

Success: This answer has been added to your Favorites

ID: QA18224 | Access Levels: Everyone

Logix and Micro820/850/870 controller Ethernet/IP Messaging

Document ID QA18224

Published Date 09/23/2022

Summary

Logix and Micro820/850/870 controller Ethernet/IP Messaging

Question

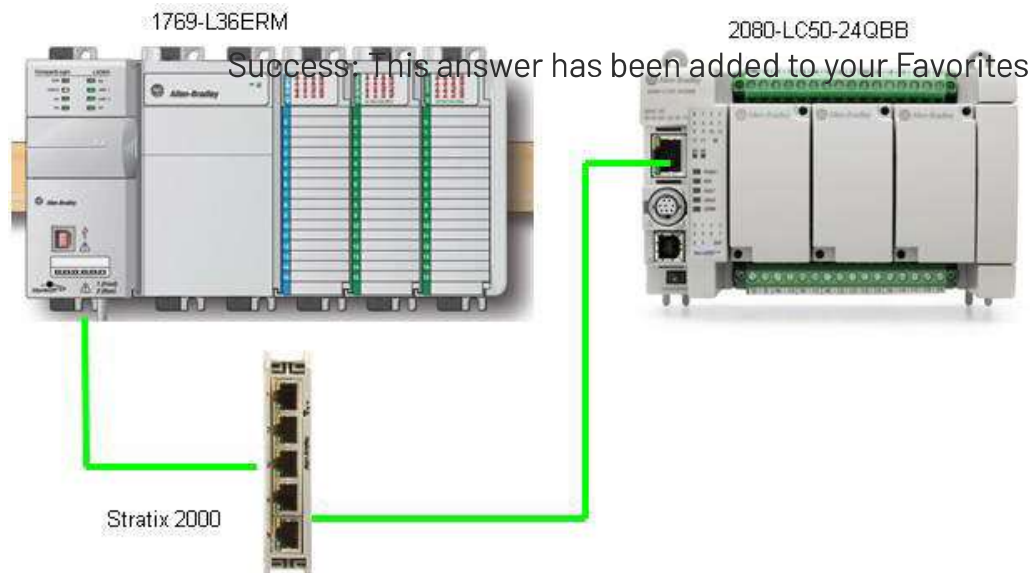
- How does a Logix controller communicate with a Micro820/850 controller over Ethernet/IP?
- Why does a message instruction in ControlLogix to a Micro820/850 have error `16#0005 Class or instance not supported`?
- How to message from Micro800 controller to Logix controller?

Background

This application note details how to create a message from a Logix controller to a Micro850 PLC (global variables only) over Ethernet/IP. If the Micro820/850 tags are not globally defined an error `16#0005 Class or instance not supported` will occur on the message instruction.

Note: Connected Components Workbench 4.0 and above support this.

This application note is based on the following hardware:



Note: It is assumed that the user is familiar with the RSLogix 5000 and CCW programming software, as many of the steps for setup are not shown.

Environment

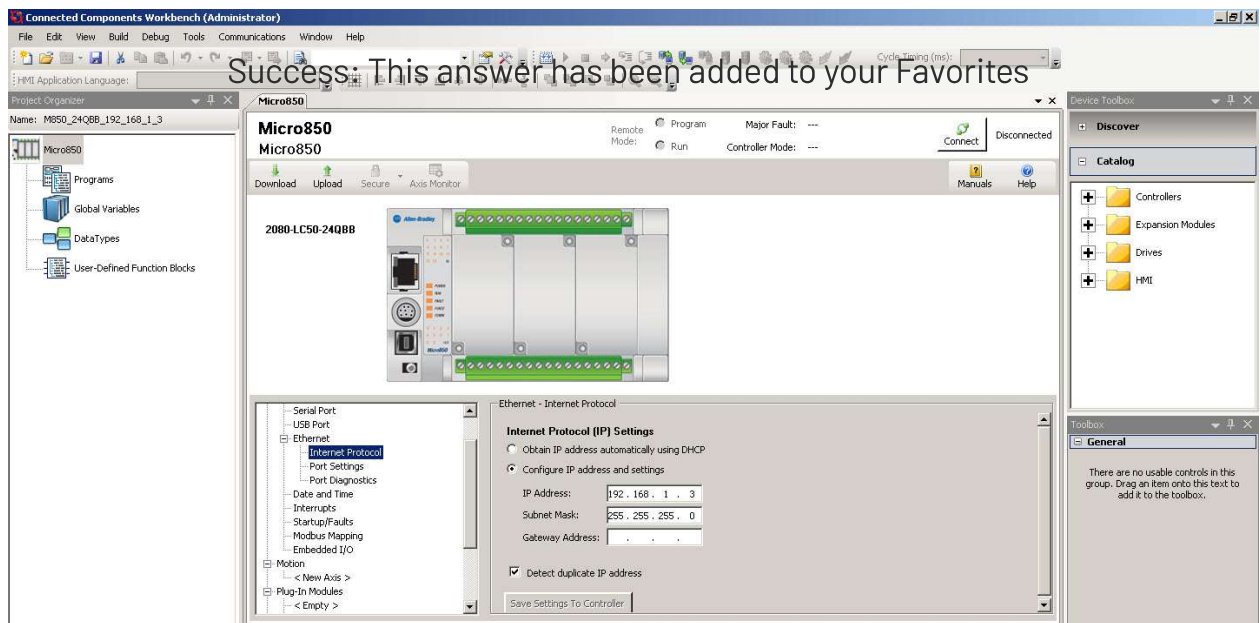
- Micro820 Controller
- Micro850 Controller
- Logix5000 Controllers
- Connected Component Workbench
- RSLogix 5000
- Studio 5000 Logix Designer

Answer

Logix to Micro800 Messaging

In order to communicate with a Micro850 over Ethernet/IP it is necessary to setup the Micro850 with an IP address in the same address range as the CompactLogix.

