instructions illustrate how you can use the CIP Generic and CIP Symbolic Client Messaging functions on Micro820, Micro830, and Micro850 controllers. It includes two sample projects that provide step-by-step instructions on how to program using the CIP messaging function blocks and assign values to the parameters to make a simple query/command from a Micro850 Controller A to a Micro850 Controller B using both messaging features.

To learn more about the MSG_CIPGeneric and MSG_CIPSymbolic function blocks and their corresponding input and output parameters, see the Connected Components Workbench Online Help and MSG_CIPSYMBOLIC Function Block on page 35 and MSG_CIPGENERIC Function Block on page 39.



IMPORTANT To use the Micro800 CIP client messaging feature effectively, users need to have a basic understanding of the following:

- MSG_CIPGeneric and MSG_CIPSymbolic Function Blocks
  See MSG_CIPSYMBOLIC Function Block on page 35 and MSG_CIPGENERIC Function Block on page 39.
- Programming and working with elements in the Connected Components Workbench software
  You need to have a working knowledge of ladder diagram, structured text, or function block diagram programming to be able to work with function blocks, and input/output parameters. For tips and recommendations on programming CIP messaging instructions, see Programming Tips and Recommendations for Micro800 CIP Messaging Instructions (MSG) on page 67.

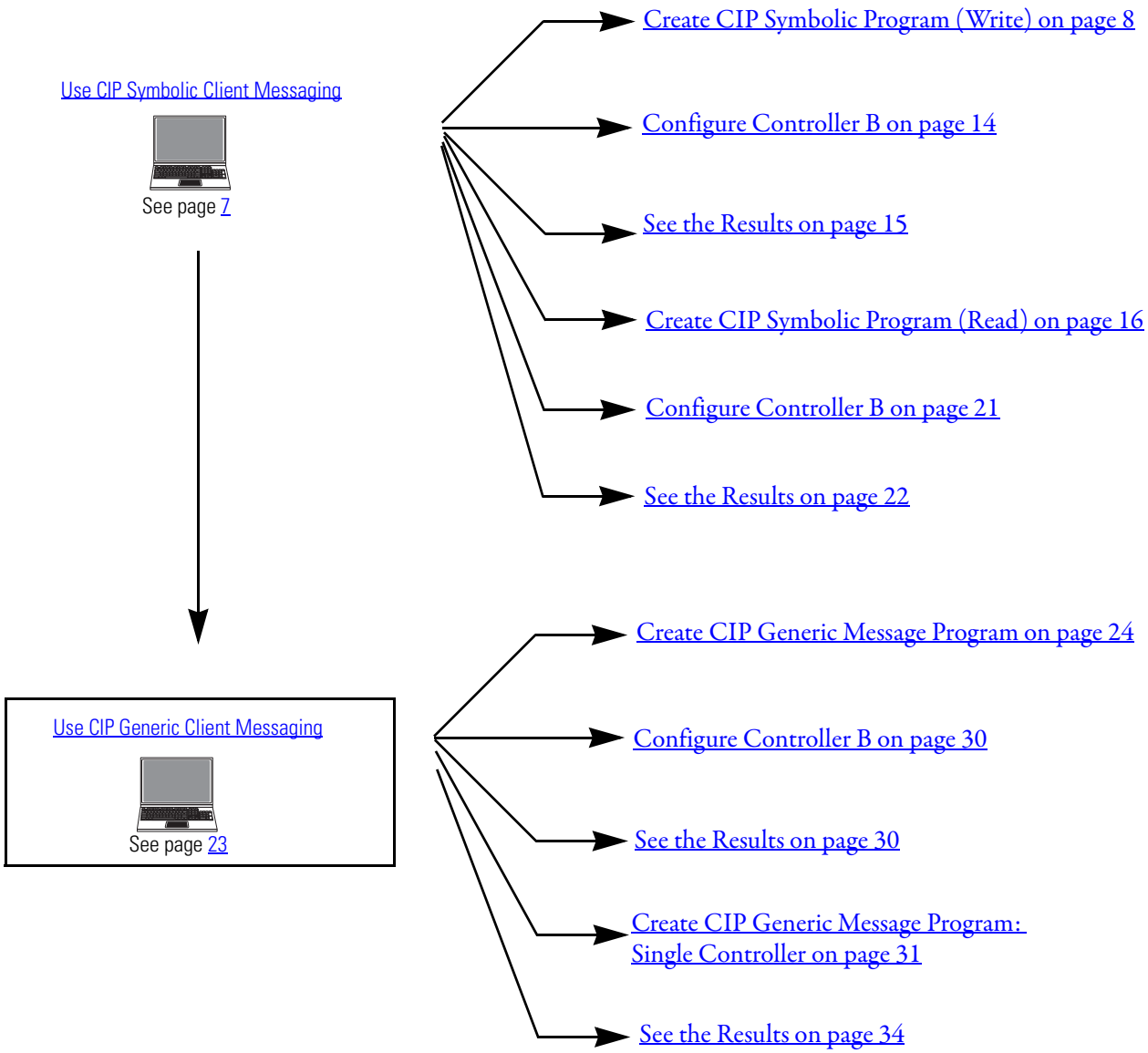
Hardware and Software Compatibility

The CIP Generic and CIP Symbolic Messaging features on Micro830 and Micro850 controllers are compatible with Connected Components Workbench revision 4 and firmware revision 4 and later. The CIP Generic and CIP Symbolic Messaging features on Micro820 controllers are compatible with Connected Components Workbench revision 6 and firmware revision 6 and later.

ATTENTION: To learn more about Connected Components Workbench and detailed descriptions of the variables for the MSG Function Blocks, refer to Connected Components Workbench Online Help that comes with your Connected Components Workbench installation.

Follow These Steps

The major subsections for this quick start project are outlined in the following flowchart. Follow the steps under each subsection to become familiar with the required procedure to configure your controller, set up a sample project, and create programs for CIP messaging.



TIP **Download the Sample Code**

You can download the sample code (ID: 90904) for the programs used in this Quick Start from the following link: http://www.rockwellautomation.com/go/scmicro800

Create a Micro800 Project

Introduction

In this chapter, you will create a Micro850 controller project through the Connected Components Workbench software.

Before You Begin

Ensure that you have properly installed revision 4 or later of the Connected Components Workbench software.

What You Need

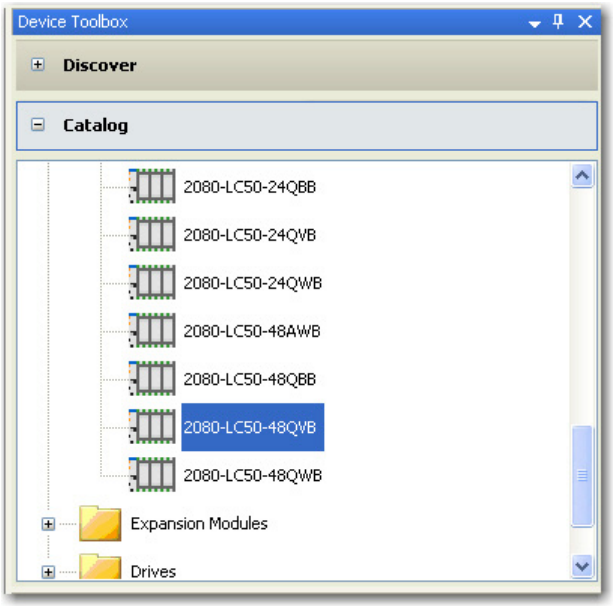
- Connected Components Workbench software revision 4 or later
- Firmware revision 4 or later

Create a Micro800 Project in Connected Components Workbench

Launch the Connected Components Workbench software.

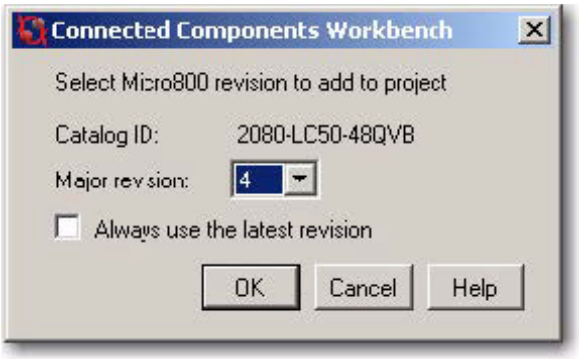
1. On the Device Toolbox, expand the list of Controllers by clicking the + sign.

   Be sure to select the catalog number that matches your Micro850 controller.

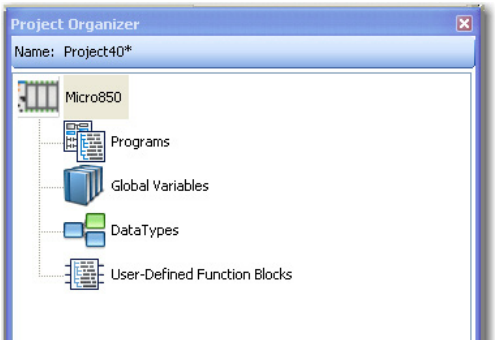


TIP If your controller is online, use the Discover feature to automatically discover your controller.

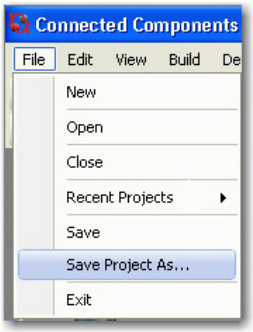
2. Select major revision 4 when manually adding a Micro850 controller.



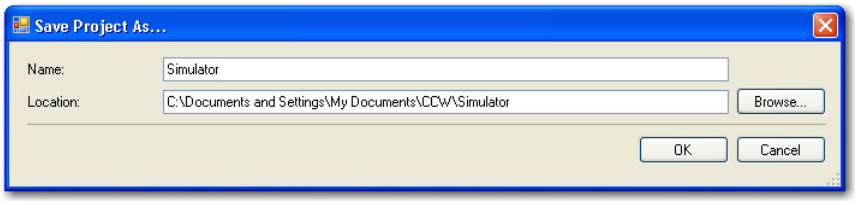
3. Drag your controller onto the Project Organizer pane.



4. Go to File → Save Project As.



5. Provide a project name for your project. Click OK.



TIP Any Micro820/Micro830/Micro850 controller can be used to follow the quickstart instructions described in this publication as long as it meets the software and firmware compatibility requirements for the Micro800 CIP Messaging feature. See Hardware and Software Compatibility on page 1.

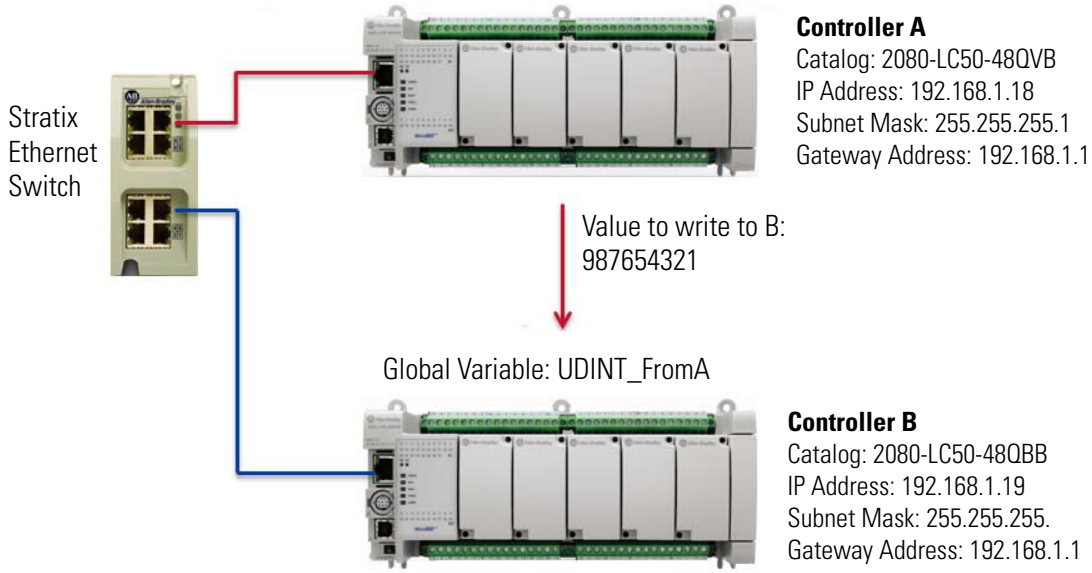
Notes:

Use CIP Symbolic Client Messaging

Introduction

Micro820, Micro830, and Micro850 controllers support CIP Symbolic client messaging through the MSG_CIPSYMBOLIC function block. This feature enables the sending of CIP Symbolic messages through Ethernet or serial cable.

In this chapter, you will learn how to use the function block to send a write instruction from Controller A to Controller B. Specifically, it shows you how to create a program that will write a value to Controller B's global variable, UDINT_FromA.



Before You Begin

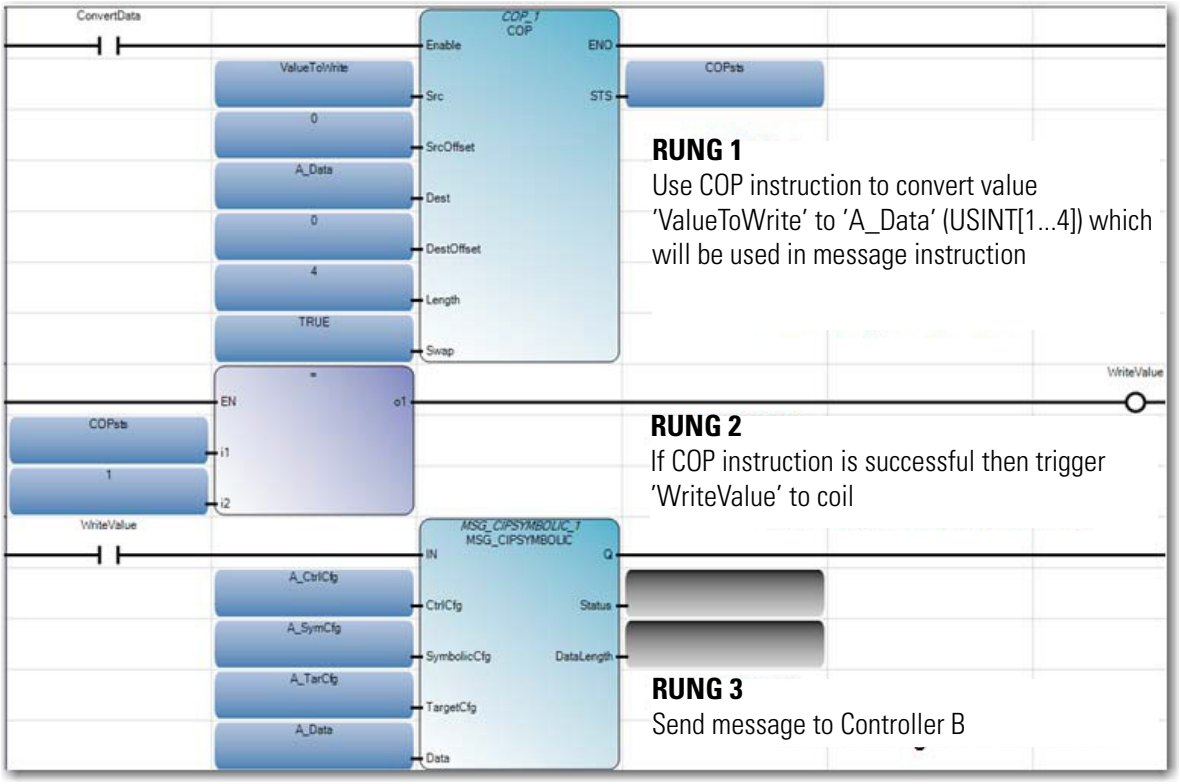
Ensure that you have properly installed revision 4 or later of the Connected Components Workbench software.

What You Need

- Connected Components Workbench software revision 4 or later
- Firmware revision 4 or later

Create CIP Symbolic Program (Write)

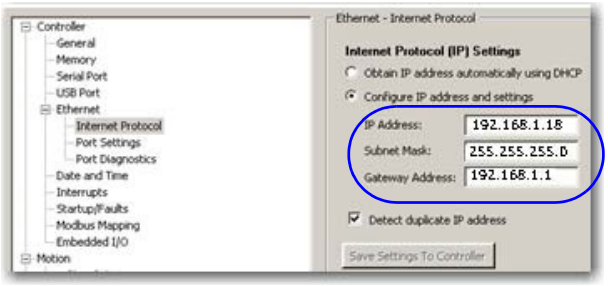
At the end of these instructions, you should have created the following program in Connected Components Workbench in your Controller A project.



Build the Code

1. Open the project you created for Controller A in the last chapter through Connected Components Workbench.

2. Configure the IP address, subnet mask and gateway of Controller A as shown.



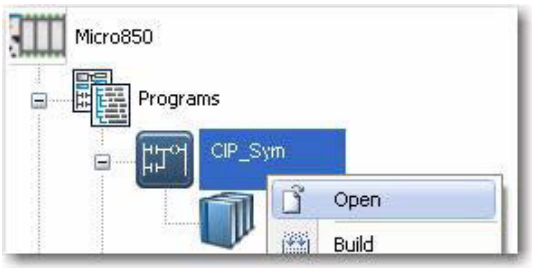
IP Address:
192.168.1.18

Subnet Mask:
255.255.255.0

Gateway Address:
192.168.1.1

3. On the Project Organizer pane, right-click Programs, and then select Add New LD: Ladder Diagram.
   You can press F2 to rename the program. Press Enter.

4. Right-click CIP_Sym program, and then choose Open.



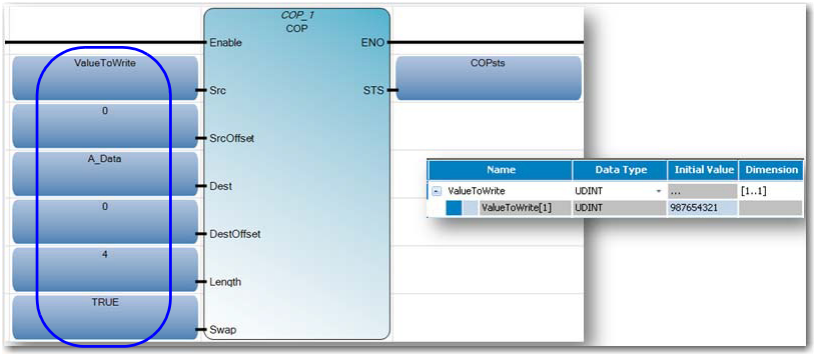
5. From the Toolbox, double-click Block to add it to the rung. Alternatively, you can drag and drop Block onto the rung.
   Your ladder rung should appear as shown.

6. Add COP instruction to the rung by typing COP on the Instruction Block Selector window that appears to filter the COP function block.

7. Choose COP. Then, click OK.

8. Create variables for the COP input parameters as shown.

   To learn more about the COP input and output parameter descriptions, see COP Function Block on page 43.

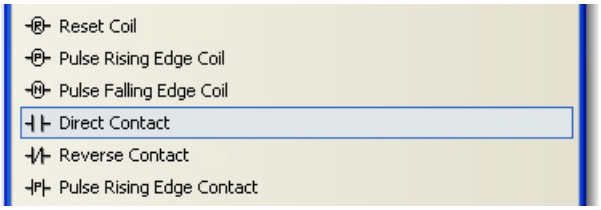


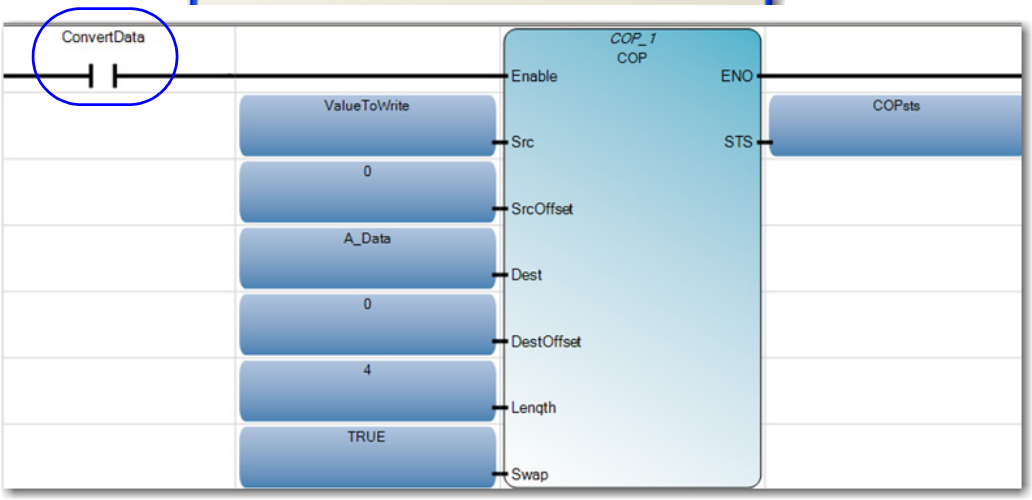
TIP You can disregard the yellow warning sign that appears. This indicates that the software expects a different data type.