In this case as our CompactLogix is setup 192.168.1.12, then we need to setup the Micro850 perhaps to 192.168.1.3 with subnet 255.255.255.0.

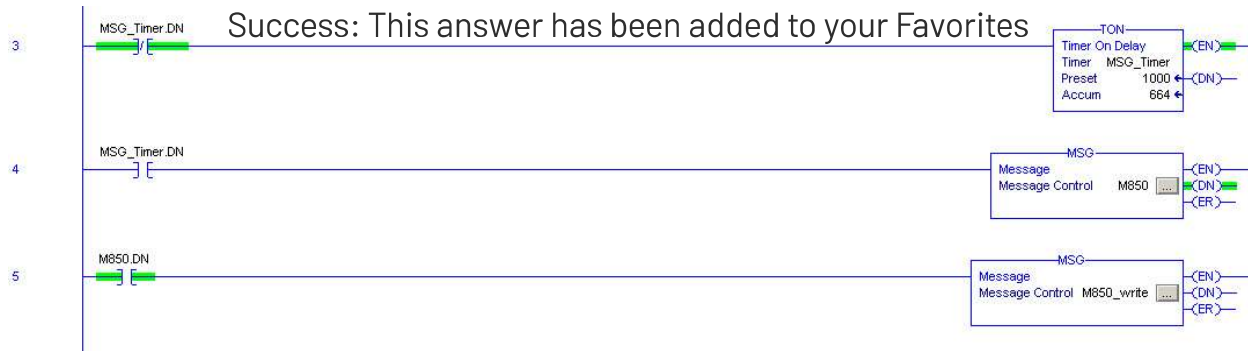


To test the messaging capability, first we need to setup some test variables in the Micro850. Enter two arrays of ten DINT's into the global variables of the M850. So create Test_DINT for the read, and create Test_Writes for the writing of values from the Logix. If the Micro 850 tags are not globally defined an error 16#0005 Class or instance not supported will occur on the message instruction.

Note: that each value of the Test_DINT is provided with an initial value. Now build, download, and go to Debug mode so that you can see the global variables.

Name	Data Type	Dimension	Alias	Initial Value	Attribute
Test_Writes	DINT	[0..9]		...	Read/Write
Test_Writes[0]	DINT			00	Read/Write
Test_Writes[1]	DINT			0	Read/Write
Test_Writes[2]	DINT			0	Read/Write
Test_Writes[3]	DINT			0	Read/Write
Test_Writes[4]	DINT			0	Read/Write
Test_Writes[5]	DINT			0	Read/Write
Test_Writes[6]	DINT			0	Read/Write
Test_Writes[7]	DINT			0	Read/Write
Test_Writes[8]	DINT			0	Read/Write
Test_Writes[9]	DINT			0	Read/Write
Test_DINT	DINT	[0..9]		...	Read/Write
Test_DINT[0]	DINT			1	Read/Write
Test_DINT[1]	DINT			2	Read/Write
Test_DINT[2]	DINT			3	Read/Write
Test_DINT[3]	DINT			4	Read/Write
Test_DINT[4]	DINT			5	Read/Write
Test_DINT[5]	DINT			6	Read/Write
Test_DINT[6]	DINT			7	Read/Write
Test_DINT[7]	DINT			8	Read/Write
Test_DINT[8]	DINT			9	Read/Write
Test_DINT[9]	DINT			10	Read/Write

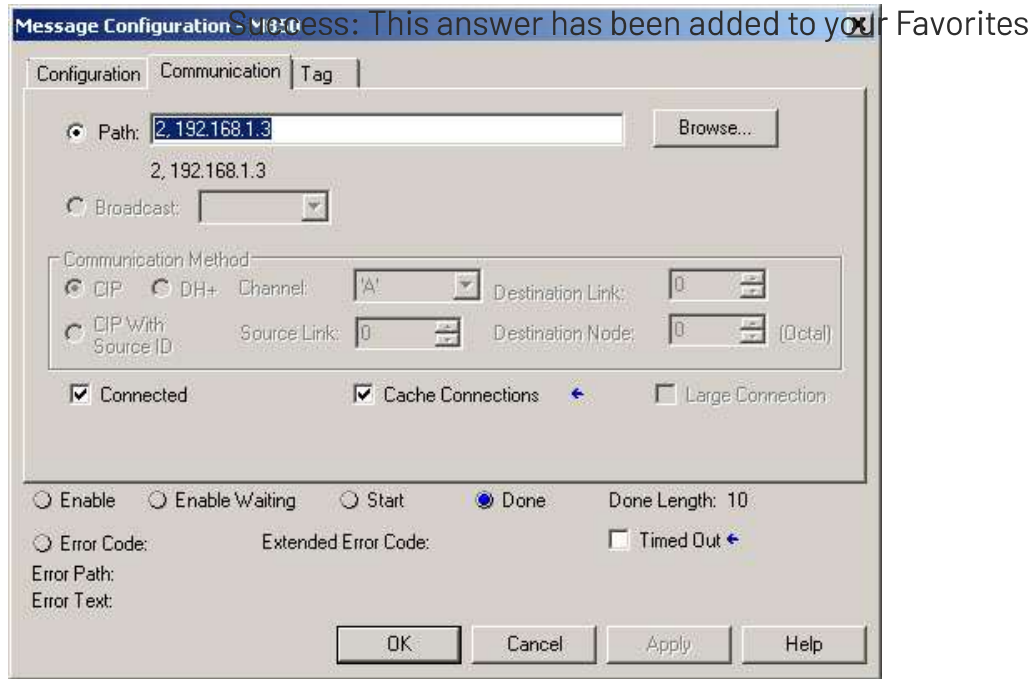
Go to the RSLogix 5000 software and create a project with the following ladder:
Create a new tag for the MSG instructions called M850 and M850_write.