COP Input Parameters

Variable	Data Type	Dimension	Init Value
ValueToWrite	UDINT	[1...1]	987654321
A_Data	USINT	[1...4]	
COPsts	UINT		
ConvertData	BOOL		
WriteValue	BOOL		

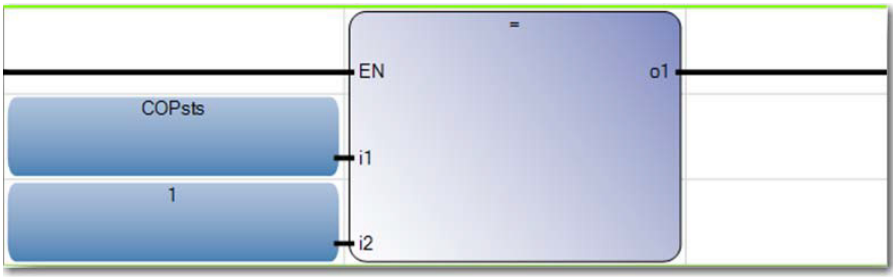
9. Add ConvertData contact to the rung by selecting Direct Contact from the Toolbox and drag it onto the rung as shown.



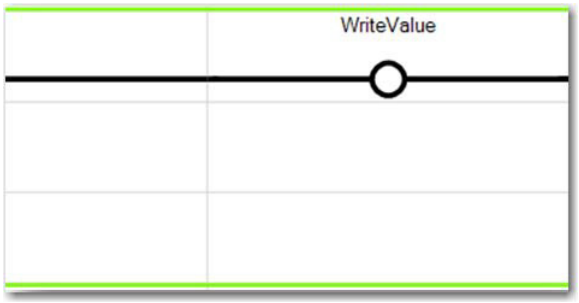


10. Create a second rung. From the Toolbox, select Rung and drag it onto the space just below the first rung.

11. Add = instruction block to the rung.

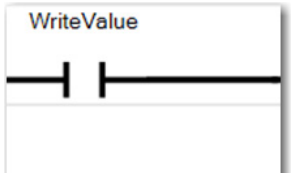


12. Add WriteValue coil to the end of the second rung by selecting Direct Coil from the Toolbox and add it to the rung as shown.

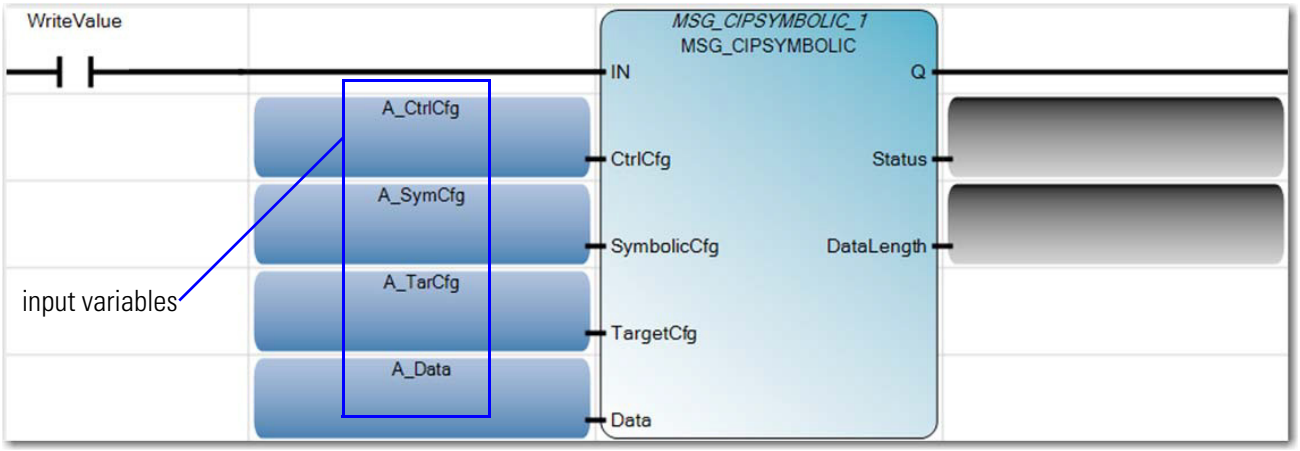


13. Create a third rung.

14. Add WriteValue contact to the rung as shown.

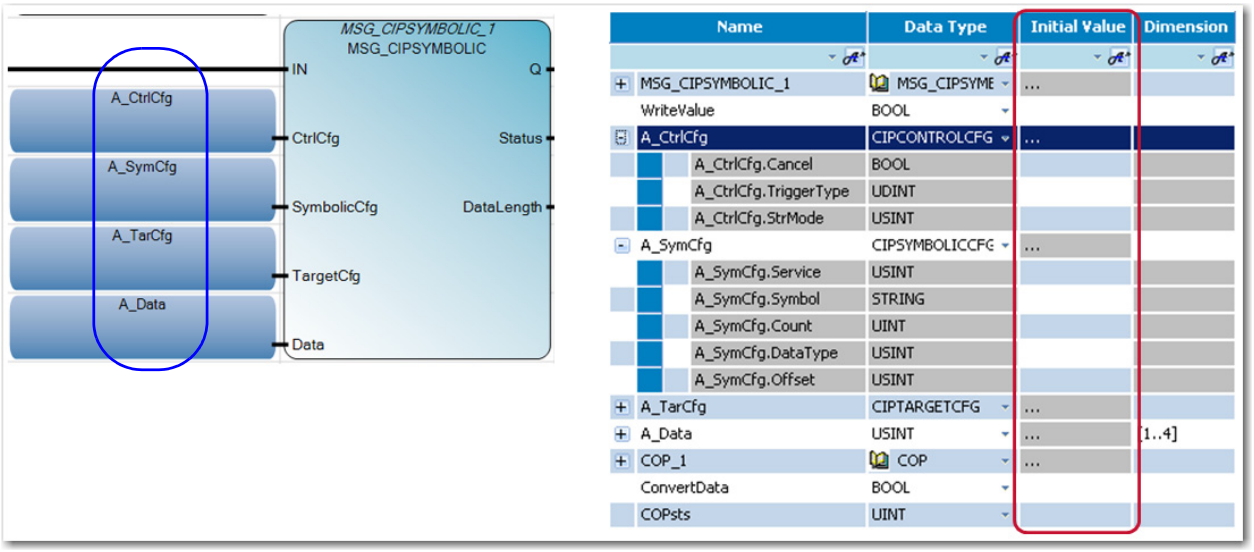


15. Add MSG_CIPSYMBOLIC function block to the rung and create the input variables as shown.



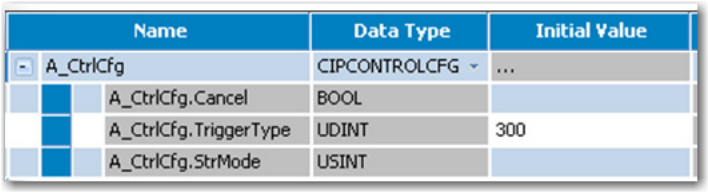
Assign Values to the MSG_CIPSYMBOLIC Input Variables

1. Set the values for the input variables through the Initial Value column of the Global Variables table.



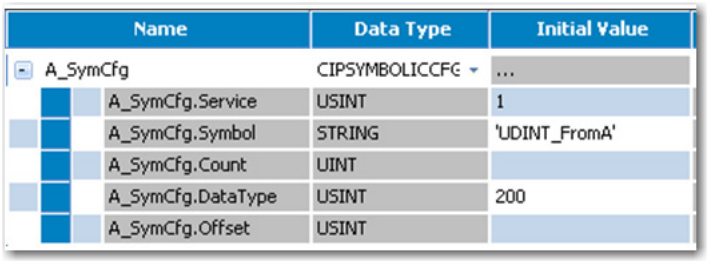
See MSG_CIPSYMBOLIC Function Block on page 35 for more information about MSG_CIPSYMBOLIC parameters.

2. Set the value for the A_CtrlCfg input variable as shown. See CtrlCfg on page 35 for parameter details and description.



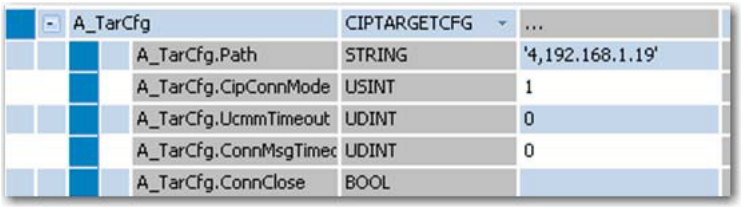
In this example, we write the value to controller B every 300 ms, and set initial value as 300. Set A_CtrlCfg.Cancel to False. By default, this parameter is set to False.

3. Set the value for the A_SymCfg input variable as shown.
   See SymbolicCfg on page 36 for parameter details and description.
   Leave the initial value of A_SymCfg.Count at 0. The value of 1 is used for count if this value is left at the default value of 0.



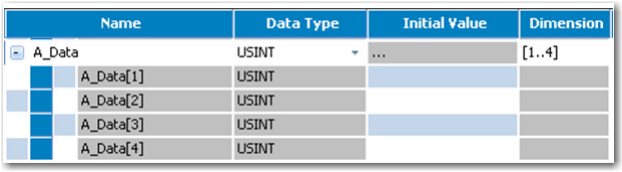
Name	Data Type	Initial Value
A_SymCfg	CIPSYMBOLICCFG	...
A_SymCfg.Service	USINT	1
A_SymCfg.Symbol	STRING	'UDINT_FromA'
A_SymCfg.Count	UINT	
A_SymCfg.DataType	USINT	200
A_SymCfg.Offset	USINT	

4. Set the value for the A_TarCfg input variable as shown.
   See TargetCfg on page 37 for parameter details and description.
   Set A_TarCfg.ConnClose to False. By default, this variable is set to false.



A_TarCfg	CIPTARGETCFG	...
A_TarCfg.Path	STRING	'4,192.168.1.19'
A_TarCfg.CipConnMode	USINT	1
A_TarCfg.UcmmTimeout	UDINT	0
A_TarCfg.ConnMsgTimec	UDINT	0
A_TarCfg.ConnClose	BOOL	

IMPORTANT The value for A_Data input variable is automatically obtained from the COP instruction in rung 1.



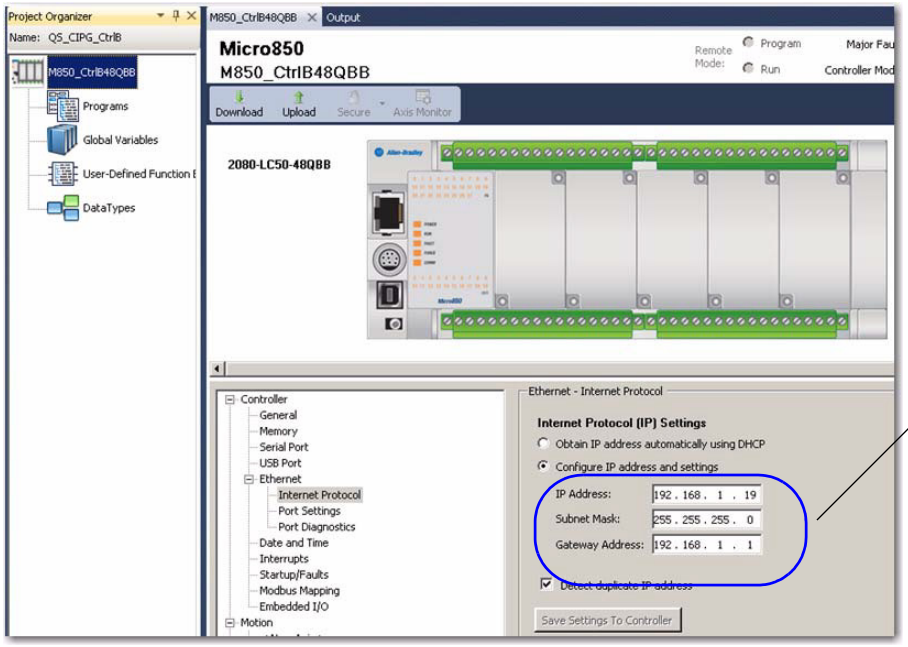
Name	Data Type	Initial Value	Dimension
A_Data	USINT	...	[1..4]
A_Data[1]	USINT		
A_Data[2]	USINT		
A_Data[3]	USINT		
A_Data[4]	USINT		

Note that USINT is 8-bit data (whereas A UDINT is 32-bit data), so the A_Data input variable is a one-dimension array with four elements.

5. Next, build and download the project to Controller A.

Configure Controller B

1. Configure Controller B as shown:

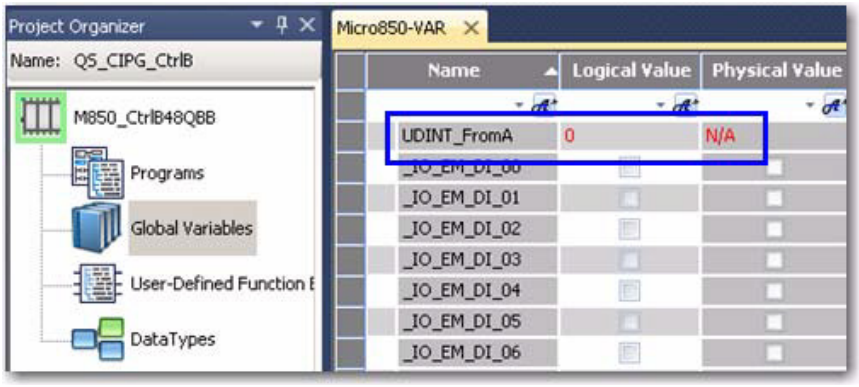


IP Address:
192.168.1.19

Subnet Mask:
255.255.255.0

Gateway Address:
192.168.1.1

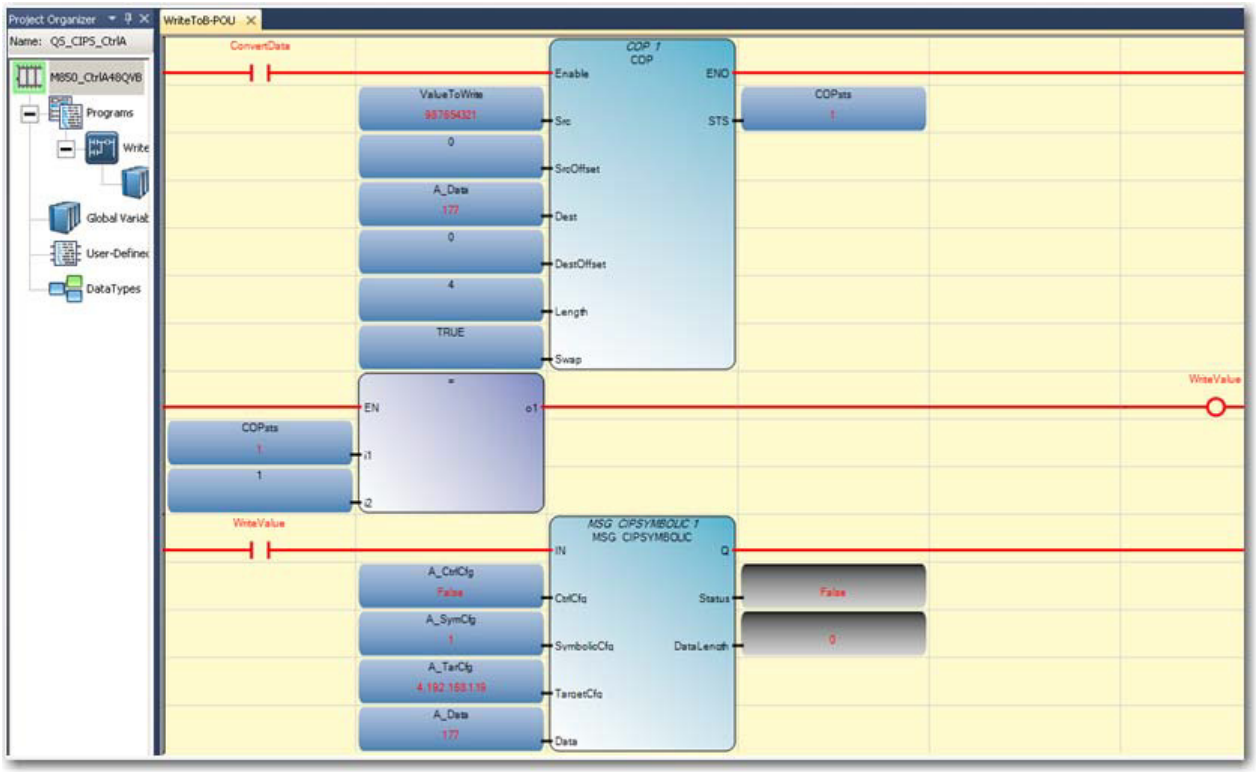
2. Create the Global Variable 'UDINT_FromA'.



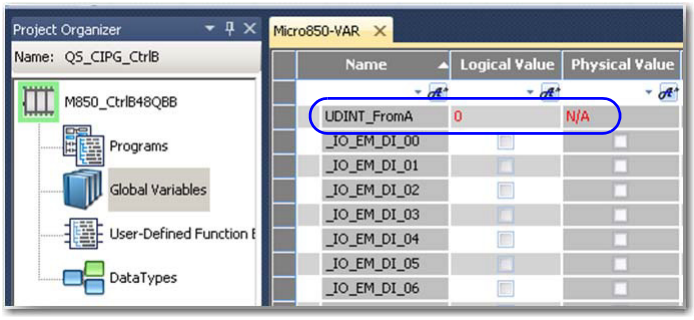
3. Next, build and download the project to Controller B.