Double click on the properties of the MSG instruction M850 for the Configuration tab. Select a **CIP Data Table Read** and then enter into the **Source Element**, the name of the tag created in the M850. Specify the **Number of Elements**, in this case 10. Then enter the Destination Element where you will send the values. As we have an array of 10 values, so you will need to create a controller scoped array, in this case `DINT_array_Read[10]`.

Now go to the **Communication** tab, to set the **Path**. The message path will depend on the family of controller you are using, the Ethernet port = 2, and then the IP address of the M850 = 192.168.1.3.

Note: For a Micro 820 you may have to use an unconnected message. To do this go to the **Communication** tab and uncheck **Connected**. This was tested in v29 and v30 of Studio 5000.



Double click on the properties of the MSG instruction M850_write for the Configuration tab. Select a **CIP Data Table Write**.

Now create a controller tag in the Logix to hold our write values `DINT_array_Write[10]`. and then enter this into the **Source Element**. Specify the **Number of Elements**, in this case 10. Then enter the Destination Element which is the name of the tag created in the M850 where you will send the values, `Test_Writes`

As before go to the **Communication** tab, to set the **Path**. The Ethernet port = 2, and then the IP address of the M850 = 192.168.1.3

Verify your controller has no errors, download and go online.

Go to the Controller Tags in the Monitor tab, and expand your tags `DINT_array_Read` and to see the actual values.

Expand the tags `DINT_array_Write` and enter values to send to the M850.

Logix Designer - Fund_of_Logix in Test.ACD [1769-L36ERM 21.5] - [Controller Tags - Fund_of_Logix(controller)]

File Edit View Search Logic Communications Tools Window Help

Success: This answer has been added to your Favorites

Rem Run Run Mode Controller OK Energy Storage OK I/O OK

No Forces No Edits

Path: AB_ETHIP:1\192.168.1.12

Controller Organizer

- Controller Fund_of_Logix
 - Controller Tags
 - Controller Fault Handler
 - Power-Up Handler
- Tasks
 - MainTask
 - MainProgram
 - Program Tags
 - MainRoutine
 - Unscheduled Programs / Phases
 - Motion Groups
 - Ungrouped Axes
 - Add-On Instructions
 - Pump_Management
 - Data Types
 - User-Defined
 - Strings
 - Add-On-Defined
 - Predefined
 - Module-Defined
 - Trends
 - I/O Configuration
 - 1769 Bus
 - [0] 1769-L36ERM Fund_of_Logix
 - [1] 1769-IQ16F/A Local_Digital_Inputs
 - [2] 1769-OB16/B Local_Digital_Outputs
 - [3] 1769-IF4XOF2/A Local_Analog
 - Ethernet
 - 1769-L36ERM Fund_of_Logix
 - 1734-AENT/A Distributed_Control_Panel

Scope: Fund_of_Logix Show: All Tags

Name	Value	Force Mask	Style	Data Type
DINT_array_Read	{...}	{...}	Decimal	DINT[10]
DINT_array_Read[0]	1		Decimal	DINT
DINT_array_Read[1]	2		Decimal	DINT
DINT_array_Read[2]	3		Decimal	DINT
DINT_array_Read[3]	4		Decimal	DINT
DINT_array_Read[4]	5		Decimal	DINT
DINT_array_Read[5]	6		Decimal	DINT
DINT_array_Read[6]	7		Decimal	DINT
DINT_array_Read[7]	8		Decimal	DINT
DINT_array_Read[8]	9		Decimal	DINT
DINT_array_Read[9]	10		Decimal	DINT
DINT_array_Write	{...}	{...}	Decimal	DINT[10]
DINT_array_Write[0]	10		Decimal	DINT
DINT_array_Write[1]	11		Decimal	DINT
DINT_array_Write[2]	12		Decimal	DINT
DINT_array_Write[3]	13		Decimal	DINT
DINT_array_Write[4]	14		Decimal	DINT
DINT_array_Write[5]	15		Decimal	DINT
DINT_array_Write[6]	16		Decimal	DINT
DINT_array_Write[7]	17		Decimal	DINT
DINT_array_Write[8]	18		Decimal	DINT
DINT_array_Write[9]	19		Decimal	DINT
Distributed_Control_Panel:1:C	{...}	{...}		AB:1734_DI8:C:0
Distributed_Control_Panel:1:I	2#0000_0000		Binary	SINT
Distributed_Control_Panel:2:C	{...}	{...}		AB:1734_DO8:N...
Distributed_Control_Panel:2:I	2#0000_0000		Binary	SINT
Distributed_Control_Panel:2:Q	2#0000_0000		Binary	SINT
Distributed_Control_Panel:3:I	{...}	{...}		AB:1734_8CFGD...
Distributed_Control_Panel:3:Q	{...}	{...}		AB:1734_8CFGD...
Distributed_Control_Panel:I	{...}	{...}		AB:1734_4SLOT:1:0
Distributed_Control_Panel:Q	{...}	{...}		AB:1734_4SLOT:...
Local:1:C	{...}	{...}		AB:1769_IQ16F:C:0