See the Results

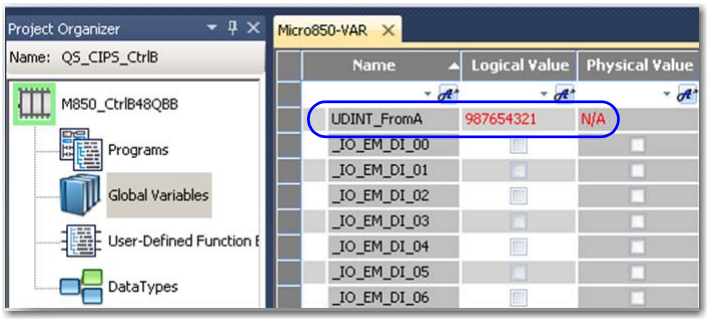
Controller A on execution of program



Controller B before message is sent by Controller A

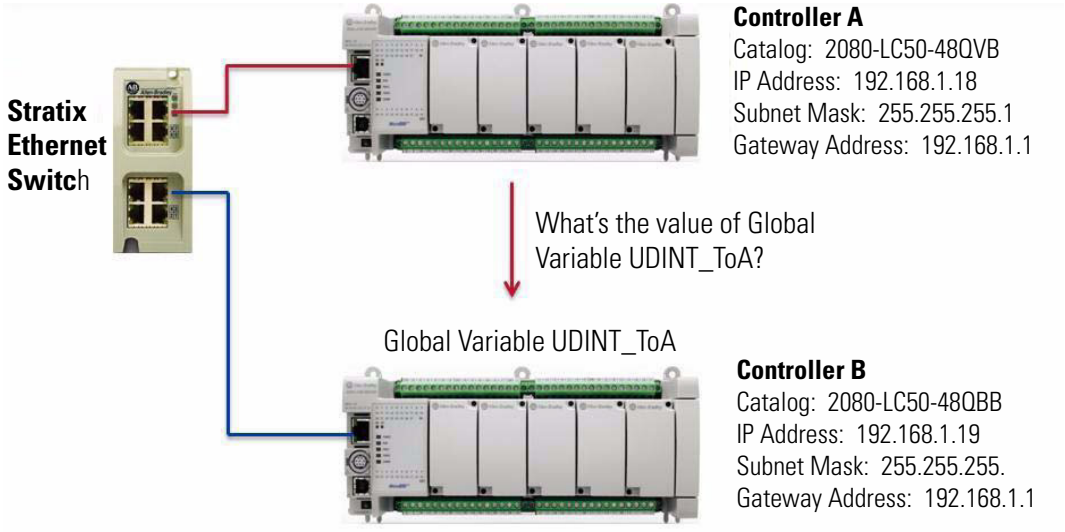


Controller B after message is sent by Controller A

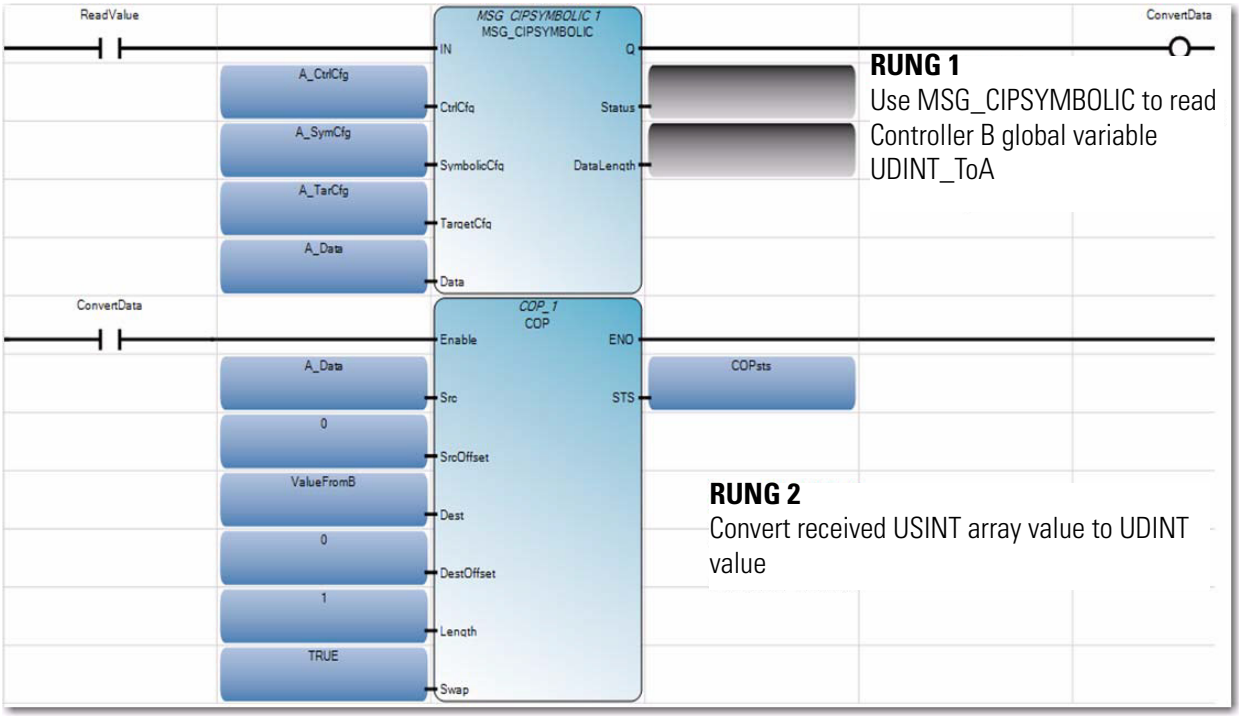


Create CIP Symbolic Program (Read)

The previous program illustrated how a write instruction is sent from Controller A to Controller B using the MSG_CIPSYMBOLIC function block. This section shows how Controller A reads the value of Controller B's global variable, UDINT_ToA.



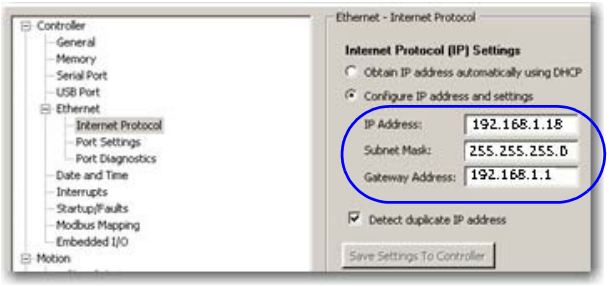
At the end of these instructions, you should have created the following program in Connected Components Workbench in your Controller A project.



Build the Code

1. Open the Connected Components Workbench project you have created for Controller A in the last chapter.

2. Configure the IP address, subnet mask and gateway of Controller A as shown.



IP Address: 192.168.1.18

Subnet Mask: 255.255.255.0

Gateway Address: 192.168.1.1

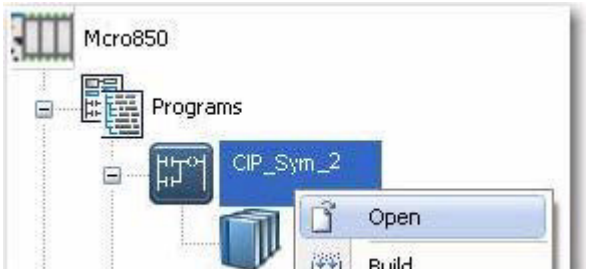
3. On the Project Organizer pane, right-click Programs, and then select Add New LD: Ladder Diagram.
You can press F2 to rename the program. Press Enter.

4. Create the following local variables.

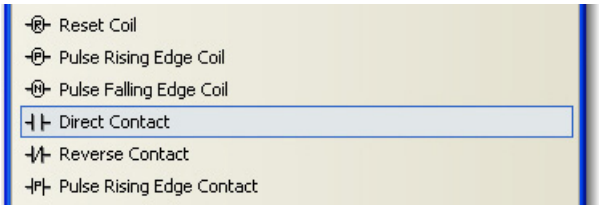
Local Variables

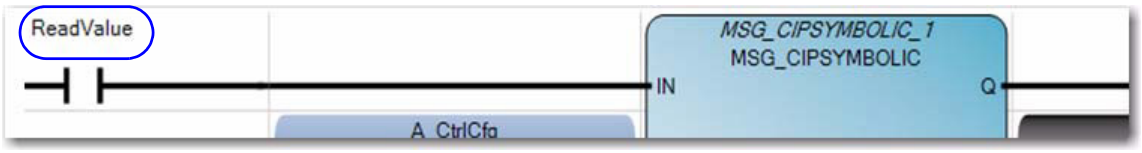
Variable Name	Data Type	Dimension
ValuleToRead	UDINT	[1…1]
A_Data	USINT	[1…4]
COPsts	UINT	
ConvertData	BOOL	
ReadValue	BOOL	

5. Right-click CIP_Sym_2 program, choose Open.



6. Add ReadValue contact to the rung by selecting Direct Contact from the Toolbox and drag it onto the rung as shown.

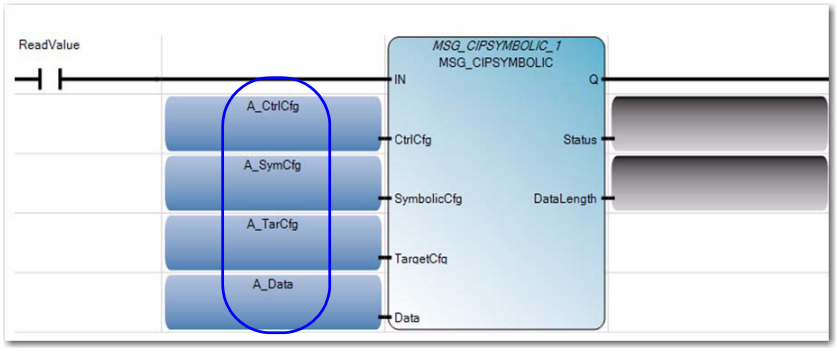




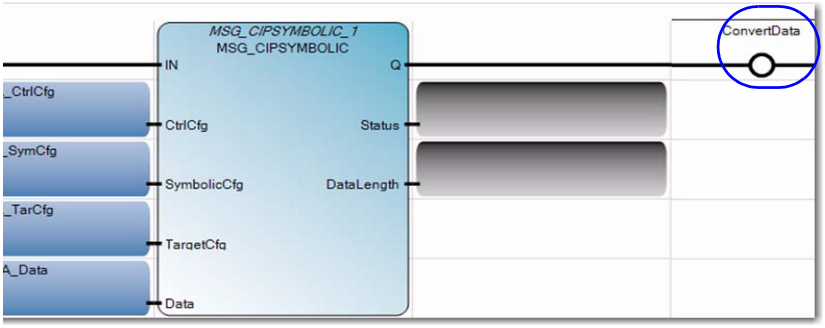
7. Add MSG_CIPSYMBOLIC function block to the rung and add the input variables as shown.

   To add the function block, double-click Block on the Toolbox to add it to the rung. Alternatively, you can drag and drop Block onto the rung.

   Type MSG_CIPSYMBOLIC on the Instruction Block Selector window that appears to filter the function block. Choose MSG_CIPSYMBOLIC. Then, click OK.

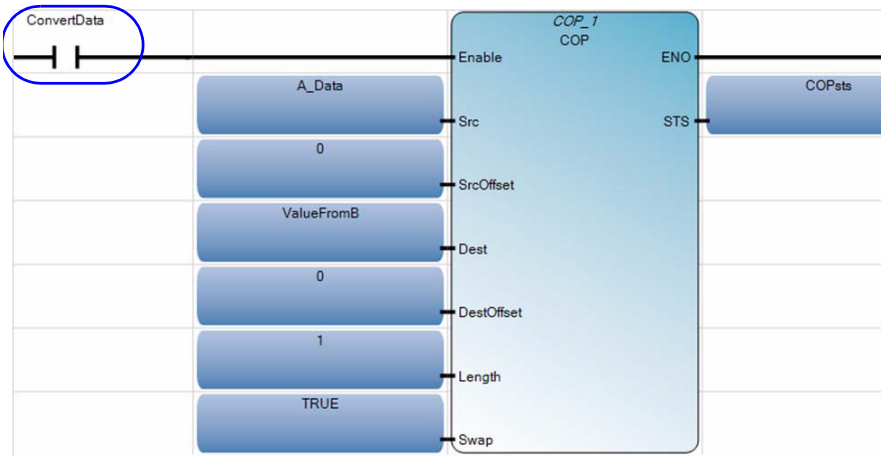


8. Add ConvertData coil to the end of the rung by selecting Direct Coil from the Toolbox and add it to the rung as shown.



9. Create a second rung. From the Toolbox by selecting rung and drag it onto the space just below the first rung.

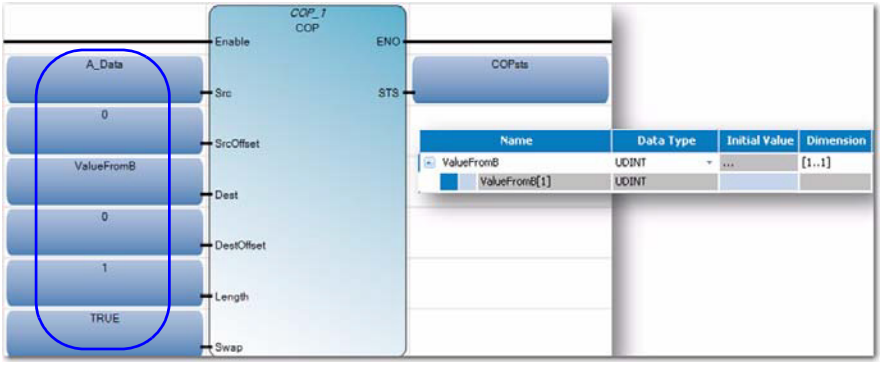
10. Add ConvertData contact to the rung. To do this, select Direct Contact from the Toolbox and drag it onto the rung as shown.



11. Add COP instruction to the rung.

12. Create or specify values/variables for the COP input parameters as shown.

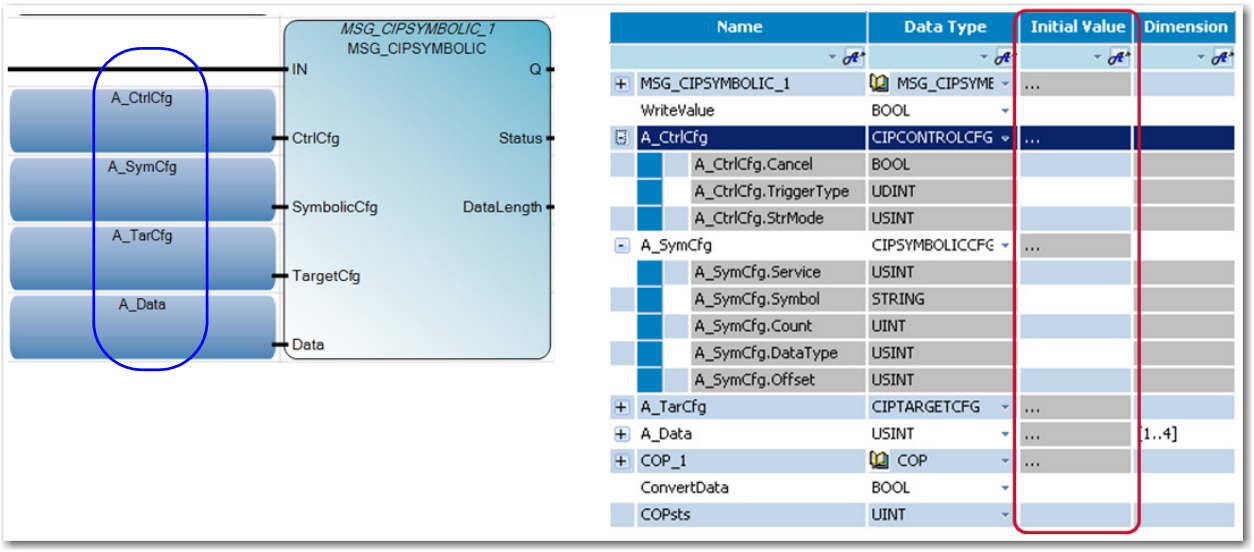
    To learn more about the COP input and output parameter descriptions, see COP Function Block on page 43.



IMPORTANT ValueFromB input variable is a one-dimensional array with one element.

Assign Values to the MSG_CIPSYMBOLIC Input Variables

1. Set the values for the input variables through the Initial Value column of the Global Variables table.



Name	Data Type	Initial Value	Dimension
MSG_CIPSYMBOLIC_1	MSG_CIPSYME	...	
WriteValue	BOOL		
A_CtrlCfg	CIPCONTROLCFG	...	
A_CtrlCfg.Cancel	BOOL		
A_CtrlCfg.TriggerType	UDINT		
A_CtrlCfg.StrMode	USINT		
A_SymCfg	CIPSYMBOLICCFG	...	
A_SymCfg.Service	USINT		
A_SymCfg.Symbol	STRING		
A_SymCfg.Count	UINT		
A_SymCfg.DataType	USINT		
A_SymCfg.Offset	USINT		
A_TarCfg	CIPTARGETCFG	...	
A_Data	USINT	...	[1..4]
COP_1	COP	...	
ConvertData	BOOL		
COPsts	UINT		

See MSG_CIPSYMBOLIC Function Block on page 35 for more information about MSG_CIPSYMBOLIC parameters.

2. Set the value for the A_CtrlCfg input variable as shown. See CtrlCfg on page 35 for parameter details and description.

Name	Data Type	Initial Value
A_CtrlCfg	CIPCONTROLCFG	...
A_CtrlCfg.Cancel	BOOL	
A_CtrlCfg.TriggerType	UDINT	300
A_CtrlCfg.StrMode	USINT	

   Set A_CtrlCfg.TriggerType as 300 ms. Set A_CtrlCfg.Cancel to False. By default, this parameter is set to False.

3. Set the value for the A_SymCfg input variable as shown. See SymbolicCfg on page 36 for parameter details and description.
   Leave the default initial value of A_SymCfg.Count at 0.

Name	Data Type	Initial Value	Dimension
A_SymCfg	CIPSYMBOLICCFG	...	
A_SymCfg.Service	USINT	0	
A_SymCfg.Symbol	STRING	'UDINT_ToA'	
A_SymCfg.Count	UINT		
A_SymCfg.DataType	USINT	200	
A_SymCfg.Offset	USINT		

4. Set the value for the A_TarCfg input variable as shown.
   See TargetCfg on page 37 for parameter details and description.
   Set A_TarCfg.ConnClose to False. By default, this variable is set to false.

A_TarCfg	CIPTARGETCFG	...
A_TarCfg.Path	STRING	'4,192.168.1.19'
A_TarCfg.CipConnMode	USINT	1
A_TarCfg.UcmmTimeout	UDINT	0
A_TarCfg.ConnMsgTimec	UDINT	0
A_TarCfg.ConnClose	BOOL	

IMPORTANT A_Data variable will store the value that is read from Controller B.

Name	Data Type	Initial Value	Dimension
A_Data	USINT	...	[1..4]
A_Data[1]	USINT		
A_Data[2]	USINT		
A_Data[3]	USINT		
A_Data[4]	USINT		

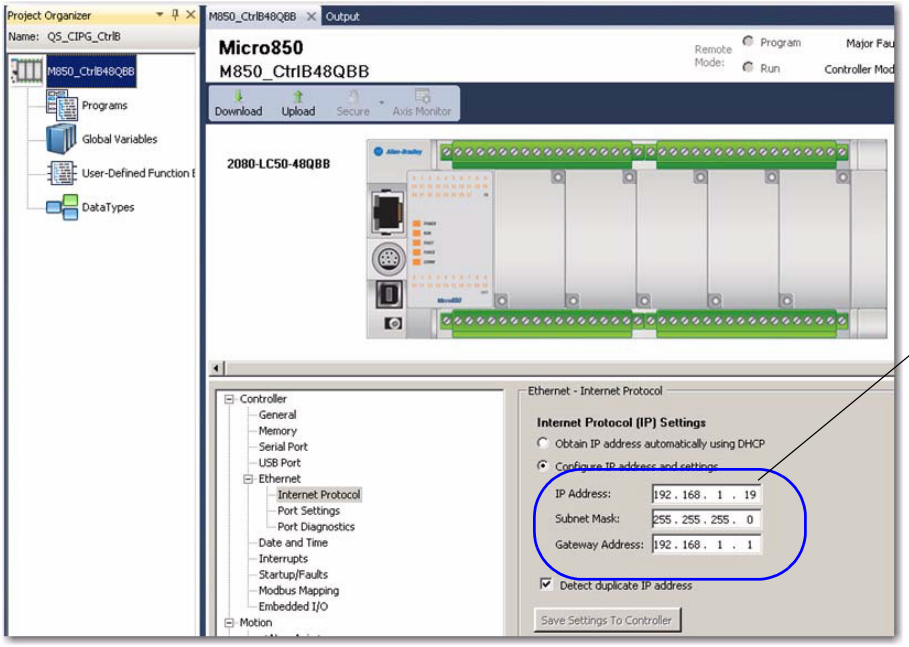
Note that USINT is 8-bit data (whereas A UDINT is 32-bit data), so the A_Data input variable is a one-dimension array with four elements.

Build and Download the Project

Next, build and download the project to Controller A. Then, proceed to the next section.

Configure Controller B

1. Make sure that Controller B is configured as shown:

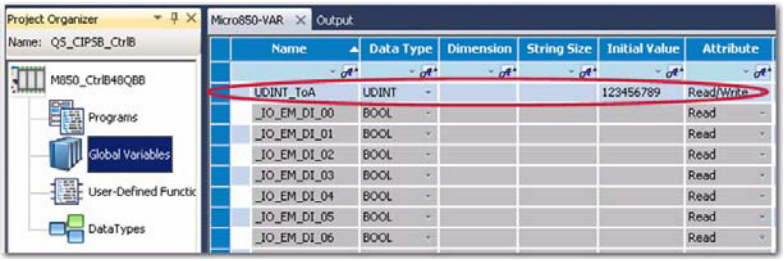


IP Address: 192.168.1.19

Subnet Mask: 255.255.255.0

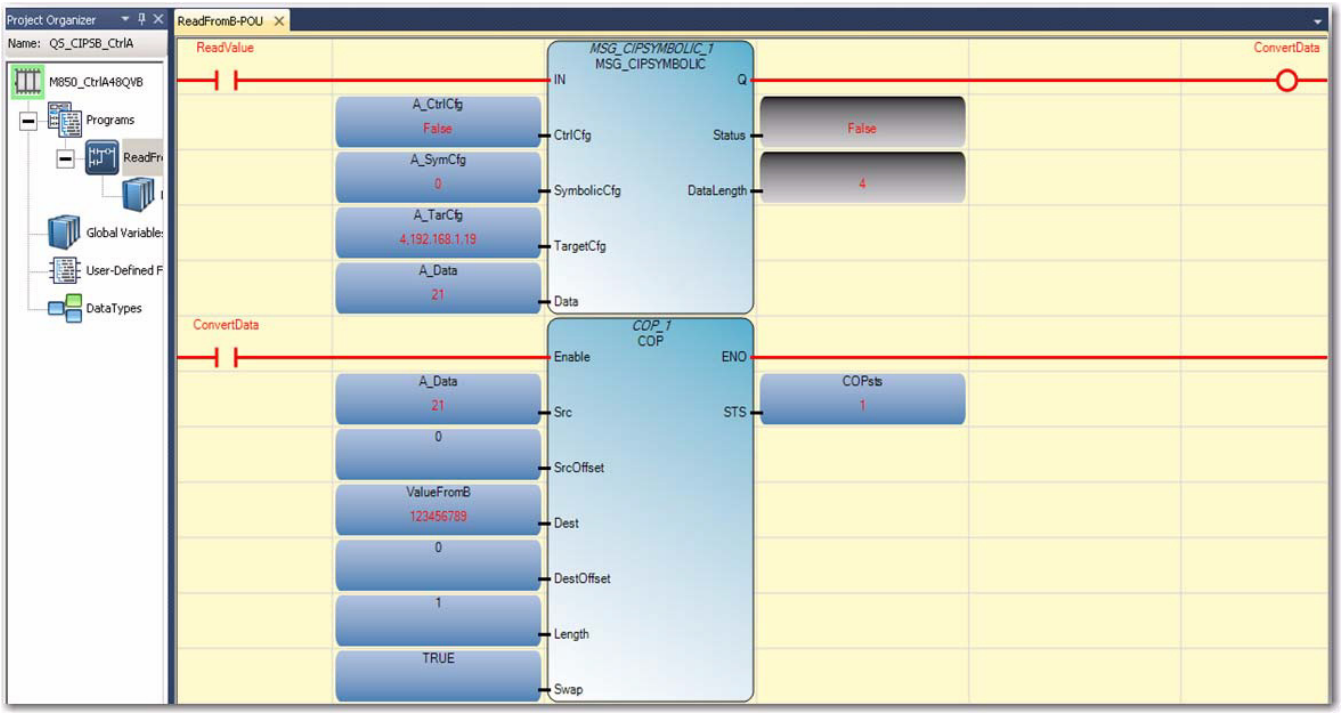
Gateway Address: 192.168.1.1

2. Create the Global Variable 'UDINT_ToA' with an initial value of 123456789.



3. Build and download the project to Controller B.

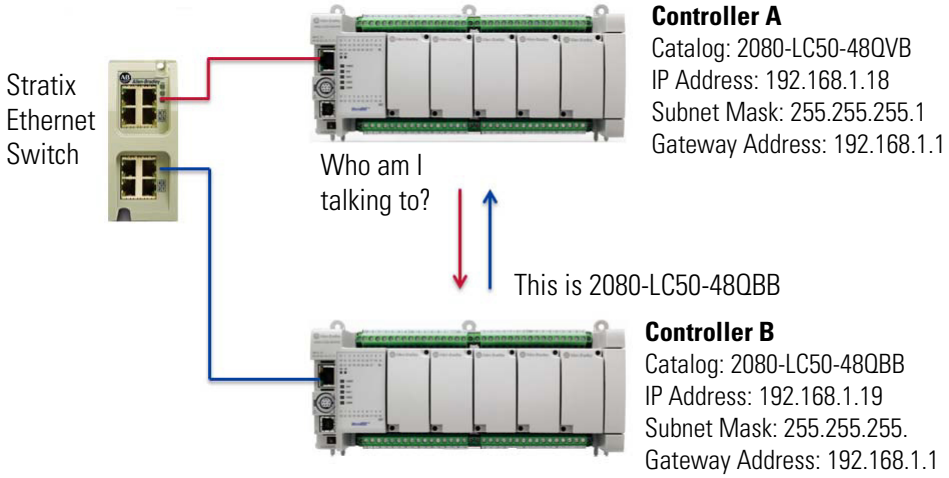
See the Results



Use CIP Generic Client Messaging

Introduction

The MSG_CIPGENERIC function block enables the sending of CIP explicit messages through Ethernet or serial cable. In the following example, the function block is used by Controller A to query the catalog number of Controller B.



The catalog information can be retrieved by accessing Micro800 Identity Object:

Class Code: 01
Instance: 01
Instance Attribute: 07 (Catalog number in Short String format)
Service: 14 (Get Single attribute value)

Before You Begin

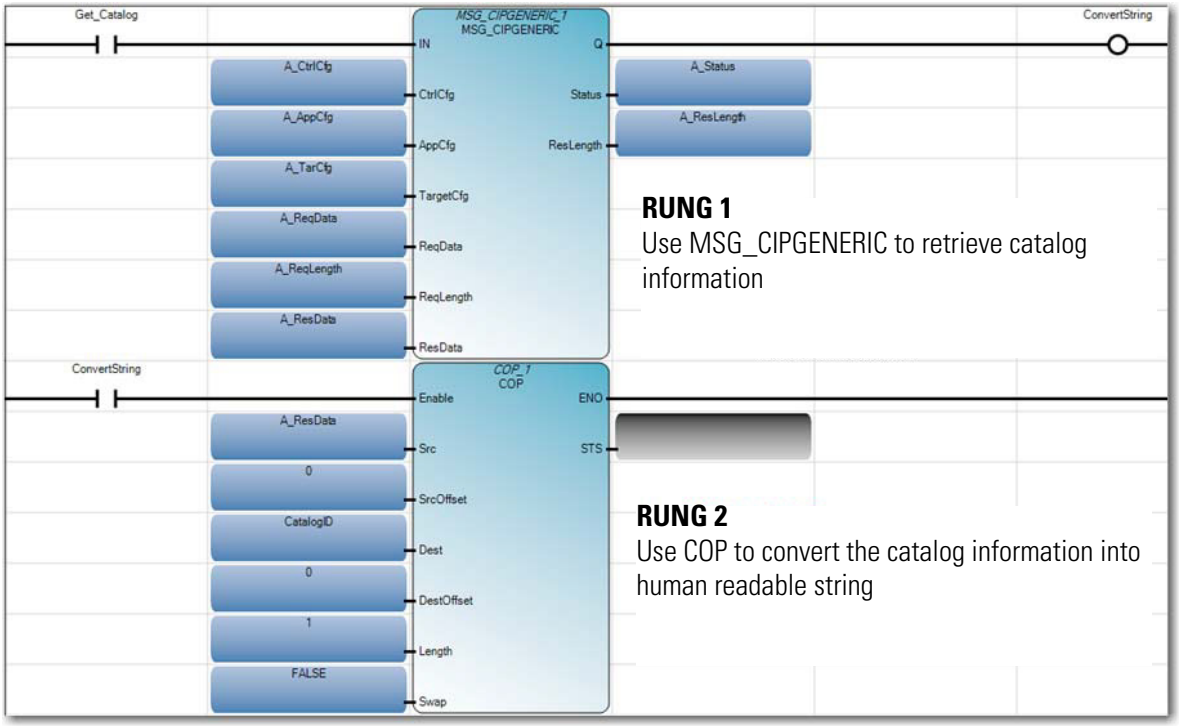
Ensure that you have properly installed revision 4 or later of the Connected Components Workbench software.

What You Need

- Connected Components Workbench software revision 4 or later
- Firmware revision 4 or later

Create CIP Generic Message Program

At the end of these instructions, you should have created the following program in Connected Components Workbench.

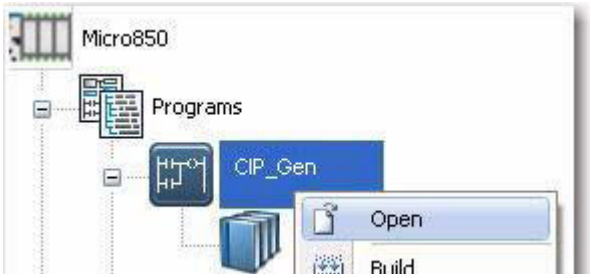


Build the Code

1. Open the project you have created for Controller A in chapter 1, through Connected Components Workbench.

2. On the Project Organizer pane, right-click Programs by selecting Add New LD: Ladder Diagram. You can press F2 to rename the program to CIP_Gen. Press Enter.

3. Right-click CIP_Gen program, choose Open.



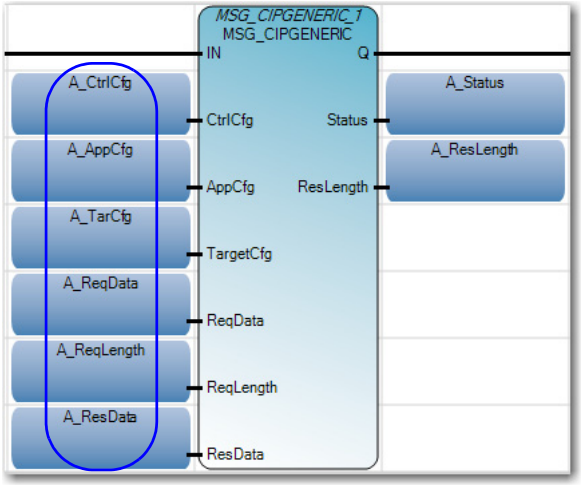
4. From the Toolbox, double-click Block to add it to the rung. Alternatively, you can drag and drop Block onto the rung. Your ladder rung should appear as shown.

5. Add MSG_CIPGENERIC function block to the rung by typing MSG_CIPGENERIC on the Instruction Block Selector window that appears to filter the function block. Choose MSG_CIPGENERIC. Then, click OK.

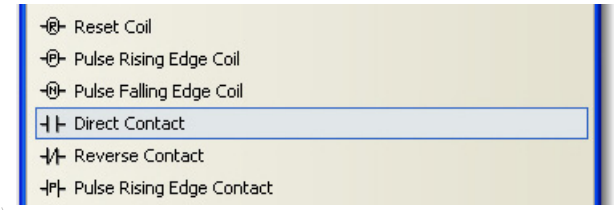
6. Create the input variables for the MSG_CIPGENERIC as shown.

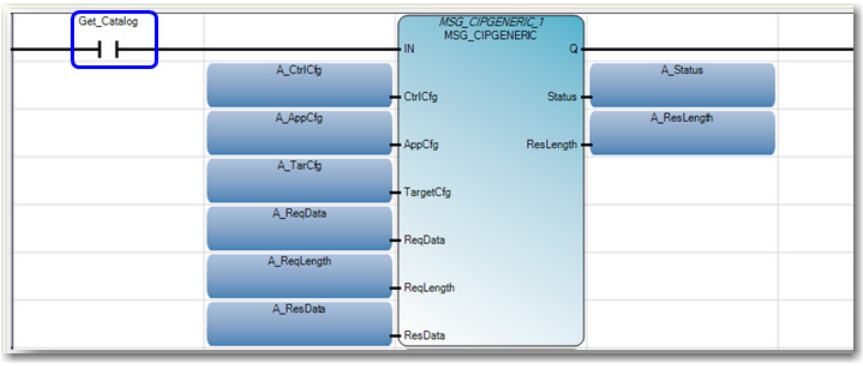
   Also, create BOOL variables GetCatalog and ConvertString.

   To learn more about the MSG_CIPGENERIC input and output parameter descriptions, see MSG_CIPGENERIC Function Block on page 39.

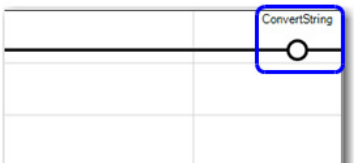


7. Add Get_Catalog contact to the rung by selecting Direct Contact from the Toolbox and drag it onto the rung as shown.



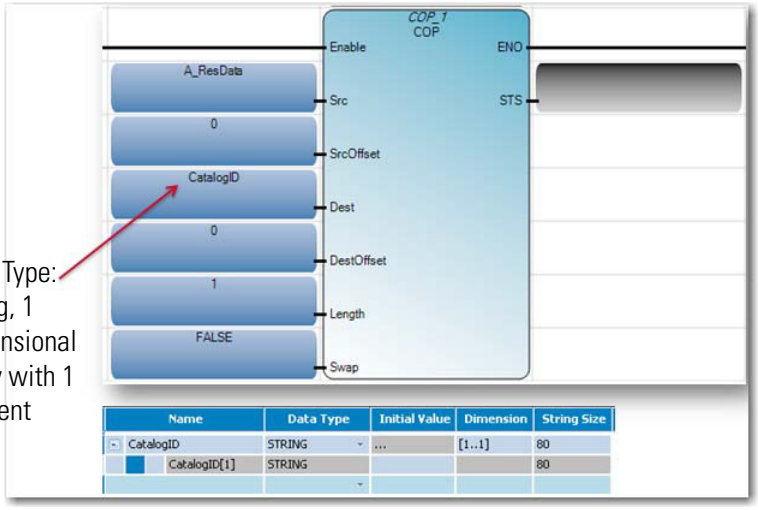


8. Add ConvertString coil to the end of the rung by selecting Direct Coil from the Toolbox and add it to the rung as shown.



9. Create a second rung. From the Toolbox by selecting rung and drag it onto the space just below the first rung.

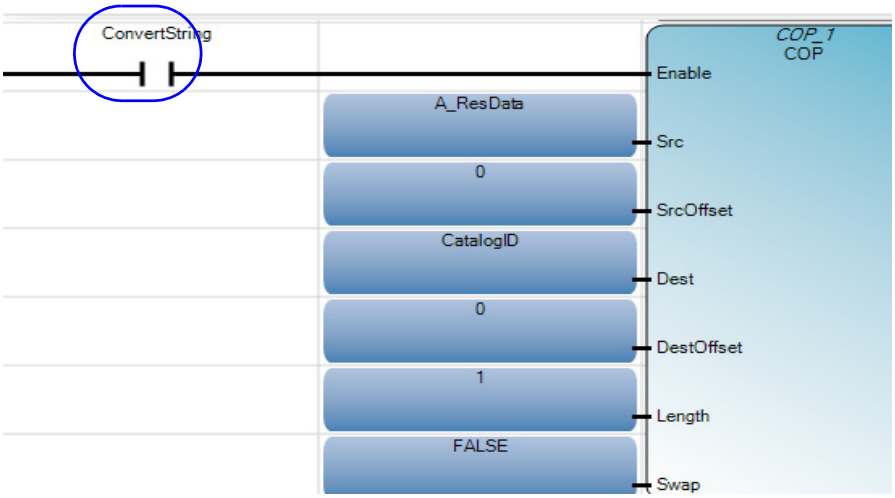
10. Add COP instruction to the rung by typing COP on the Instruction Block Selector window that appears to filter the COP function block. Choose COP. Then, click OK.



11. Create the COP input variables as shown.

    For more information about COP input and output parameters, see COP Function Block on page 43.

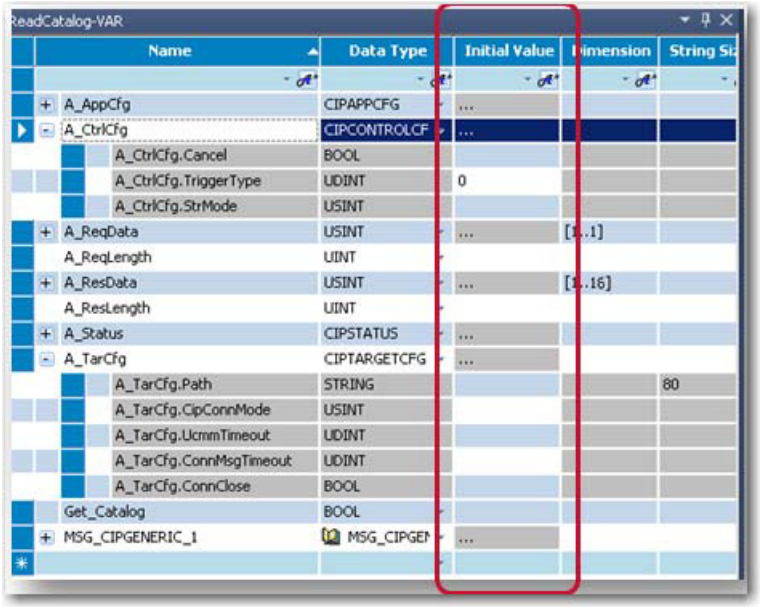
12. Add ConvertString contact to the start of the rung by selecting Direct Contact from the Toolbox and drag it onto the rung as shown.



Assign Values to MSG_CIPGENERIC Input Variables

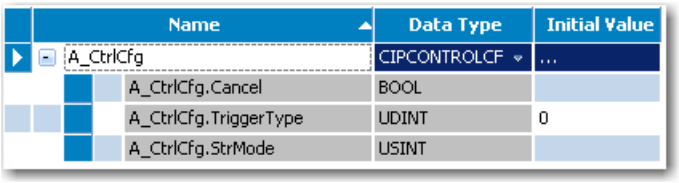
1. Set the values for the input variables through the Initial Value column of the Global Variables table.

   See MSG_CIPGENERIC Function Block on page 39 for details and description of the function block parameters.

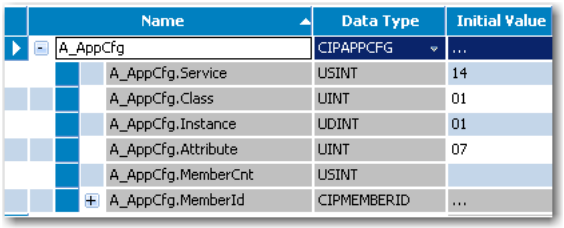


2. Set the value for the A_CtrlCfg input variable as shown. See CtrlCfg on page 39 for parameter details and description.

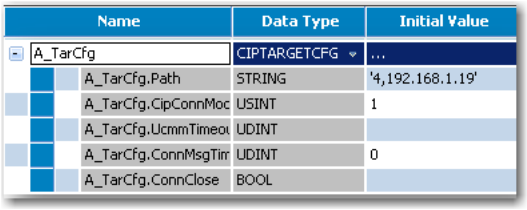
   In this example, we just need to know the catalog number once, and set A_CtrlCfg.TriggerType initial value to 0.



3. Set the value for the A_AppCfg input variable as shown. See AppCfg on page 40 for parameter details and description.



4. Set the value for the A_TargetCfg input variable as shown.
   See TargetCfg on page 40 for parameter details and description.



Name	Data Type	Initial Value
A_TarCfg	CIPTARGETCFG	...
A_TarCfg.Path	STRING	'4,192.168.1.19'
A_TarCfg.CipConnMoc	USINT	1
A_TarCfg.UcmmTimeou	UDINT	
A_TarCfg.ConnMsgTim	UDINT	0
A_TarCfg.ConnClose	BOOL	

TIP **ReqData and ReqLength Input Variables**

As this project only needs to read the destination value, there is no need to assign values to the input variables, A_ ReqData and A_ReqLength.

See ReqData on page 40 and ReqLength on page 40 for parameter details and description.

IMPORTANT **ResData Input Variable**

This is an output parameter that stores the response data in USINT array format.

In our example, we are reading the catalog number string. When it is stored as USINT array, it will follow the ODVA short string format which is the first element in the array that will specify the string length. The next array elements will store the string character's HEX value.