Monitor Tags Edit Tags

The write values sent to the M850 are shown below in debug mode.

(Running) - Connected Components Workbench (Administrator)

File Edit View Build Debug Tools Communications Window Help

HMI Application Language:

Project Organizer

Micro850-VAR

Name: M850_24QBB_192_168_1

Micro850

- Programs
- Global Variables
- DataTypes
- User-Defined Func

Name	Logical Value	Physical Value	Lock	Data Type	Dimension	Alias	Initial Value	Attribute
Test_DINT	...	...		DINT	[0..9]		...	Read/Write
Test_DINT[0]	1	N/A		DINT			1	Read/Write
Test_DINT[1]	2	N/A		DINT			2	Read/Write
Test_DINT[2]	3	N/A		DINT			3	Read/Write
Test_DINT[3]	4	N/A		DINT			4	Read/Write
Test_DINT[4]	5	N/A		DINT			5	Read/Write
Test_DINT[5]	6	N/A		DINT			6	Read/Write
Test_DINT[6]	7	N/A		DINT			7	Read/Write
Test_DINT[7]	8	N/A		DINT			8	Read/Write
Test_DINT[8]	9	N/A		DINT			9	Read/Write
Test_DINT[9]	10	N/A		DINT			10	Read/Write
Test_Writes	...	...		DINT	[0..9]		...	Read/Write
Test_Writes[0]	10	N/A		DINT			00	Read/Write
Test_Writes[1]	11	N/A		DINT			0	Read/Write
Test_Writes[2]	12	N/A		DINT			0	Read/Write
Test_Writes[3]	13	N/A		DINT			0	Read/Write
Test_Writes[4]	14	N/A		DINT			0	Read/Write
Test_Writes[5]	15	N/A		DINT			0	Read/Write
Test_Writes[6]	16	N/A		DINT			0	Read/Write
Test_Writes[7]	17	N/A		DINT			0	Read/Write
Test_Writes[8]	18	N/A		DINT			0	Read/Write
Test_Writes[9]	19	N/A		DINT			0	Read/Write

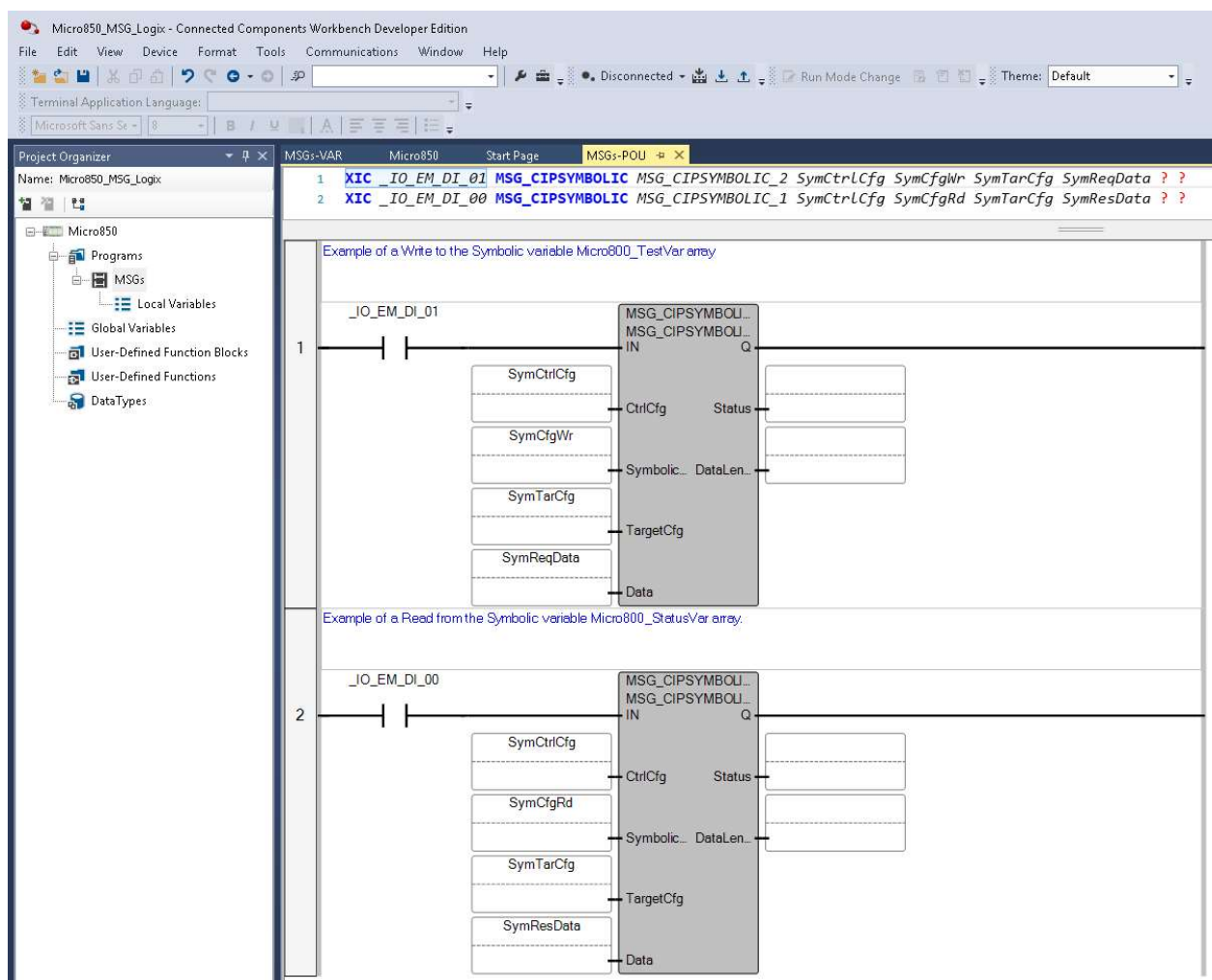
Success: This answer has been added to your Favorites
NOTE: It is not possible to write to the Micro800 Symbolic Variable (SVAR) if the same stands for User Defined Data Types created on Micro800 unit.