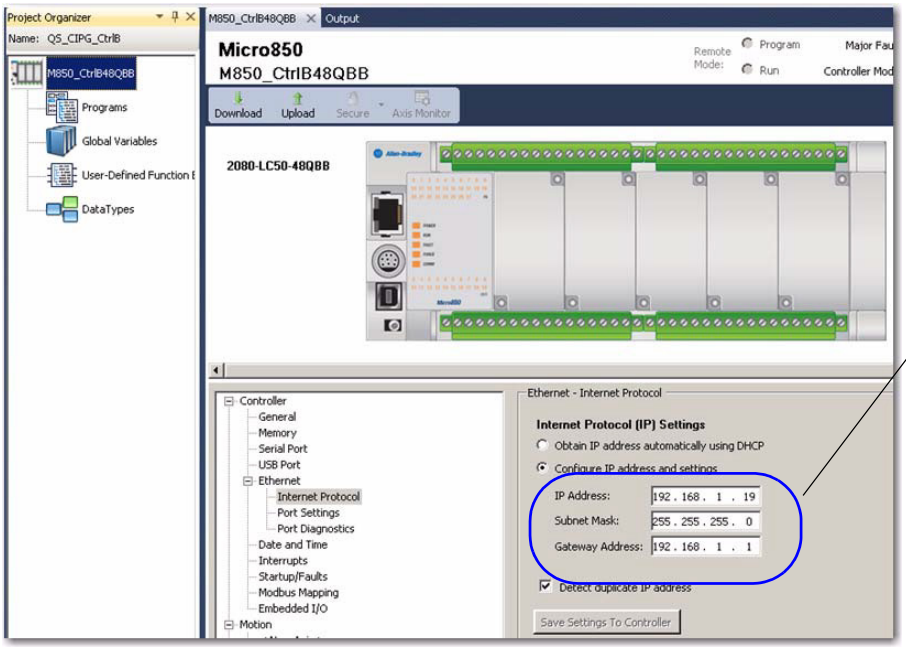
For a string the maximum number of character is 80, plus the length element, so we could define the A_ResData as 1 dimension array with 81 elements.

See ResData on page 41 for parameter details and description.

5. Build and download the project to Controller A.

Configure Controller B

Make sure that Controller B is configured as shown:

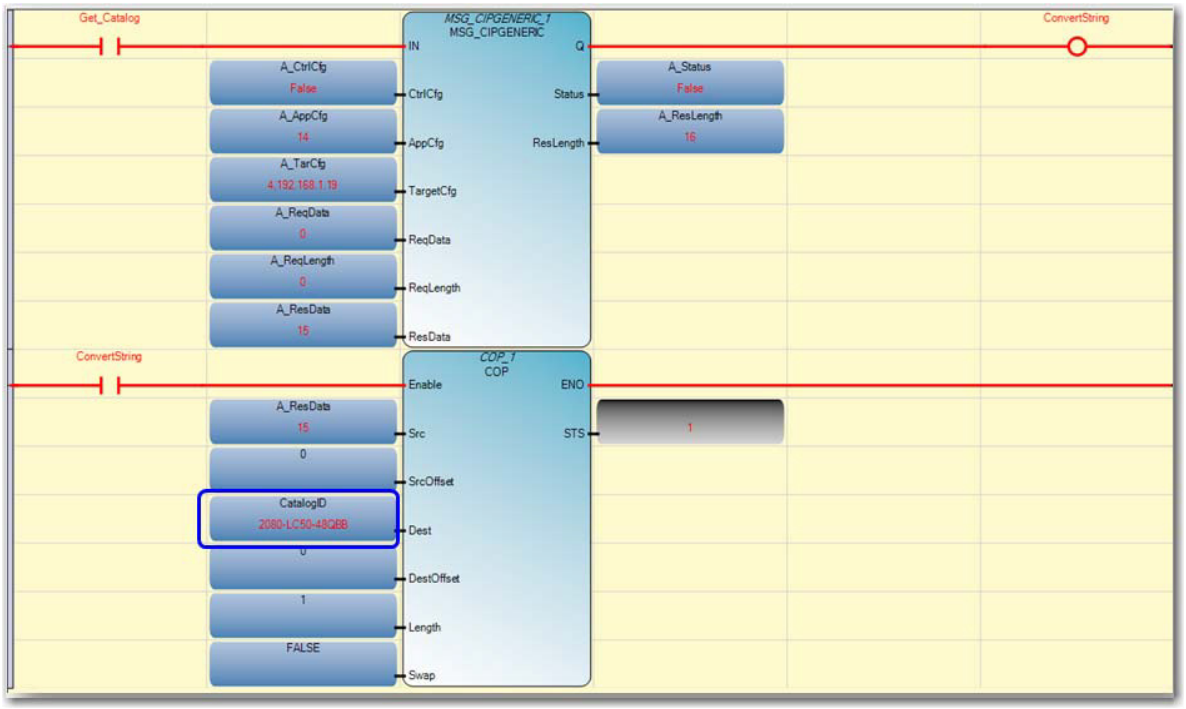


IP Address:
192.168.1.19

Subnet Mask:
255.255.255.0

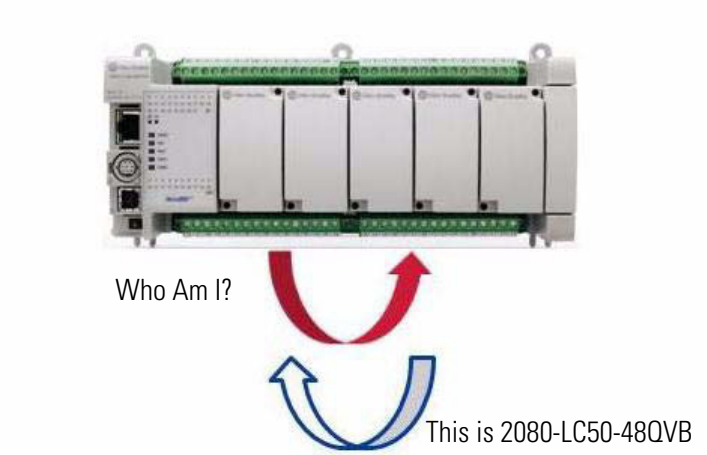
Gateway Address:
192.168.1.1

See the Results

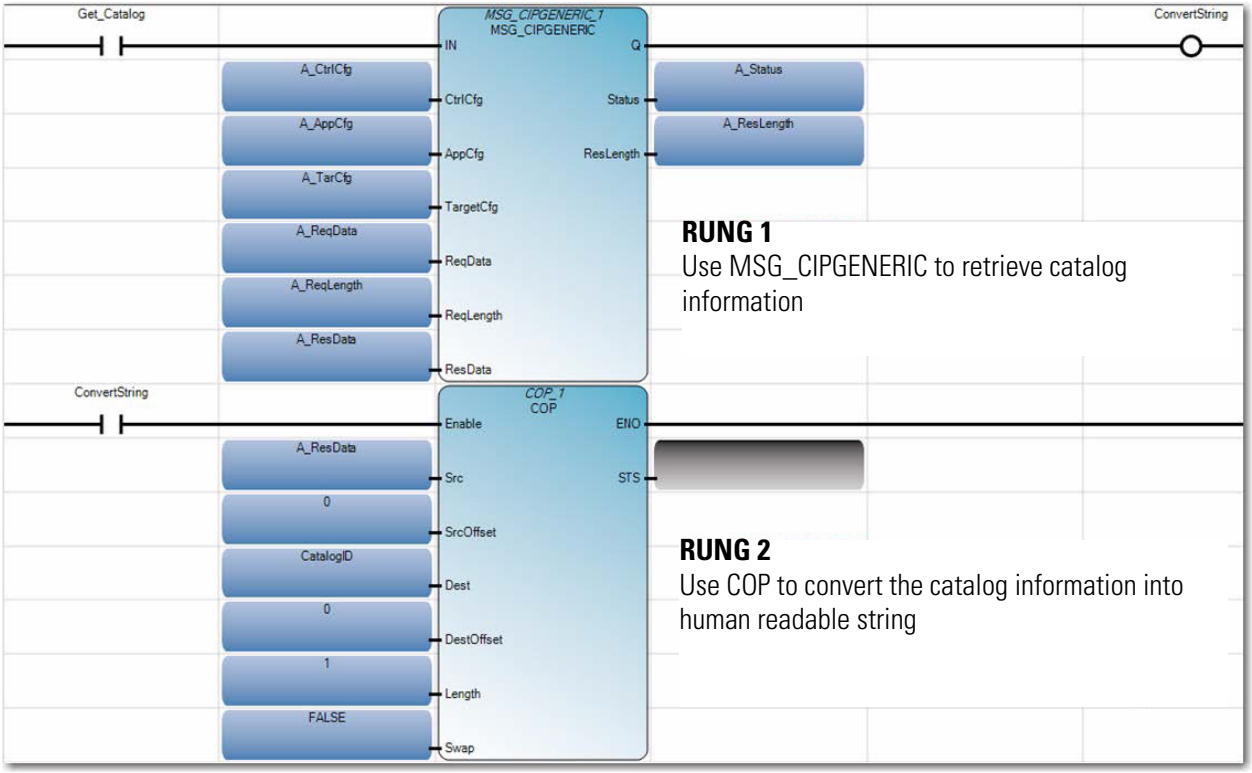


Create CIP Generic Message Program: Single Controller

The previous program illustrated how a read instruction is sent from Controller A to Controller B using the MSG_CIPGENERIC function block. This section shows how a controller queries its own catalog number using the same function block.



At the end of these instructions, you should have created the following program in Connected Components Workbench in your Controller A project.



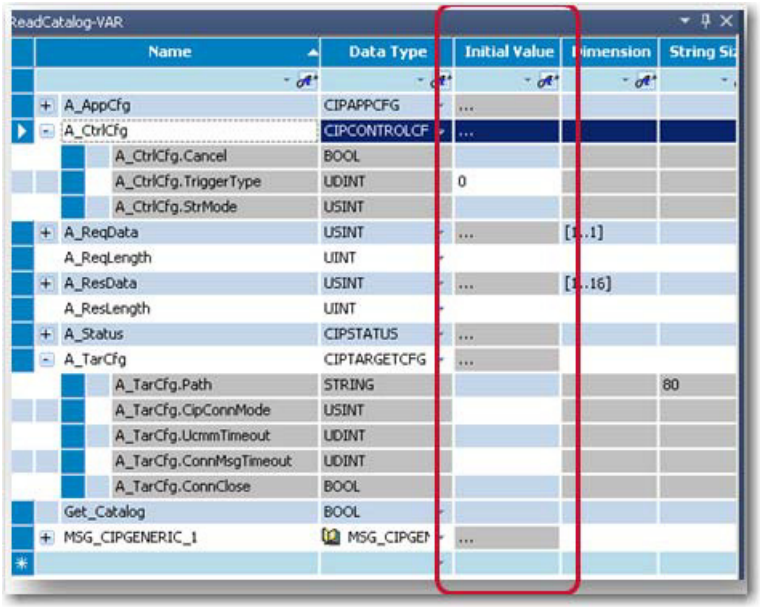
Build the Code

To set up your program, perform the same instructions on Build the Code on page 24.

Assign Values to MSG_CIPGENERIC Input Variables

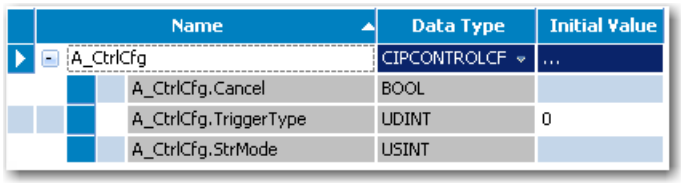
1. Set the values for the input variables through the Initial Value column of the Global Variables table.

   See MSG_CIPGENERIC Function Block on page 39 for details and description of the function block parameters.

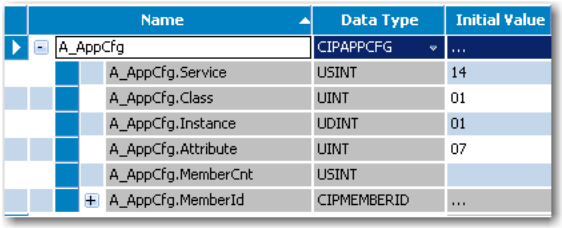


2. Set the value for the A_CtrlCfg input variable as shown. See CtrlCfg on page 39 for parameter details and description.

   In this example, we just need to know the catalog number once, and set A_CtrlCfg.TriggerType initial value to 0.

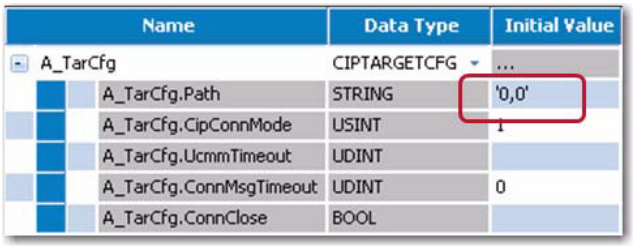


3. Set the value for the A_AppCfg input variable as shown. See AppCfg on page 40 for parameter details and description.



4. Set the value for the A_TargetCfg input variable as shown.
   See TargetCfg on page 40 for parameter details and description.

   Note that for a single controller program, the initial value for A_TarCfg.Path is '0,0'. The previous read program involving Controller A and Controller B, initial value for for this variable is '4,192.168.1.19'.



Name	Data Type	Initial Value
A_TarCfg	CIPTARGETCFG	...
A_TarCfg.Path	STRING	'0,0'
A_TarCfg.CipConnMode	USINT	1
A_TarCfg.UcmmTimeout	UDINT	
A_TarCfg.ConnMsgTimeout	UDINT	0
A_TarCfg.ConnClose	BOOL	

TIP **ReqData and ReqLength Input Variables**

As this project only needs to read the destination value, you do not need to assign values to the input variables, A_ ReqData and A_ReqLength. See ReqData on page 40 and ReqLength on page 40 for parameter details and description.

IMPORTANT **ResData Input Variable**

This is an output parameter that stores the response data in USINT array format.

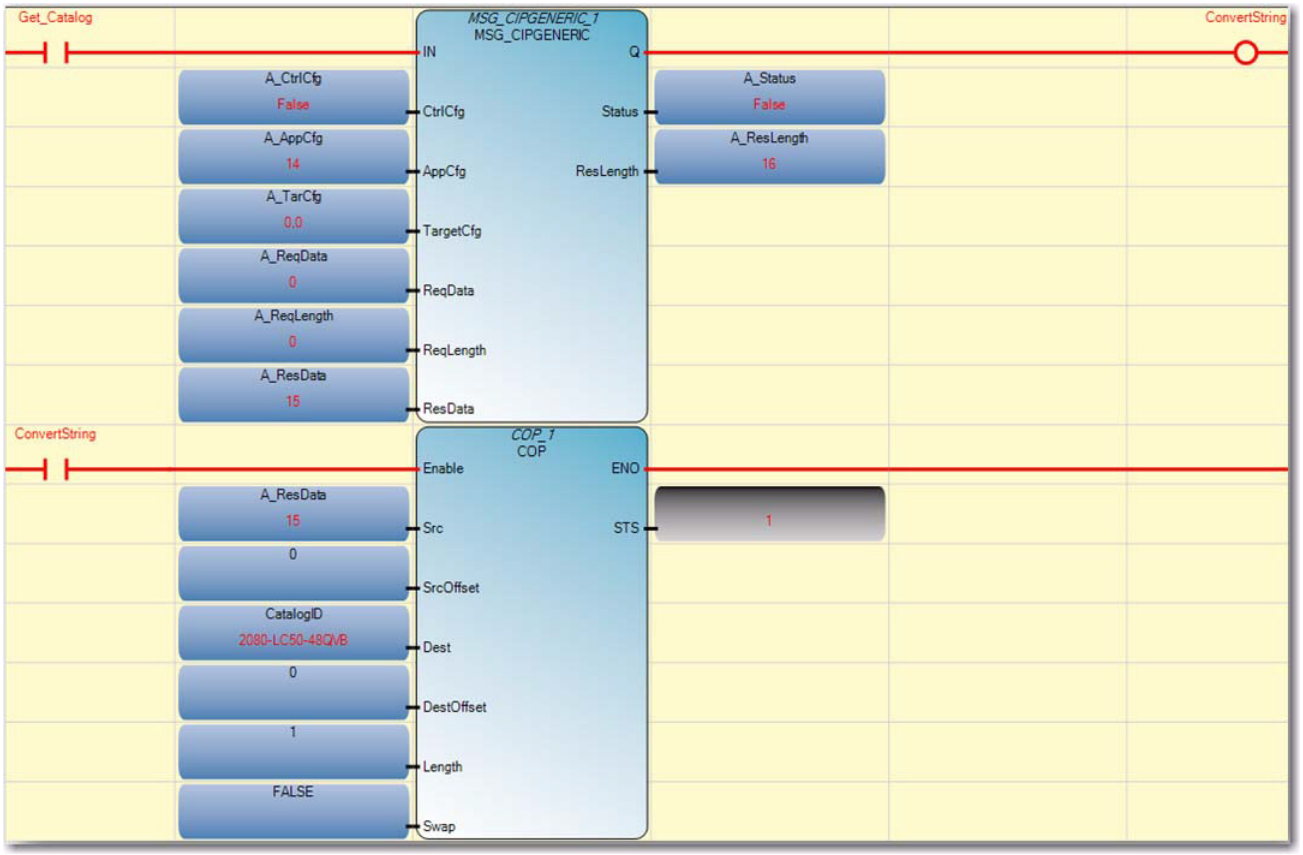
In our example, we are reading the catalog number string. When it is stored as USINT array, it will follow the ODVA short string format which is the first element in the array that will specify the string length. The next array elements will store the string character's HEX value.

For a string the maximum number of character is 80, plus the length element, so we could define the A_ResData as 1 dimension array with 81 elements.

See ResData on page 41 for parameter details and description.

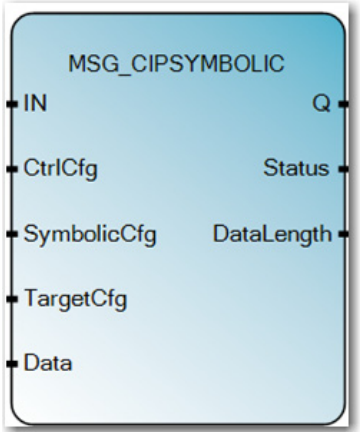
5. Build and download the project to Controller A.

See the Results



MSG_CIPSYMBOLIC Function Block

This diagram shows the arguments for the MSG_CIPSYMBOLIC function block.



The following table explains the input and output parameters for this function block.

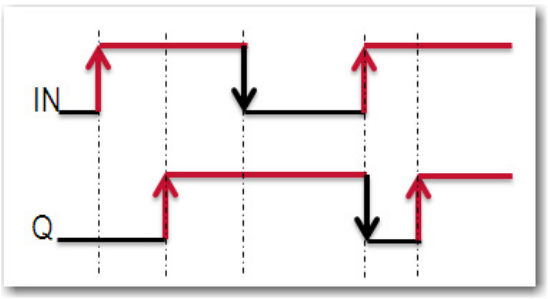
MSG_CIPSYMBOLIC Parameters

Parameter	Parameter Type	Data Type	Description
IN	Input	BOOLEAN	If Rising Edge (IN turns from 0 to 1), start the function block with the precondition that the last operation has been completed. When the function block is enabled, the receive buffers for the Read operations are cleared on the rising edge of IN.
CtrlCfg	Input	CIPCONTROLCFG	Control Configuration. This is a structured data type. It consists of the following elements: • CtrlCfg.Cancel (Boolean) - When this value is true, the execution of MSG_CIPSYMBOLIC will be cancelled - The bit is cleared when message is enabled • CtrlCfg.TriggerType (UDINT) - If set to "0", then the message only trigger once when "IN" is true - If set to 1 to 65535 (unit : milliseconds), then triggers periodically when "IN" is true. For example a value of 100 means when "IN" is true, the message will be triggered every 100 milliseconds • CtrlCfg.StrMode (USINT) - Reserved.

MSG_CIPSYMBOLIC Parameters

Parameter	Parameter Type	Data Type	Description
SymbolicCfg	Input	CIPSYMBOLICCFG	Information of symbol to Read/Write. This is a structured data type. It consists of the following elements: • SymbolicCfg.Service (USINT) Read/Write service 0 - Read (Default)| 1 - Write • SymbolicCfg.Symbol (STRING) Name of the variable to Read/Write. This field cannot be empty. Maximum 80 characters can be entered. *Example 1*: to read an array MyArray[1..100] then we should put SymbolicCfg.Symbol := 'MyArray'; *Example 2*: to read an array MyArray[1..100], Start from 5th element of array then SymbolicCfg.Symbol := 'MyArray[5]'; • SymbolicCfg.Count (UINT) Number of variable elements to Read/Write. 1...65535, 1 is used if this value is set as 0 (default). • SymbolicCfg.Type (USINT) Data type value of Data being Read/Write. This field cannot be empty for Write. For Read, function block execution fills this field with received data type. For Write, user specifies the data type to be written. See CIP Message Data Types on page 65. • SymbolicCfg.Offset (USINT) Reserved for future use. Byte offset of variable to Read/Write. This field is used to Read/Write a large size of variable which cannot be processed in one message. Range: 0...255.