Micro800 to Logix Messaging (available from CCW4).

Using the same setup on our Micro850 as before with IP address set to 192.168.1.3 with subnet 255.255.255.0. The CompactLogix used in this update, is a 1769-L18ERM at IP address 192.168.1.10 with subnet 255.255.255.0.

Note that CCW 11.0 is being used in the following pictures, so the layout will look different.

The program shows 2 off Symbolic MSG instructions, the first for writing values and the second for reading values.



The variables for the Write instruction are shown below:

- **SymCfgWr**
 - **.Server** set to 1 for **Write**.
 - **.Symbol** set to the exact tag name you will be writing to. In this case **Micro800_TestVar[0]**.

- **.Count** set to the length or number of elements to be sent. In this case **8**.
- **.DataType** set to the corresponding value from the helpfiles referenced below. In this case **194**.
- **.Offset** set to the starting offset (in bytes) of the read variable. In this case **0**.

• SymTarCfg

- **.Path** set to the path from the Micro800 to the controller. In our example to the 1769-L36ERM our path is **4,192.168.1.10**. For older CompactLogix controllers (for example 1769-L35E), path after IP address should point to backplane and slot 0 like **4,192.168.1.10,1,0**
- **.CipConnMode** set to **0** for unconnected or **1** for class 3 connected messaging. In this case **1**.
- **.UcmmTimeout** set to the timeout value for an unconnected message (**.CipConnMode 0**). In this case, since we are using connected message, **0**.
- **.ConnMsgTimeout** set to the timeout value for a connected message (**.CipConnMode 1**). In this case **4000** (4 seconds).
- **.ConnClose** set to whether the connection should be closed after completion. In this case **FALSE** as we will not need the connection closed.

The screenshot shows the Micro800 MSG_Logic - Connected Components Workbench Developer Edition interface. The main window displays a table of variables and their properties. The table has columns for Name, Alias, Data Type, Dimension, Project Value, Initial Value, Comment, and String Size.

Name	Alias	Data Type	Dimension	Project Value	Initial Value	Comment	String Size
SymCtrlCfg		CIPCONTROLCFG					
SymCtrlCfg.Cancel		BOOL				Abort the execution of message	
SymCtrlCfg.TriggerType		UDINT			0	0 - Trigger once, n - Cyclic trigger	
SymCtrlCfg.StrMode		USINT				reserved parameter	
SymCfgWr		CIPSYMBOLICCFG					
SymCfgWr.Service		USINT			1	0 - Read, 1 - Write	
SymCfgWr.Symbol		STRING			'Micro800_TestVar[0]'	Symbol name to read / write	80
SymCfgWr.Count		UINT			8	Num of variables to read/write. 1 -	
SymCfgWr.DataType		USINT			194	Symbol data type	
SymCfgWr.Offset		USINT			0	Byte offset of variable to read / writ	
SymCfgRd		CIPSYMBOLICCFG					
SymTarCfg		CIPTARGETCFG					
SymTarCfg.Path		STRING			'4,192.168.1.10'	CIP destination path	80
SymTarCfg.CipConnMode		USINT			1	0 - Unconnected, 1 - Class3 connec	
SymTarCfg.UcmmTimeout		UDINT			0	Unconnected message time out.	
SymTarCfg.ConnMsgTimeout		UDINT			4000	Connected message time out.	
SymTarCfg.ConnClose		BOOL			FALSE	TRUE: Close CIP connection upon r	
MSG_CIPSYMBOLIC_1		MSG_CIPSYMBOLIC					
MSG_CIPSYMBOLIC_2		MSG_CIPSYMBOLIC					
SymReqData		USINT	[1..8]				
SymReqData[1]		USINT			1		
SymReqData[2]		USINT			2		
SymReqData[3]		USINT			3		
SymReqData[4]		USINT			4		
SymReqData[5]		USINT			5		
SymReqData[6]		USINT			6		
SymReqData[7]		USINT			7		
SymReqData[8]		USINT			8		
SymResData		USINT	[1..8]				

Note: the SymReqData array initial values to be sent to the Logix controller.

The DataType comes from the CCW Help files:

Success: This answer has been added to your Favorites

Microsoft Help Viewer 2.1 - Connected Components Workbench 11.0 Help

Index

- MSG_CIPSYMBOLIC function block
- MSG_CIPGENERIC function block
- MSG_CIPSYMBOLIC function block
- MSG_MODBUS function block
- MSG_MODBUS2 function block
- multiple sessions
- multiplication operator
- multistate indicator
- multistate push button
- MUX4B function
- MUX8B function
- naming conventions
 - constants
 - function blocks
 - programs
 - variables
- navigation tabs
- Neg operator
- not equal operator
- NOT operator
- NOT_MASK function
- numeric display
- numeric entry object
- numeric increment decrement object
- numeric keypads
- numeric variables
- object activation
- object palette
- objects
 - controllers
 - copy
 - create
 - delete
 - drawing objects
 - format
 - interactive objects
 - move
 - object description
 - rename
 - resize
 - select
- operators
- 1Gain

Data types

Offset	USINT	Reserved for future use.
		A byte offset of Read/Write variable used to Read/Write a large size variable that cannot be processed in one message. 0 – 0xFF
Reserved for future use.		

Symbolic data type support

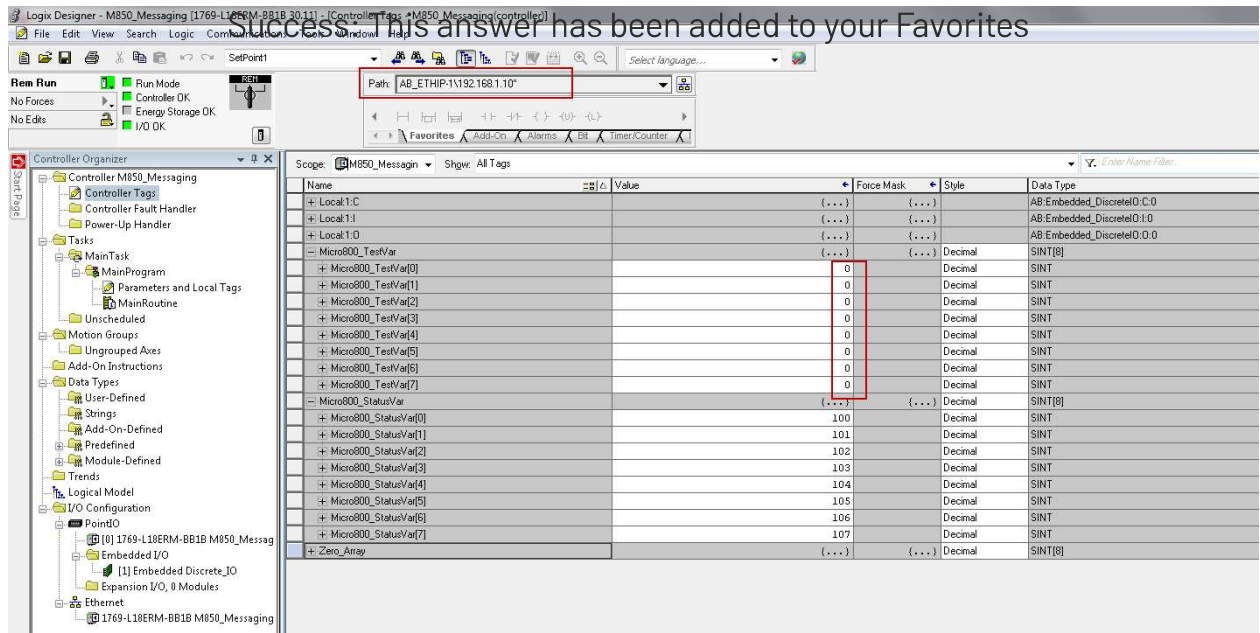
Use this table to help determine the data types MSG_CIPSYMBOLIC supports.