MSG_CIPSYMBOLIC Parameters

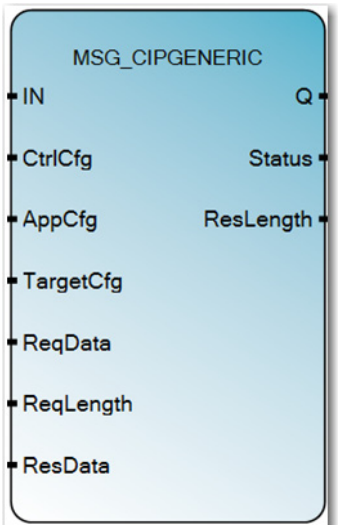
Parameter	Parameter Type	Data Type	Description
TargetCfg	Input	CIPTARGETCFG	Target device configuration. This is a structured data type. It consists of the following elements: • TargetCfg.Path (String) Target information. Maximum two hops can be specified. {"<port>,<node/slot address>"} *Example1*: From Micro850 embedded Ethernet port (which is port number 4), send message to a device with in same subset network with IP address 192.168.1.100 then the path is TargetCfg.Path := '4, 192.168.1.100' *Example2*: From Micro830 Embedded serial port (which is port number 2), to reach a device at Node 1. TargetCfg.Path := '2, 1' • TargetCfg.CipConnMode (USINT) CIP Connection type. '0' – Unconnected (Default), '1' – Class 3 connection • TargetCfg.UcmmTimeout (UDINT) Unconnected message timeout (in milliseconds) Amount of time to wait for a reply for an unconnected messages (including connection establishment for connected message) Range: 250...10,000, set to 0 to use the default value 3000. Specifying any value less than 250 is set as 250 and any value greater than maximum value is set as 10,000. • TargetCfg.ConnMsgTimeout (UINT) Class3 Connection timeout (in milliseconds) Amount of time to wait for a reply for connected messages. The CIP connection is closed if this timeout expires. When using bridge or serial communication, the value must be set to 880 or greater. Range: 800...10,000, set to 0 to use the default value 10000. Specifying any value less than 800 will be set as 800 and any value greater than maximum value is set as 10,000. • TargetCfg.ConnClose (UINT) Connection closing behavior. True: Close the CIP connection upon message completion. False: Do not close the connection upon message completion [Default].
Data	Input	USINT Array	This is an Input/Output parameter. Read command stores the data returned from server. Write command buffers the data to be sent to the server. When a MSG is triggered (or re-triggered), data is cleared for MSG Read command. Range: 1…490
Q	Output	BOOLEAN	Outputs of this instruction are updated asynchronously from the program scan. Output Q cannot be used to re-trigger the instruction since IN is edge triggered. TRUE: MSG instruction is finished successfully. FALSE: MSG instruction is not finished. After 'Q' is true, even when 'IN' turns to false, the Q stays true until 'IN' is triggered again. 

MSG_CIPSYMBOLIC Parameters

Parameter	Parameter Type	Data Type	Description
Status	Output	CIPSTATUS	Instruction execution status. When a MSG is triggered (or re-triggered), all elements inside Status are reset. This is a structured data type. It consists of the following elements: • Status.Error (Boolean) This bit is set to TRUE when function block execution encountered an error condition. • Status.ErrorID (UINT) Error code value. See CIP Error Message Codes on page 63. • Status.SubErrorID (UDINT) Sub Error code value. See CIP Error Message Codes on page 63. • Status.ExtErrorID (UINT) CIP extended status error code value • Status.StatusBits (UINT)  This parameter can be used to verify various control bits. - Bit 0: EN – Enable - Bit 1: EW – Enable Wait - Bit 2: ST – Start - Bit 3: ER – Errors (same state as Q) - Bit 4: DN – Done (same state as Q) - Other bits are reserved.
DataLength	Output	UINT	CIP read message response data length and CIP write message request data length (Unit: Byte). When a read MSG is triggered (or re-triggered), DataLength is reset to 0 for MSG Read command. Range: 0...490

MSG_CIPGENERIC Function Block

This diagram shows the arguments in the MSG_CIPGENERIC function block.



The following table explains the arguments used in this function block.

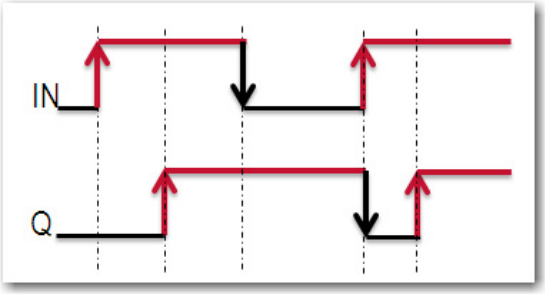
MSG_CIPGENERIC Parameters

Parameter	Parameter Type	Data Type	Description
IN	Input	BOOLEAN	If Rising Edge (IN turns from 0 to 1), start the function block with the precondition that the last operation has been completed. When the function block is enabled, the receive buffers for the Read operations are cleared on the rising edge of IN.
CtrlCfg	Input	CIPCONTROLCFG	Control Configuration. This is a structured data type, it consists of the following elements: • CtrlCfg.Cancel (Boolean) - When this value is true, the execution of MSG_CIPGENERIC is cancelled. - The bit is cleared when message is enabled. • CtrlCfg.TriggerType (UDINT) - If set to "0", then the message only trigger once when "IN" is true. - If set to 1 to 65535 (unit : milliseconds), then triggers periodically when "IN" is true. For example a value of 100 means when "IN" is true, the message is triggered every 100 milliseconds. • CtrlCfg.StrMode (USINT) - Reserved.

MSG_CIPGENERIC Parameters

Parameter	Parameter Type	Data Type	Description
AppCfg	Input	CIPAPPCFG	CIP Service and application path configuration. This is a structured data type, that consists of the following elements: • AppCfg.Service (USINT) - Service code • AppCfg.Class (UINT) - Logical segment's Class ID value • AppCfg.Instance (UDINT) - Logical segment's Instance ID value • AppCfg.Attribute (UINT) - Logical segment's attribute value • AppCfg.MemberCnt (UINT) - Maximum Member ID values used (Normally this parameter not used, which means set to 0); 0 =No Member ID used • AppCfg.MemberId (UINT) - Member ID values (Normally this parameter is not used.)
TargetCfg	Input	CIPTARGETCFG	Target device configuration. This is a structured data type. It consists of the following elements: • TargetCfg.Path (String) Target information. Maximum two hops can be specified. {"<port>,<node/slot address>"} *Example 1*: From Micro850 embedded Ethernet port (which is port number 4) send message to a device within same subset network with IP address 192.168.1.100 then the path is TargetCfg.Path := '4, 192.168.1.100' *Example 2*: From Micro830 embedded serial port (which is port number 2) to reach a device at Node 1. TargetCfg.Path := '2, 1' • TargetCfg.CipConnMode (USINT) CIP Connection type. '0' – Unconnected (Default), '1' – Class 3 connection • TargetCfg.UcmmTimeout (UDINT) Unconnected message timeout (in milliseconds) Amount of time to wait for a reply for an unconnected messages (including connection establishment for connected message) Range: 250...10,000, set to 0 to use the default value 3000. Specifying any value less than 250 is set as 250 and any value greater than maximum value is set as 10,000. • TargetCfg.ConnMsgTimeout (UINT) Class3 Connection timeout (in milliseconds) Amount of time to wait for a reply for connected messages. The CIP connection is closed if this timeout expires. When using bridge or serial communication, the value must be set to 880 or greater. Range: 800...10,000, Set to 0 to use the default value 10000. Specifying any value less than 800 is set as 800 and any value greater than maximum value is set as 10,000. • TargetCfg.ConnClose (UINT) Connection closing behavior. True: Close the CIP connection upon message completion. False: Do not close the CIP connection upon message completion [Default].
ReqData	Input	USINT Array	CIP message request data (data to be written to the destination). The array size should not be less than the 'ReqLength' size.
ReqLength	Input	UINT	CIP message request data length (unit: byte), Number of bytes to write to the destination. Range: 0...490

MSG_CIPGENERIC Parameters

Parameter	Parameter Type	Data Type	Description
ResData	Input	USINT	CIP message response data. The array size should not be less than the 'ResLength' size. When a MSG is triggered (or re-triggered), data in ResData array is cleared. This is an Input/Output parameter.
Q	Output	BOOLEAN	Outputs of this instruction are updated asynchronously from the program scan. Output Q cannot be used to re-trigger the instruction since IN is edge triggered. TRUE: MSG instruction is finished successfully. FALSE: MSG instruction is not finished. After 'Q' is true, even when 'IN' turns to false, the Q stays true until 'IN' is triggered again. 
Status	Output	CIPSTATUS	Instruction execution status. When a MSG is triggered (or re-triggered), all elements inside Status are reset. This is a structured data type. It consists of the following elements: • Status.Error (Boolean) - This bit is set to TRUE when function block execution encountered an error condition. • Status.ErrorID (UINT) - Error code value. See CIP Error Message Codes on page 63. • Status.SubErrorID (UDINT) - Sub Error code value. See CIP Error Message Codes on page 63. • Status.ExtErrorID (UINT) - CIP extended status error code value • Status.StatusBits (UINT)  This parameter can be used to verify various control bits. - Bit 0: EN – Enable - Bit 1: EW – Enable Wait - Bit 2: ST – Start - Bit 3: ER – Error (same state as Q) - Bit 4: DN – Done (same state as Q) - Other bits are reserved.
ResLength	Output	UINT	CIP message response data length (Unit: Byte). When a MSG is triggered (or re-triggered), ResLength is reset to 0 Range: 0...490

Notes:

COP Function Block

This diagram shows the arguments in the COP function block. This instruction can be used to copy the binary data in the Source (Src) array to the Destination (Dest) array. The Source remains unchanged.

This function could be used together with CIP Message instructions to convert data from original, for example, DINT, REAL array data type to the sending data format with CIP message instruction support (that is, USINT array).

The following table explains the arguments used in this function block.

COP Parameters

Parameter	Parameter Type	Data Type	Description
Enable	Input	BOOLEAN	Function block enable. This function block is level triggered. When Enable=TRUE, perform copy. When Enable=FALSE, the function block is not executed.

COP Parameters

Parameter	Parameter Type	Data Type	Description
Src	Input	Supported data types[1]: Boolean, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, DATE, STRING, LWORD, ULINT, LINT, LREAL	Initial element to copy. If source or destination is String data, the other party needs to be String data or USINT (UCHAR and BYTE considered as same data type). Otherwise, data type mismatch is reported. When copied to or from a String Data, ODVA short String format shall be used for data in USINT array. For COP instruction between any pair of data types (no String data as either Source or Destination), copy operation is considered valid, even although the data in the destination are possibly in invalid format. User need to handle the logic at the application level. To copy USINT array to a String array, the data in USINT array need to be in format: - Byte1: Length of first String, - Byte2: First Byte Character, - Byte3: Second Byte Character, - Byte n: Last Byte Character, - Byte (n+1): Length of second String, - Byte (n+2): First Byte Character for second String, and so on.
SrcOffset	Input	UINT	Source element offset. Element offset if source is array data type, otherwise 0 should be set. For array data type, to copy from the first element, the offset should be set as 0.
Dest	Input	Supported data types[1]: Boolean, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, TIME, DATE, STRING, LWORD, ULINT, LINT, LREAL	Copy destination.
DestOffset	Input	UINT	Destination element offset. Element offset if destination is array data type, otherwise 0 should be set. For array data type, to copy to the first element, the offset should be set as 0.
Length	Input	UINT	Number of destination elements to copy. When destination is String data type, it means number of strings to be copied.
Swap	Input	BOOLEAN	TRUE: Swap bytes according to Data Type. If either source or destination is String data, there is no Swap. If both source and destination are one-byte length data, there is no Swap.
STS	Output	UINT	0: Function block not enabled. No operation. 1: COP operation success 2: Destination has spare bytes when copying from String (considering copy is successful). 3: Source data is truncated (considering copy successful). 4: Copy Length is invalid. 5: Data type mismatch when there is String Data type as either Source, or Destination. 6: Source data size is too small for copy. 7: Destination data size is too small for copy. 8: Source Data offset is invalid. 9: Destination Data offset is invalid 10: Data is invalid in either source or destination.

(1) Src and Dest must be in the Array Format, only if one variable is required to be copied, then it has to be defined in array format (for example, variable[1..1]). Structured variables are not supported by the COP instruction.

CIP Objects

This appendix provides information on the CIP objects used with Micro800 controllers.

Identity Object

Class Code: 0x01 (01h)

This object provides identification of and general information about the device. The Identity Object is present in all CIP products.

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Revision	UINT	Revision		0x01
2	Read Only	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device	01	0x01
3	Read Only	Number of Instances	UINT	Number of instances in this class hierarchy	01	0x01

Instance Attribute

Only one instance is supported (Instance ID 01).

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Vendor ID	UINT	Identification of each vendor by number	01 (Allen-Bradley)	0x01
2	Read Only	Device Type	UINT	Identification of general type of product	0x0E (Programmable Logic Controller)	0x0E
3	Read Only	Product Code	UINT	Identification of a particular product of an individual vendor		
4	Read Only	Revision	Struct of:	Revision of the item the Identity Object represents		Product/Revision specific
		Major Revision	USINT			
		Minor Revision	USINT			
5	Read Only	Status	WORD	Summary status of the device	See CIP specifications, Volume 1	0x04 (Configured)
6	Read Only	Serial Number	UDINT	Serial number of the device		Unique 4 bytes
7	Read Only	Product Name	SHORT-STRING	Human readable identification		Example: 2080-LC30-16QVB
8	Read Only	State	WORD	Present state of the device as represented by the state transition diagram	See CIP specifications, Volume 1	0x02 (Standby)
11[(1)]	Read/Write	Active Language	Struct of:	Currently active language for the device	"eng" (English)	
			USINT	The language1 field from the STRING data type		"e"
			USINT	The language2 field from the STRING data type		"n"
			USINT	The language3 field from the STRING data type		"g"

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
12[2]	Read Only	Supported Language List	Array of Struct of:	List of languages supported by character strings of data type STRING within the device	Micro830 and Micro850 controllers "eng" (English) Micro820 controller 1 = "eng" (English) 2 = "deu" (German) 3 = "fra" (French) 4 = "spa" (Spanish) 5 = "ita" (Italian) 6 = "por" (Portuguese) 7 = "dut" (Dutch) 8 = "swe" (Swedish) 9 = "pol" (Polish) 10 = "tur" (Turkish) 11 = "cze" (Czech) 12 = "hun" (Hungarian) 13 = "chi" (Chinese) 14 = "jpn" (Japanese)	
			USINT	The language1 field from the STRING data type		
			USINT	The language2 field from the STRING data type		
			USINT	The language3 field from the STRING data type		
15[1]	Read/Write	Assigned_Name	STRINGI	User assigned name	2-byte unicode of embedded STRING2 data type. Max 64 bytes (32 unicode characters)	Blank
16[1]	Read/Write	Assigned_Description	STRINGI	User assigned description	2-byte unicode of embedded STRING2 data type. Max 100 bytes (50 unicode characters)	Blank
17[1]	Read/Write	Assigned_Geographic Description	STRINGI	User assigned location	2-byte unicode of embedded STRING2 data type. Max 64 bytes (32 unicode characters)	Blank

(1) Non-volatile
(2) Volatile

Services

Service Code	Implemented for:		Service Name
	Class	Instance	
0x01	Yes	Yes	Get_Attribute_All
0x05	Yes	Yes	Reset
0x0E	Yes	Yes	Get_Attribute_Single
0x10	Yes	Yes	Set_Attribute_Single
0x18	Yes	Yes	Get_Member

Wall-Clock Time Object Class Code: 0x8B (8Bh)

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Revision	UINT	Revision of this object	03	0x03
2	Read Only	Max Instance	UINT	Largest instance number of an object currently created in this class level of the device	01	0x01
3	Read Only	Instances	UDINT	Number of instances	01	0x01

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
5(1)	Read/Write	Date and Time (Local Time)	DINT[7] – Array of seven DINT	Current adjusted Local time in human readable format	DINT[0] – year DINT[1] – month DINT[2] – day DINT[3] – hour DINT[4] – minute DINT[5] – second DINT[6] – µsec	
6(1)	Read/Write	Current UTC Value (UTC Time)	LINT	Current value of Wall Clock Time	64-bit µS value referenced from 0000 hrs, January 1, 1970	
7(1)	Read/Write	UTC Date and Time (UTC Time)	DINT[7] – Array of seven DINT	Current time in human readable format	DINT[0] – year DINT[1] – month DINT[2] – day DINT[3] – hour DINT[4] – minute DINT[5] – second DINT[6] – µsec	
8(1)	Read Only	Time Zone String	Struct of:	This string specifies the time zone where the controller is located, and the adjustment in hours and minutes applied to the UTC value to generate the Local time value	Time Zone String can be specified in the following formats: GMT+hh:mm <location> GMT-hh:mm <location> hh:mm portion is used internally to calculate the local time, and the <location portion> is used to describe the time zone and is optional. Length of the Data array can be from 0...82. Example: GMT-05:00 Eastern Time GMT+01:00 Greenwich Meantime	
			UDINT	LENGTH		
			SINT [LENGTH]	DATA		
9(1)	Read Only	DST Adjustment	INT	Daylight Saving Time adjustment	The number of minutes to be adjusted for daylight saving time (Not configurable, always 60 minutes).	
10(1)	Read/Write	Enable DST	USINT	It specifies if daylight saving time is required. Needs user to set manually	0 = False, do not apply daylight saving time adjustment 1 = True, apply daylight saving time adjustment	
11(1)	Read/Write	Current Value (Local Time)	LINT	Adjusted Local value of Wall Clock Time	64-bit µS value referenced from 0000 hrs, January 1, 1970	

Services

Service Code	Implemented for:		Service Name
	Class	Instance	
0x01	Yes	Yes	Get_Attribute_All
0x0E	Yes	Yes	Get_Attribute_Single
0x10	Yes	Yes	Set_Attribute_Single

Modbus Serial Link Object

Class Code: 0x46 (46h)

Modbus Serial Link Object is used for configuration of Modbus RTU serial data communication. It includes link-specific counters and status information for the port. Each instance represents the client portion of the Modbus Serial channel.

Instance number shall correspond to respective Port (serial) instance number.