Data type	Data type value (hexadecimal)	Description
BOOL	193 (0xC1)	Logical Boolean with values TRUE (1) and FALSE (0)
SINT	194 (0xC2)	Signed 8-bit integer value
INT	195 (0xC3)	Signed 16-bit integer value
DINT	196 (0xC4)	Signed 32-bit integer value
LINT	197 (0xC5)	Signed 64-bit integer value
USINT	198 (0xC6)	Unsigned 8-bit integer value
UINT	199 (0xC7)	Unsigned 16-bit integer value
UDINT	200 (0xC8)	Unsigned 32-bit integer value
ULINT	201 (0xC9)	Unsigned 64-bit integer value
REAL	202 (0xCA)	32-bit floating point value
LREAL	203 (0xCB)	64-bit floating point value
STRING	218 (0xDA)	Character string

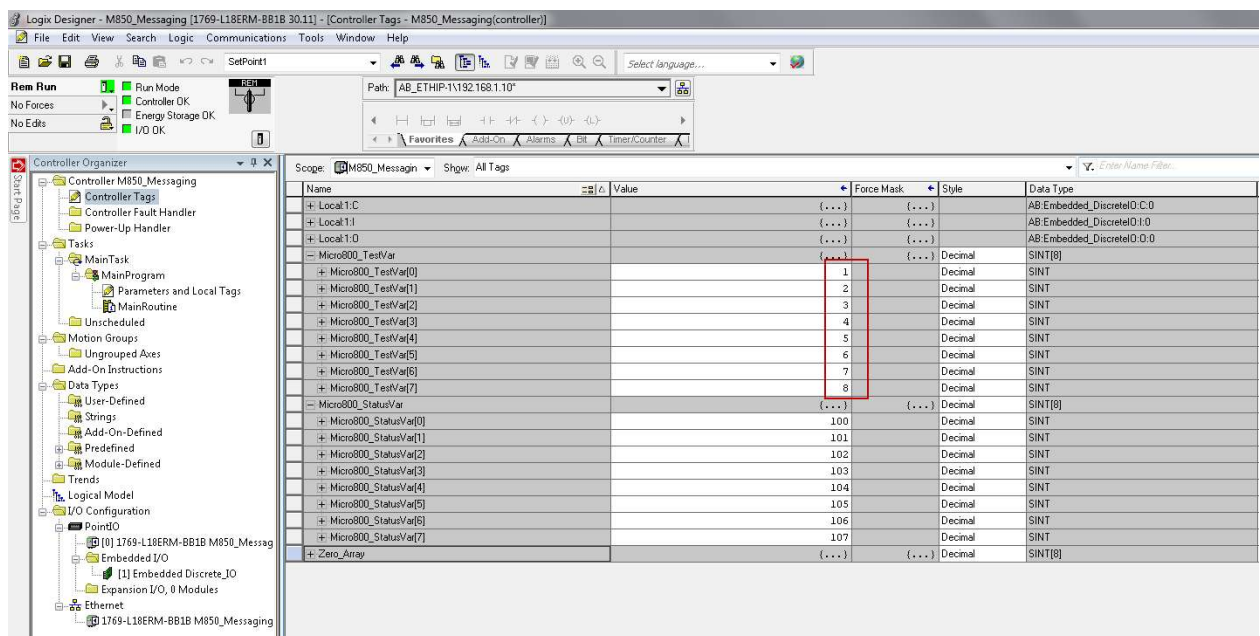
See also

- [MSG_CIPSYMBOLIC](#)
- [Symbolic Read/Write syntax](#)

Download the program, and connect via Ethernet to the Logix controller: The Micro850_TestVar array tags show zeros.



Toggling the input **_IO_EM_DI_01**, enables the Write MSG instruction, and provides the following change to the array values:



The variables for the Read instruction are shown below:

- **SymCfgRd**
 - **.Server** set to 0 for **Read**.
 - **.Symbol** set to the exact tag name you will be writing to. In this case **Micro800_StatusVar[0]**.
 - **.Count** set to the length or number of elements to be sent. In this case **8**.

- **.DataType** set to the corresponding value from the helpfiles referenced above. In this case **Success**: This answer has been added to your Favorites
- **.Offset** set to the starting offset (in bytes) of the read variable. In this case **0**.

The **SymTarCfg** can be the same as for the write message.

Note: the **SymResData** array tags are all at zeros for the start.

The screenshot shows the Micro850_MSG_Logix - Connected Components Workbench Developer Edition. The main window displays a table of variables for the Micro850 device. The table has columns for Name, Alias, Logical Value, Physical Value, Initial Value, Lock, Data Type, Dimension, Comment, and String Size. The SymResData array is highlighted in blue, and its values are all 0.

Name	Alias	Logical Value	Physical Value	Initial Value	Lock	Data Type	Dimension	Comment	String Size
SymCtrlCfg						CIPCONTROL			
SymCtrlCfg.Cancel			N/A			BOOL		Abort the execution of message	
SymCtrlCfg.TriggerType		0	N/A	0		USINT		0 - Trigger once, n - Cyclic trigger	
SymCtrlCfg.StrMode		0	N/A			USINT		reserved parameter	
SymCfgWr						CIPSYMBOLIC			
SymCfgRd						CIPSYMBOLIC			
SymCfgRd.Service		0	N/A	0		USINT		0 - Read, 1 - Write	
SymCfgRd.Symbol		Micro800_StatusVar(0)	N/A	"Micro800_StatusVar(0)"		STRING		Symbol name to read / write	80
SymCfgRd.Count		8	N/A	8		USINT		Num of variables to read/ write. 1 -	
SymCfgRd.DataType		194	N/A	194		USINT		Symbol data type	
SymCfgRd.Offset		0	N/A	0		USINT		Byte offset of variable to read / write	
SymTarCfg						CIPTARGETCF			
SymTarCfg.Path		4,192.168.1.10	N/A	"4,192.168.1.10"		STRING		CIP destination path	80
SymTarCfg.CipConnMode		1	N/A	1		USINT		0 - Unconnected, 1 - Class3 connection	
SymTarCfg.UcnmTimeout		0	N/A	0		USINT		Unconnected message time out.	
SymTarCfg.ConnMsgTimeout		4000	N/A	4000		USINT		Connected message time out.	
SymTarCfg.ConnClose			N/A	FALSE		BOOL		TRUE: Close CIP connection upon	
MSG_CIPSYMBOLIC_1						MSG_CIPSYME			
MSG_CIPSYMBOLIC_2						MSG_CIPSYME			
SymReqData						USINT	[1..8]		
SymResData						USINT	[1..8]		
SymResData[1]		0	N/A	0		USINT			
SymResData[2]		0	N/A	0		USINT			
SymResData[3]		0	N/A	0		USINT			
SymResData[4]		0	N/A	0		USINT			
SymResData[5]		0	N/A	0		USINT			
SymResData[6]		0	N/A	0		USINT			
SymResData[7]		0	N/A	0		USINT			
SymResData[8]		0	N/A	0		USINT			