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Class Revision	UINT	Class Revision		1
2	Read Only	Max Instance	UINT	Maximum instance number		
3	Read Only	Num Instances	UINT	Number of ports currently instantiated		

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Link Status	Struct of:		See CIP specifications, Volume 7	
		Status Values	DWORD	Link Status		
		Status Mask	DWORD	Link Status supported		
2	Read Only	Capabilities	Struct of:		See CIP specifications, Volume 7	
		Control Types Supported	DWORD	Control Types supported		0x01
		Data Rates Supported	DWORD	Data Rates supported		0x35FA
		Parity Supported	BYTE	Parity supported		0x07
		Data Bits Supported	BYTE	Data Bits supported	Bit 0 = 7 bit (Not supported) Bit 1 = 8 bit Bit 3...7 = Reserved	0x02
		Stop Bits Supported	BYTE	Stop Bits supported		0x03
		Media Supported	BYTE	Media supported		0x13

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
3	Read/Write	Characteristics	Struct of:			
		Control Types	DWORD	Link Control	Bit 0 = Port Enable (Default) Bit 1 = Enable Wait for CTS Bit 2 = RS485 Bias Enable Bit 3= RS485 Termination Enable Bit 4...31 = Reserved	
		Data Rates	UDINT	Data Rates	Exact data rate: 1200 2400 9600 19200 – Micro830 and Micro850 controllers 38400 – Micro820 controllers	
		Parity	USINT	Parity	1 = Even 2 = None (Default) 3 = Odd	
		Data Bits	USINT	Data Bits	7 = 7 bits (Not supported) 8 = 8 bits (Default)	
		Stop Bits	USINT	Stop Bits	1 = 1 bit (Default) 2 = 2 bits	
		Media	USINT	Electrical form	1 = RS232 – DTR, RTS should be fixed high/asserted. DSR/CTS/DCD ignored. (Default) 2 = RS232 – RTS/CTS radio modem support. DTR should be fixed high/asserted. DSR/DCD ignored. 3 = RS422 (Not supported) 4 = RS485 – TX-pair, tri-stated, RX-pair fixed active (Not supported) 5 = RS485 – 3-wire, tri-stated	
		Protocol	USINT	Modbus Serial protocol	1 = Modbus/RTU (Default) 2 = Modbus ASCII (Not supported)	
4	Read/Write	Unit Id Range	Struct of:	Provides a range of valid Unit Id values for the network		
		Min Unit Id	USINT			0
		Max Unit Id	USINT			255
5	Read/Write	Response Timer	UDINT	Minimum amount of time Modbus client waits for a response from the server before a new transmission can begin. This value is independent of the CIP timeout mechanism	Timer value in milliseconds	200 ms
6	Read/Write	Delay	Struct of:			
		Inter-Frame Delay	UDINT	Line turn delay (RTU only)	Delay in microseconds [0...(2^{32}-1)]	5000
		RTS Pre-Delay	UDINT	Pause Data Tx after RTS rise	Delay in microseconds [0...(2^{32}-1)]	
		RTS Post-Delay	UDINT	Hold RTS after Tx ends	Delay in microseconds [0...(2^{32}-1)]	
		Broadcast Pause	UDINT	Delay after Broadcast	Delay in milliseconds [0...(2^{32}-1)]	200 ms
		Gap	UDINT	Inter-character timeout (RTU only)	Timeout in microseconds [0...(2^{32}-1)]	0

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
7	Read Only	Link counters	Struct of:			
		Characters Received	UDINT	Total number of characters received on the serial link		0
		Frames Received	UDINT	Message frames received		0
		Characters Sent	UDINT	Total number of characters sent on the serial link		0
		Frames Sent	UDINT	Message frames sent		0
		Good Transactions	UDINT	Good CRC, good context, no exception		0
		Broadcasts	UDINT	Broadcast sent		0
		Good Exceptions	UDINt	Good CRC, Server Exception		0
		Mismatch Errors	UDINT	Good CRC, Mismatched Response		0
		Bad CRC	UDINT	Bad CRC		0
		No Response	UDINT	No bytes returned		0
		Other Errors	UDINt	HW/vendor specific		0

Services

	Implemented for:		
Service Code	Class	Instance	Service Name
0x01	Yes	Yes	Get_Attribute_All
0x05	No	Yes	Reset
0x0E	Yes	Yes	Get_Attribute_Single
0x10	Yes	Yes	Set_Attribute_Single
0x4B	Yes	Yes	Get_and_Clear(1)

(1) Gets and clears Link counter and Performance attributes only.

TCP/IP Object

Class Code: 0xF5 (F5h)

The TCP/IP Interface Object provides the mechanism to configure a device's TCP/IP network interface. Examples of configurable items include the device's IP Address, Network Mask, and Gateway Address.

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Revision	UINT	Revision	See CIP specifications	4
2	Read Only	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device	See CIP specifications	1
3	Read Only	Number of Instances	UINT	Number of object instances currently created at this class level of the device	See CIP specifications	1

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Status	DWORD	Interface status	See TCP/IP Object: Status Instance Attribute table	NA
2	Read Only	Configuration Capability	DWORD	Interface capability flags	See TCP/IP Object: Configuration Capability Attribute table	0x94
3	Read/Write	Configuration Control	DWORD	Interface control flags	Bit map of control flags. See TCP/IP Object: Configuration Control Attribute table	0x02
4	Read Only	Physical Link Object	Struct of:	Path to physical link object		
		Path Size	UINT	Size of path	See CIP specifications	[02]
		Path	Padded EPATH	Logical segments identifying the physical link object		[20][F6][24][01]
5	Read/Write	Interface Configuration	Stuct of:	TCP/IP network interface configuration		
		IP Address	UDINT	The device's IP address	See CIP specifications	NA
		Network Mask	UDINT	The device's network mask		NA
		Gateway Address	UDINT	Default gateway address		NA
		Name Server	UDINT	Primary name server		NA
		Name Server 2	UDINT	Secondary name server		NA
		Domain Name	STRING	Default domain name		NA
6	Read/Write	Host Name	STRING	Host name		NA
10	Read/Write	SelectAcd	BOOL	Activates the use of ACD	0 = Disable ACD (Default) 1 = Enable ACD When ACD is disabled, controller continues to detect conflict and will persistently defend the IP in use. Last conflict will not be updated, Ethernet State will not fault because of this, and IP address will not be released due to conflict	

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
11	Read/Write	Last ConflictDetected	Struct of:	Structure containing information related to last conflict detected	See CIP specifications	
		AcdActivity	USINT	State of ACD activity when last conflict detected		NA
		RemoteMAC	Array of 6 USINT	MAC address of remote node from the ARP PDU in which a conflict was detected		NA
		ArpPdu	Array of 28 USINT	Copy of the raw ARP PDU in which a conflict was detected		NA
13	Read/Write	Encapsulation Inactivity Timeout	UINT		Number of seconds of inactivity before TCP connection 0 = Disable 1...3600 = Timeout in seconds	120

Status Instance Attribute

Bit	Called	Definition
0...3	Interface Configuration Status	Indicates the status of the Interface Configuration attribute 0 = The Interface Configuration attribute has not been configured or IP address is Link Local. For example 169.254.x.x or Configured static IP address is duplicate over network (see bit 6, ACD status). 1 = The Interface Configuration attribute contains valid configuration obtained from BOOTP, DHCP or nonvolatile storage. 2 = The Interface Configuration attribute contains valid configuration obtained from hardware settings (pushwheel, thumbwheel, and so on). 3...15 = Reserved for future use.
4	Mcast Pending	Indicates a pending configuration change in the TTL Value and/or Mcast Config attributes. this bit shall be set when either the TTL Value of Mcast Config attribute is set, and shall be cleared the next time the device starts.
5	Interface Configuration Pending	This bit shall be 1 (TRUE) when Interface Configuration attribute is set and the device requires a reset in order for the configuration change to take effect (as indicated in the Configuration Capability attribute). The intent of the Interface Config Pending bit is to allow client software to detect that a device's IP configuration has changed, but will not take effect until the device is reset.
6	AcdStatus	0 = No Address Conflict Detected 1 = Address Conflict Detected
7	AcdFault	Follows AcdStatus value
8...31	Reserved	Reserved for future use and shall be set to zero.

Configuration Capability Attribute

Bit	Called	Definition
0	BOOTP Client	1 = The device is capable of obtaining its network configuration using BOOTP.
1	DNS Client	1 = The device is capable of resolving host names by querying a DNS server.
2	DHCP client	1 = The device is capable of obtaining its network configuration using DHCP.
3	DHCP-DNS Update	1 = The device is capable of sending its host name in the DHCP request as documented in the Internet-draft, "Interaction between DHCP and DNS", version 12.
4	Configuration Settable	1 = The Interface Configuration attribute is settable. Some devices (for example a PC or workstation) may not allow the Interface Configuration to be set using the TCP/IP Interface Object.
6	Interface Configuration Change Requires Reset	0 = A change in the Interface Configuration attribute will take effect immediately. 1 = The device requires a restart in order for a change to the Interface Configuration attribute to take effect.
7	AcdCapable	1 = The device is ACD capable.
8...31	Reserved	Reserved for future use and shall be set to zero.

Configuration Control Attribute

Bit	Called	Definition
0...3	Startup Configuration	Determines how the device shall obtain its initial configuration at start up. 0 = The device shall use the interface configuration values previously stored (for example, in the non-volatile memory or hardware switches, and so on). 1 = The device shall obtain its interface configuration values using BOOTP. 2 = The device shall obtain its interface configuration values using DHCP upon start-up. 3...15 = Reserved for future use
4	DNS Enable	1 = The device shall resolve host names by querying a DNS server.
5...31	Reserved	Reserved for future use and shall be set to zero.

Services

	Implemented for:		
Service Code	Class	Instance	Service Name
0x01	No	Yes	Get_Attribute_All
0x0E	Yes	Yes	Get_Attribute_Single
0x10	No	Yes	Set_Attribute_Single

Ethernet Link Object

Class Code: 0xF6 (F6h)

The Ethernet Link Object maintains link-specific counters and status information for an IEEE 802.3 communications interface.

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Revision	UINT	Revision of this object	See CIP specifications	4
2	Read Only	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device	See CIP specifications	1
3	Read Only	Number of Instances	UINT	Number of object instances currently created at this class level of the device	See CIP specifications	1

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1	Read Only	Interface Speed	UDINT	Interface speed currently in use. Speed in Mbps	0, 10, 100, 1000	NA
2	Read Only	Interface Flags	DWORD	Interface status flags	See Interface Flags on page 57	NA
3	Read Only	Physical Address	Array of 6 USINT	MAC layer address	See CIP specifications Volume 2	
4	Read Only	Interface Counters	Struct of:		See CIP specifications Volume 2	
		In Octets	UDINT	Octets received on the interface		NA
		In Ucast Packets	UDINT	Unicast packets received on the interface		NA
		In NUcast Packets	UDINT	Non-unicast packets received on the interface		NA
		In Discards	UDINT	Inbound packets received on the interface but discarded		NA
		In Errors	UDINT	Inbound packets that contain errors (does not include In Discards)		NA
		In Unknown Protos	UDINT	Inbound packets with unknown protocol		NA
		Out Octets	UDINT	Octets sent on the interface		NA
		Out Ucast Packets	UDINT	Unicast packets sent on the interface		NA
		Out NUcast Packets	UDINT	Non-unicast packets sent on the interface		NA
		Out Discards	UDINT	Outbound packets discarded		NA
		Out Errors	UDINT	Outbound packets that contain errors		NA

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
5	Read Only	Media Counters	Struct of:	Media-specific counters	See CIP specifications Volume 2	
		Alignment Errors	UDINT	Frames received that are not an integral number of octets in length		NA
		FCS Errors	UDINT	Frames received that do no pass the FCS check		NA
		Single Collisions	UDINT	Successfully transmitted frames which experienced exactly one collision		NA
		Multiple Collisions	UDINT	Successfully transmitted frames which experienced more than one collision		NA
		SQE Test Errors	UDINT	Number of times SQE test error message is generated		NA
		Deferred Transmissions	UDINT	Frames for which first transmission attempt is delayed because the medium is busy		NA
		Late Collisions	UDINT	Number of times a collision is detected later than 512 bit-times into the transmission of a packet		NA
		Excessive Collisions	UDINT	Frames for which transmission fails due to excessive collisions		NA
		MAC Transmit Errors	UDINT	Frames for which transmission fails due to an internal MAC sublayer transmit error		NA
		Carrier Sense Errors	UDINT	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame		NA
		Frame Too Long	UDINT	Frames received that exceed the maximum permitted frame size		NA
		MAC Receive Errors	UDINT	Frames for which reception on an interface fails due to an internal MAC sublayer receive error		NA
6	Read/Write	Interface Control	Struct of:	Configuration for physical interface		
		Control Bits	WORD	Interface Control Bits	See Interface Control Bits on page 57	0x0001
		Forced Interface Speed	UINT	Speed at which interface shall be forced to operate	Speed in Mbps (10, 100, 1000)	0x0000
7	Read Only	Interface Type	USINT	Type of interface – twisted pair, fiber, internal, and so on	See CIP specifications Volume 2	2
8	Read Only	Interface State	USINT	Current state of the interface – operational, disabled, and so on	See CIP specifications Volume 2	NA
9	Read/Write	Admin State	USINT	Administrative state – enable, disable	See CIP specifications Volume 2	0x01
11	Read Only	Interface Capability	Struct of:	Indication of capabilities of the interface		
		Capability Bits	DWORD	Interface capabilities, other than speed/duplex	Bit map	
		Speed/Duplex Options	Struct of:	Indicates speed/duplex pairs supported in the Interface Control attribute		
			USINT	Speed/Duplex Array Count	Number of elements	
			Array of Struct of:	Speed/Duplex Array		
			UINT	Interface Speed	Semantics are the same as the Forced Interface Speed in the Interface Control attribute: speed in Mbps.	
			USINT	Inferface Duplex Mode	0 = Half-duplex 1 = Full duplex 2...255 = Reserved	

Interface Flags

Bit	Called	Definition
0	Link Status	Indicates whether or not the IEEE 802.3 communications interface is connected to an active network. 0 = Active link 1 = Inactive link The determination of link status is implementation specific. In some cases, device can tell whether the link is active through hardware/driver support. In other cases, the device can only tell whether the link is active by the presence of incoming packets.
1	Half/Full Duplex	Indicates the duplex mode currently in use. 0 = Half duplex 1 = Full duplex If the Link Status flag = 0, then the value of the Half/Full Duplex flag is indeterminate.
2...4	Negotiation Status	Indicates the status of link auto-negotiation. 0 = Auto-negotiation in progress 1 = Auto-negotiation and speed detection failed. Using default values for speed and duplex. Default values are product dependent. Recommended defaults are 10 Mbps and half duplex. 2 = Auto-negotiation failed but detected speed. Duplex was defaulted. Default value is product depended. Recommended default is half duplex. 3 = Successfully negotiated speed and duplex. 4 = Auto-negotiation not attempted. Forced speed and duplex.
5	Manual Setting Requires Reset	0 = Indicates the interface can activate changes to link parameters (auto-negotiate, duplex mode, interface speed) automatically. 1 = Indicates the device requires a Reset service be issued to its Identity Object in order for the changes to take effect.
6	Local Hardware Fault	0 = Indicates the interface detects no local hardware 1 = Indicates a local hardware fault is detected The meaning of this is product specific. Examples are AUI/MII interface detects no transceiver attached or a radio modem detects no antennae attached. In contrast to the soft, possible self-correcting nature of the Link Status being inactive, this is assumed a hard fault requiring user intervention.
5...31	Reserved	Shall be set to zero

Interface Control Bits

Bit	Called	Definition
0	Auto-negotiate	0 = Indicates 802.3 link auto-negotiation is disabled 1 = Indicates auto-negotiation is enabled If auto-negotiation is disabled, then the device shall use the settings indicated by the Forced Duplex Mode and Forced Interface Speed bits.
1	Forced Duplex Mode	If the Auto-negotiate bits is zero, the Forced Duplex Mode bit indicates whether the interface shall operate in full or half duplex mode. 0 = Indicates the interface duplex should be half duplex 1 = Indicates the interface duplex should be full duplex Interfaces not supporting the requested duplex shall return a GRC hex 09h (Invalid Attribute Value). If auto-negotiation is enabled, attempting to set the Forced Duplex Mode bit shall result in a GRC hex 0Ch (Object State Conflict).
2...15	Reserved	Shall be set to zero

Services

Service Code	Implemented for:		Service Name
	Class	Instance	
0x01	No	Yes	Get_Attribute_All
0x0E	Yes	Yes	Get_Attribute_Single
0x10	No	Yes	Set_Attribute_Single
0x4C	No	Yes	Get_and_Clear

USB Object

Class Code: 0x33A (33Ah)

For Micro830 and Micro850 controllers only.

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1(1)	Read Only	Revision	UINT	Revision of this object	1	0x01
2(1)	Read Only	Max Instance	UINT	Maximum instance number of an object currently created in this class level of the device		
3(1)	Read Only	Number of Instances	UINT	Number of instances in this class hierarchy		

(1) Non-volatile.

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
1(1)	Read Only	Logical Address	USINT	Logical address of this instance	0x01...0x7F 0x80 – 0xFE invalid See Semantics	0xFF
2(1)	Read Only	State	USINT	State of this instance	0 = Initializing 1 = Fault 2 = Initialized 3 = Configured 4 = Ready 5...0xFF = Reserved. See Semantics	
3(1)	Read Only	Suspended	BOOL	USB Suspend State of this instance	0 = Not suspend (Default) 1 = Suspend See Semantics	
4(1)	Read/Write	Disabled	BOOL	Enable/Disable state of this instance	0 = Enabled (Default) 1 = Disabled See Semantics	
5(2)	Read Only	Bus_Speed	USINT	The highest USB bus speed this instance supports	0 = Low speed 1 = Full speed (Default) 2 = High speed 3...255 = Reserved	
6(2	Read Only	OTG_Support	BOOL	USB On-The-Go capability support	0 = No (Default) 1 = Yes	

Attribute ID	Access Rule	Name	Data Type	Description	Values	Defaults
7[1]	Read Only	Mode	USINT	The current mode of this instance	0 = Slave mode (Default) 1 = Host mode 2...255 = Reserved	
8[1]	Read Only	CIP_Interface_Counter _Length	USINT	The size of the Interface Counters structure in bytes	24 = Default All other values are reserved. See Semantics	
9[1]	Read Only	CIP_Interface_Counters	Struct of:	Number of USBCIP transfers received and sent by this instance	See Semantics	
		Rx_Good_Counter	UDINT	Total number of good USBCIP transfers received	0...0xFFFFFFFF	
		Rx_Bad_Counter	UDINT	Total number of bad USBCIP transfers received	0...0xFFFFFFFF	
		Rx_Dropped_Counter	UDINT	Total number of dropped USBCIP transfers	0...0xFFFFFFFF	
		Tx_Good_Counter	UDINT	Total number of USBCIP transfers sent	0...0xFFFFFFFF	
		Tx_Retry_Counter	UDINT	Total number of USBCIP transfers retried	0...0xFFFFFFFF	
		Tx_Dropped_Counter	UDINT	Total number of USBCIP transfers dropped	0...0xFFFFFFFF	
10[1]	Read Only	USB Media Counters	Struct of:	Media-specific counters	See Semantics	
		Type	UINT	The predefined type of Media Counters structure	Type 0 = Undefined (Default) Type 1 = Intel IXP465 2...0xFFFF = Reserved	
		Length	UINT	The size of the Media Counters that follows in bytes	0...0xFFFF	0xFFFF
		Diagnostic_Counters	Variable	Hardware specific diagnostic counters	See Semantics	

(1) Volatile.
(2) Non-volatile.

Services

Service Code	Implemented for:		Service Name
	Class	Instance	
0x01	No	Yes	Get_Attribute_All
0x0E	Yes	Yes	Get_Attribute_Single
0x10	Yes	Yes	Set_Attribute_Single
0x4B	Yes	Yes	Reset_Counters

Symbol Object

Class Code: 0x6B (6Bh)

This object services is available over all CIP communication ports (Ethernet, Serial and USB).

Class Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Micro800 Controller Reporting Value
1	Get	Revision	UINT	Revision of object class definition	4
2	Get	Max Instance	UINT	Maximum instance number	Maximum instance number in Symbol table.(2)
3	Get	Num Instances	UINT	Number of symbols currently instantiated	Number of Symbols in Symbol table.(2)
8	Set(1)	Symbol UID	UDINT	Software identifier for symbol collection	Unique value that changes with any changes in Symbol table (Symbol table's CRC value is used for this value).

(1) These attributes are not settable for Micro800 controllers.

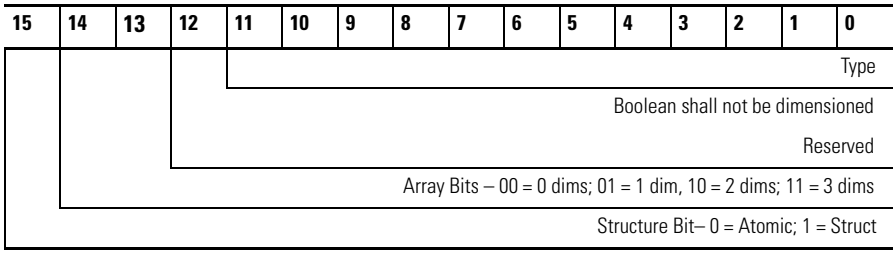
(2) The symbol list will have only the instances of supported data types. Class attribute 2 and 3 corresponds to the actual number of symbols in the system's Symbol table which is superset of symbol list used for symbol object.