toggling input **_IO_EM_DI_00**, enables the Read MSG instruction, and provides the following change to the array values:

Success: This answer has been added to your Favorites

Name	Alias	Logical Value	Physical Value	Initial Value	Lock	Data Type	Dimension	Comment	String Size
SymCtrlCfg						CIPCONTROL			
SymCtrlCfg.Cancel			N/A			BOOL		Abort the execution of message	
SymCtrlCfg.TriggerType		0	N/A	0		USINT		0 - Trigger once, n - Cyclic trigger	
SymCtrlCfg.StbMode		0	N/A			USINT		reserved parameter	
SymCfgWr						CPSYMBOLIC			
SymCfgRd						CPSYMBOLIC			
SymCfgRd.Service		0	N/A	0		USINT		0 - Read, 1 - Write	
SymCfgRd.Symbol		Micro800_StatusVar[0]		Micro800_StatusVar[0]		STRING		Symbol name to read / write	80
SymCfgRd.Count		0	N/A	0		USINT		Num of variables to read/ write, 1 -	
SymCfgRd.DataType		194	N/A	194		USINT		Symbol data type	
SymCfgRd.Offset		0	N/A	0		USINT		Byte offset of variable to read / writ	
SymTarCfg						CIPTARGETCH			
SymTarCfg.Path		4.192.168.1.10	N/A	4.192.168.1.10		STRING		CIP destination path	80
SymTarCfg.CipConnMode		1	N/A	1		USINT		0 - Unconnected, 1 - Class3 connect	
SymTarCfg.UcmmTimeout		0	N/A	0		USINT		Unconnected message time out.	
SymTarCfg.ConnMsgTimeout		4000	N/A	4000		USINT		Connected message time out.	
SymTarCfg.ConnClose			N/A	FALSE		BOOL		TRUE: Close CIP connection upon r	
MSG_CPSYMBOLIC_1						MSG_CPSYME			
MSG_CPSYMBOLIC_2						MSG_CPSYME			
SymReqData						USINT	[1..8]		
SymReqData[1]		100	N/A	0		USINT	[1..8]		
SymReqData[2]		101	N/A	0		USINT			
SymReqData[3]		102	N/A	0		USINT			
SymReqData[4]		103	N/A	0		USINT			
SymReqData[5]		104	N/A	0		USINT			
SymReqData[6]		105	N/A	0		USINT			
SymReqData[7]		106	N/A	0		USINT			
SymReqData[8]		107	N/A	0		USINT			

Attached are similar examples of messages between a Logix controller and a Micro850.

Attachments

File

[Messaging_to_Micro850.ACD](#)

DISCLAIMER

This knowledge base web site is intended to provide general technical information on a particular subject or subjects and is not an exhaustive treatment of such subjects. Accordingly, the information in this web site is not intended to constitute application, design, software or other professional engineering advice or services. Before making any decision or taking any action, which might affect your equipment, you should consult a qualified professional advisor.

ROCKWELL AUTOMATION DOES NOT WARRANT THE COMPLETENESS, TIMELINESS OR ACCURACY OF ANY OF THE DATA CONTAINED IN THIS WEB SITE AND MAY MAKE CHANGES THERETO AT ANY TIME IN ITS SOLE DISCRETION WITHOUT NOTICE. FURTHER, ALL INFORMATION CONVEYED HEREBY IS PROVIDED TO USERS "AS IS." IN NO EVENT SHALL ROCKWELL BE LIABLE FOR ANY DAMAGES OF ANY KIND INCLUDING DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, LOSS PROFIT OR DAMAGE, EVEN IF ROCKWELL AUTOMATION HAVE BEEN ADVISED ON THE POSSIBILITY OF SUCH DAMAGES.

ROCKWELL AUTOMATION DISCLAIMS ALL WARRANTIES WHETHER EXPRESSED OR IMPLIED IN RESPECT OF THE INFORMATION (INCLUDING SOFTWARE) PROVIDED HEREBY, INCLUDING THE IMPLIED WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, AND NON-INFRINGEMENT. Note that certain jurisdictions do not countenance the exclusion of implied warranties; thus, this disclaimer may not apply to you.

Success: This answer has been added to your Favorites