Instance Attribute

Attribute ID	Access Rule	Name	Data Type	Description	Micro800 Controller Reporting Value
1	Set(1)	Symbol Name	STRING	Name of the Symbol. Up to 254 byte character string	Symbol name
2	Set(1)	Symbol Type	WORD	Data type of the Symbol	See Data type format below
3	Set(1)	Symbol Address	UDINT	Physical address of Symbol data or object	Symbol memory offset
5	Get	Symbol Object Address	UDINT	Physical address of the Symbol object	Symbol table memory offset
6	Set(1)	Software Control	UDINT	32-bits for storage for whatever	0
7	Get	Element Size	UINT	Size of the Symbol element in bytes	Symbol element size
8	Get	Array Dimensions	Struct of: UDINT UDINT UDINT	Dimension of the array	Symbol array dimension
9	Get	Safety Flag	BOOL	The Symbol references a safety object or safety data	False (Target object is standard data)
10	Set(1)	PPD Control	BYTE	Public Private Data (PPD) access rules Bit 0...1: 0 = Full access 1 = Reserved 2 = Read only 3 = No access	0 or 2
11	Set(1)	CONSTANT TAG INDICATOR	BYTE	Indicates that the tag is marked as a constant tag for data protection 0 = Normal tag 1 = Constant value tag	0

(1) These attributes are not settable for Micro800 controllers.

Data Type Format



Services

Service Code	Implemented for:		Service Name
	Class	Instance	
0x01	NA	–	Get_Attribute_All
0x03	–	–	Get_Attribute_List
0x04	–	–	Set_Attribute_List
0x08	–	–	Create Object
0x09	NA	–	Delete Object
0x0E	Supported(1)	–	Get_Attribute_Single
0x4B	–	–	Get_Instance_List
0x55	–	Supported	Get_Instance_Attribute_List

(1) This service is intended to read the 'Software UID' attribute value.

Notes:

CIP Error Message Codes

CIP Error Message Codes

Error ID	Description		
33	Parameter configuration errors.		
	Sub Error ID	*Description*	*Corrective Actions*
	32	Bad channel number.	Provide a valid channel number.
	36	Unsupported CIP connection type.	Provide one of the valid CIP connection type: 0 = UCMM; 1 = Class3 Connection
	40	Unsupported CIP symbolic data type.	Provide a valid CIP symbolic data type. See CIP Messages Data Type on page 65 for a list of supported data types.
	41	Invalid CIP symbol name.	See SymbolicCfg on page 36 for correct CIP Symbol Syntax.
	42	Unsupported CIP class value or MemberID count.	CIP Class value should be greater than zero, or MemberID count should be greater than three.
	48	Function block's input data array size is not sufficient.	Verify data buffer size is more than requested data length.
	49	Invalid target path.	See TargetCfg on page 37 for correct target path setting.
	50	Bad service code.	MSG_CIPGENERIC: Valid service code is 1...127 MSG_CIPSYMBOLIC: Supported service code is Read (0) and Write (1).
	51	Function block's transmit data array size is too big for CIP communication.	
	52	Bad Segment type value.	
	53	Bad UCMM timeout value. If the encapsulation timeout value is less than UCMM timeout, or difference between encapsulation timeout and UCMM timeout is less than or equal to one second, this error will occur.	
	54	Bad connected timeout value. If the encapsulation timeout value is less than CONNECTED message timeout, or difference between encapsulation timeout and CONNECTED message timeout is less than or equal to one second, this error will occur.	

CIP Error Message Codes

Error ID	Description		
55	Timeout errors		
	Sub Error ID	*Description*	
	112	Message timed out while waiting in message wait queue.	Per port at a time, four CIP client control blocks are available. Verify that per port only four function blocks are executing, otherwise this error may occur.
	113	Message timed out while waiting for link layer/connection establishment.	• Verify the cable is working properly and the target exists. • Controller supports maximum eight CIP client message processing. If the total number of MSG_CIPSYMBOLIC and MSG_CIPGENERIC instructions executing exceeds eight, this error may occur.
	114	Message timed out while waiting for transmit to link layer.	Controller supports maximum eight CIP client message processing. If the total number of MSG_CIPSYMBOLIC and MSG_CIPGENERIC instructions executing exceeds eight, this error may occur.
	115	Message timed out while waiting for response from link layer.	Verify the cable is working properly and the target exists.
69	Server response format error codes.		
	Sub Error ID	*Description*	
	65	Message reply not matching with request.	
	68	Message reply data type not valid/supported (MSG_CIPSYMBOLIC).	
208	No IP address configured for the network.		Configure a valid IP address.
209	Maximum number of connections used - no connections available.		Reduce the number of active connections.
210	Invalid internet address or node address.		Verify that target IP address is not zero, broadcast, multicast, or loopback.
217	Message execution was cancelled by user. Cancel parameter was set to TRUE.		Cancel bit should not be set and RMD operation is not performed.
218	No network buffer space available.		Firmware has limited combuffers and CIP requests Verify there is not too much communication activity.
222	Reserved.		
223	Link address not available (TCP/IP or Ethernet configuration change is in progress).		Configuration change is in progress due to user changing Ethernet or TCP/IP configuration. Wait for configuration to be completed then try again.
224	CIP response error code. SubErrorID specifies the CIP status and ExtErrorID specifies the CIP extended status value. Refer to CIP specifications for possible error code values.		
255	Channel is in shutdown or reconfiguration is in progress.		Wait for configuration to be completed then try again.

CIP Message Data Types

CIP Messages Data Type

Data Type[1]	Description
BOOL	Logical Boolean with values TRUE(1) and FALSE(0) (Uses up 8 bits of memory)
SINT	Signed 8-bit integer value
INT	Signed 16-bit integer value
DINT	Signed 32-bit integer value
LINT[2]	Signed 64-bit integer value
USINT	Unsigned 8-bit integer value
UINT	Unsigned 16-bit integer value
UDINT	Unsigned 32-bit integer value
ULINT[2]	Unsigned 64-bit integer value
REAL	32-bit floating point value
LREAL[2]	64-bit floating point value
STRING	Character string (1 byte per character)
DATE[3]	Unsigned 32-bit integer value
TIME[3]	Unsigned 32-bit integer value

(1) Logix MSG instruction can read/write SINT, INT, DINT, LINT and REAL data types using "CIP Data Table Read" and "CIP Data Table Write" message types.
BOOL, USINT, UINT, UDINT, ULINT, LREAL, STRING, SHORT_STRING, DATE, and TIME data types are not accessible with the Logix MSG instruction.
(2) Not supported in PanelView™ Component or PanelView 800.
(3) Mainly for use with PanelView Plus and PanelView 800 HMI terminals.

Notes:

Programming Tips and Recommendations for Micro800 CIP Messaging Instructions (MSG)

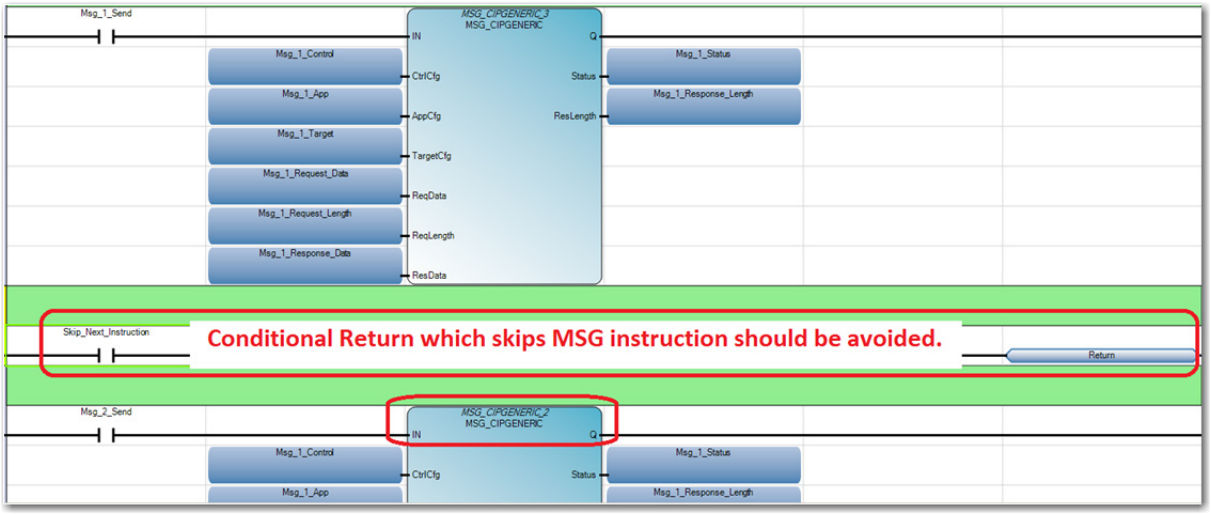
Programming Recommendations for MSG Client Instructions

When programming with the Ethernet MSG instructions (MSG_CIPSYMBOLIC, MSG_CIPGENERIC, MSG_MODBUS2), execute the instructions as part of the normal program scan in order for the message status to be promptly updated. Once a message is initiated it will always complete as Success or Failure, but the instruction only updates its status when executed.

Recommended Practices

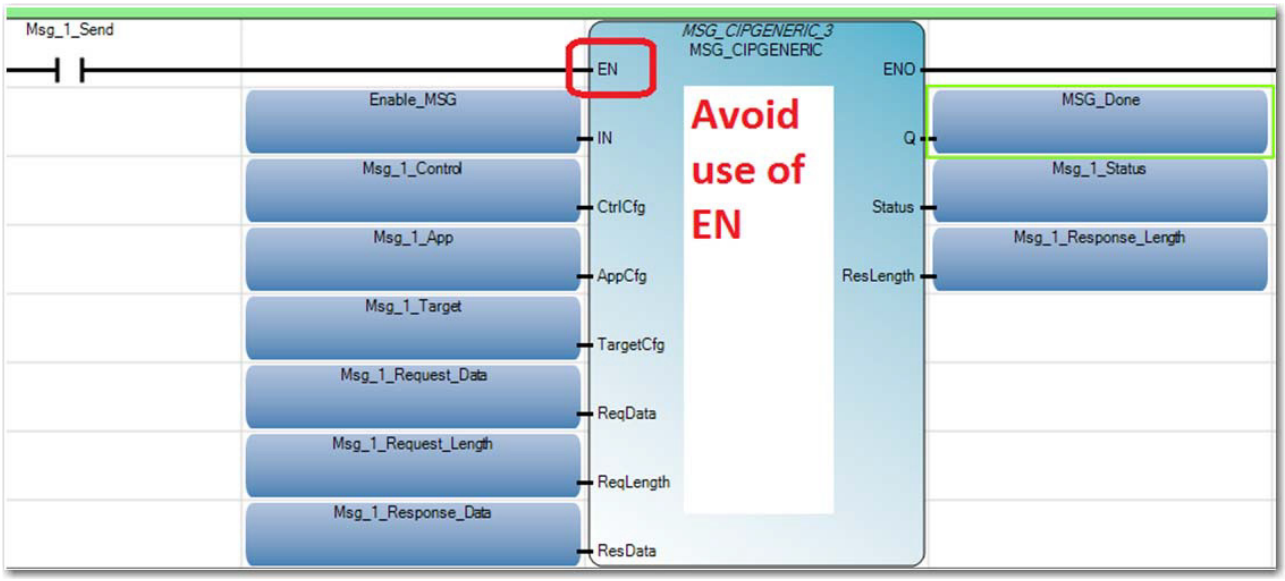
1. Do not place MSG instructions in interrupts (for example, STI) since status bits (for example, message done status) may not be updated every program scan.
2. Do not use MSG instructions with RETURN, TND, JUMP (LD, FBD), IF-THEN (ST only), CASE (ST only) as status bits, (for example, message done status) may not be updated every program scan.

   For example: First MSG instruction is acceptable as it is evaluated as part of the normal program scan. Second MSG instruction should not be used with the Return as the message status may not be updated with correct status if Skip_Next_Instruction becomes TRUE before the message instruction finishes.



3. Do not use MSG instructions EN/ENO as it could leave the MSG instruction unscanned. Normally, without the EN input, the MSG instruction is always scanned even if input is FALSE.

   For example: EN/ENO option should not be used with MSG instruction because EN is an optimization to completely skip over the function block if the EN input is False.



4. Execute only a single message instruction to the same server by using the output status (for example, Q/Done) to trigger the next message instruction. This optimizes communication performance. In case of cable break, this limits the number of message instruction timeouts to the same server.
   Client messages are inserted into a queue when initially executed. Multiple messages to the same server can be queued, but messages are only transmitted one at a time.
   For example, if two message instructions are executed to the same destination server in the same program scan, the first message is sent but the second message is delayed until the first message completes.

5. Do not have the user program take action on initial communication timeouts. Client messages that use Ethernet may fail the first time with error, especially after controller is powered up directly into run mode. This is expected because when the controller powers up or after a change in the Ethernet settings, it takes about 1 to 5 seconds (or more depends on network settings/DHCP delay) for communications to initialize to send the first message. Any client message activated during this period will result in timeout error.

Supported Data Packet Size for CIP Serial Function

For Micro820, Micro830, and Micro850 controllers, both embedded serial port and plug-in serial ports can support CIP serial communication. CIP serial communication data packet includes user data and CIP packet header.

When working as a CIP serial client, Micro820/Micro830/Micro850 serial ports can support a maximum of 490 bytes of read/write user data. This maximum specification applies to CIP serial data packets with a minimum packet header size. When the size of a packet header is bigger than the minimum packet header size, the maximum size of user data that the CIP client can support is less than 490 bytes. If data packet size is greater than the maximum data size supported by the CIP client, the function block reports an error (0x21) and a sub-error (0x33).

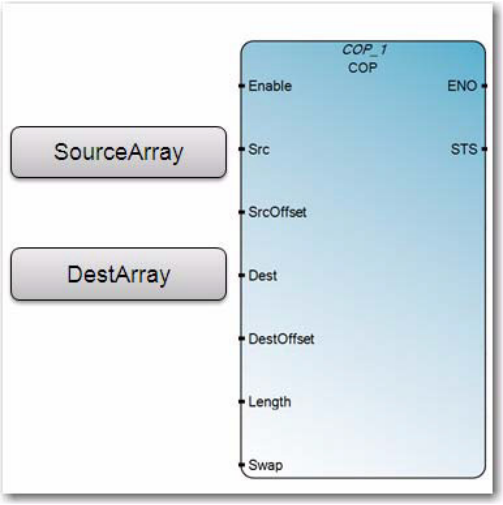
When working as a CIP serial server, Micro820/Micro830/Micro850 serial ports can support a minimum of 255 bytes of read/write user data. This minimum user data size specification applies to CIP serial data packets with maximum packet header size. When the size of CIP packet header is less than the maximum packet header size, the CIP client can support data packet size that is greater than the minimum specification (that is, greater than 255 bytes). However, if user data size is greater than the maximum data size supported by the CIP server function, the CIP data packet could be dropped, and the client will time out.

IMPORTANT For CIP serial server function, it is recommended not to read/write more than 255 bytes of user data in a single CIP message.

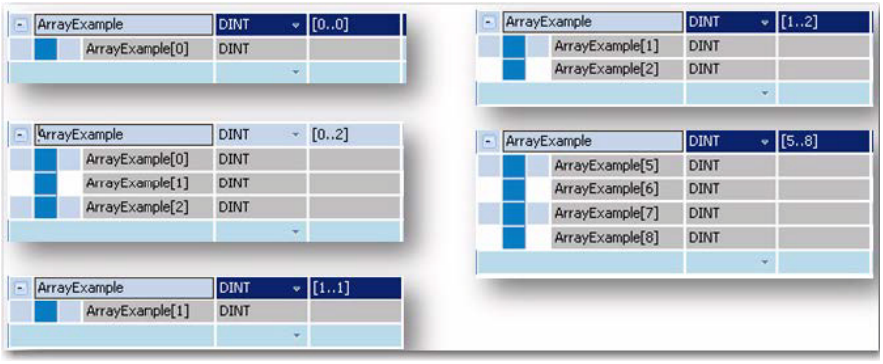
Programming Tips for COP Instruction

The following tips apply when programming with the COP instruction.

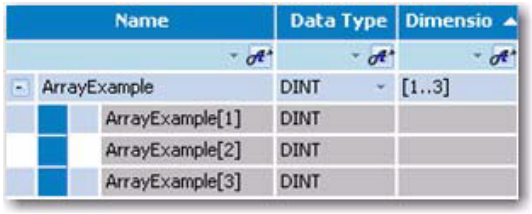
1. Both Source and Destination data has to be in the form of array. If only one variable needs to be copied, define the variable as one element array.



2. Connected Components Workbench flexibly supports Array elements starting from 0 or any non-zero number. Below are some valid examples.

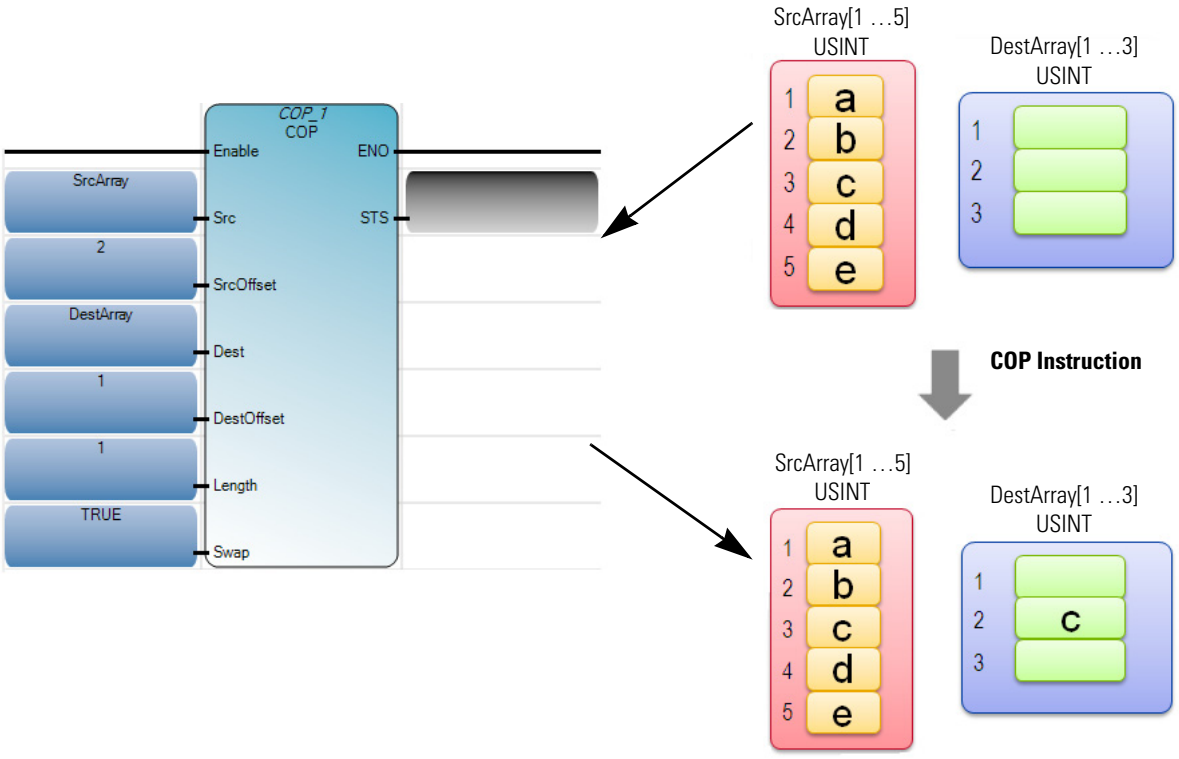


IMPORTANT Array in Connected Components Workbench is defined as **ArrayName[x...y]**, where:
x – Starting element number
y – End element number



3. When copying the source array from the middle of an array, use SrcOffset to specify how many elements you need to offset. For example, if the Source array is SrcArray[1..5] and SrcOffset is 2, then the first element to be copied to destination is SrcArray[3].

4. To store copied data to the Destination array, start at the middle of the array. Use DestOffset to specify how many elements you need to offset. For example, if the Destination array is DestArray[1..3] and DestOffset is 1, then the first location to store copied data is DestArray[2].



Notes:

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/.

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

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Rockwell Automation Publication 2080-QS002B-EN-E - November 2016
Supersedes Publication 2080-QS002A-EN-E - April 2013