Quantum

140 NOC 77100

EtherNet/IP Communication Module

User Manual

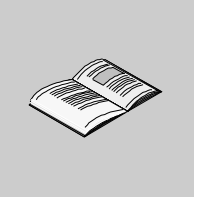
6/2008

31008209.00

www.schneider-electric.com

Schneider Electric

Table of Contents



Safety Information 7

About the Book 9

Chapter 1 Installation 11

Hardware Installation. 12
Module Specifications 15

Chapter 2 Configuring the 140 NOC 771 00 17

2.1 Creating a Project in Unity Pro 18
Creating a Project in Unity Pro 19
Configuring the 140 NOC 771 00 Ethernet/IP Communication Module 21
2.2 Using the Unity Pro EtherNet/IP Configuration Tool 29
EtherNet/IP Configuration Tool User Interface 30
Devices Window 33
Configuring Properties in the Devices Window 34
2.3 Configuring Network Channel Properties 36
Configuring Channel Properties: The General page 37
Configuring Channel Properties: The Ethernet page 39
Configuring Channel Properties: The EtherNet/IP page 40
Configuring Channel Properties: The Module Information page 42
2.4 Configuring the TCP/IP Address Settings 49
TCP/IP Properties: The General Page 50
TCP/IP Properties: Configuring the SNMP Agent 52
TCP/IP Properties: Configuring the DHCP Server 54
2.5 Configuring the EtherNet/IP Communication Module as an I/O Adapter 57
Identifying the Local Slave 58
Local Slave Inputs and Outputs 59
Configuring Local Slave Properties: The General page 61

Chapter 3 Adding Devices to an EtherNet/IP Network 65
3.1 Adding Devices to an EtherNet/IP Network 66
Effect of Device Position on Input and Output %MW Memory Addresses. 66
3.2 Adding and Configuring Remote Devices 71
Device Library 72
Add an EDS File to the Device Library 74
Adding A Remote Device. 76
Configuring Remote Device Properties 78
Managing Project Files 82
3.3 Configuring the STB NIC 2212 84
Setting Up Your Network 85
Automatically Detect and Add the STB NIC 2212 87
Configuring STB NIC 2212 Properties. 88
Connecting to the Advantys STB Island 92
Configuring I/O Items. 97
3.4 Connecting to Third Party Devices 112
Adding a Third Party Device to the Sample Network 113
Add an EDS File 114
Automatically Detect and Add the 1734-AENT PointIO Adapter 116
Configuring 1734-AENT PointIO Adapter Properties. 117
Viewing 1734-AENT PointIO Adapter I/O Addresses 121

Chapter 4 Optimizing Performance 125
4.1 Selecting a Switch 126
Role of a Switch in an EtherNet/IP Network 127
Full Duplex. 127
IGMP Snooping 128
Port Mirroring. 128
Virtual Local Area Network (VLAN) 129
Simple Network Management Protocol (SNMP) Agent 129
4.2 Understanding EtherNet/IP Connections. 130
EtherNet/IP Messaging Overview 131
TCP Connections. 133
CIP Connections and Messages 134
Messaging Performance 134
4.3 Estimating EtherNet/IP System Performance 135
Determining EtherNet/IP Network Performance 136
Network Load Calculation Example. 138

Chapter 5 Explicit Messaging In Unity Pro 141
Explicit Messaging Services 142
Configuring Explicit Messaging Using MBP_MSTR 144
MBP_MSTR Example - Get_Attributes_Single 148
MBP_MSTR Example - Reset. 154
Explicit Messaging Error Codes 160
Explicit Messaging - Online Action: Get_Attributes_Single 162
Explicit Messaging - Online Action: Reset. 164

Chapter 6 CIP Objects 167
Adapter Diagnostic Object. 168
Assembly Object 172
Connection Manager Object 173
Ethernet Link Object 175
Identity Object 178
Module Diagnostic Object 180
Scanner Diagnostic Object 182
TCP/IP Interface Object 186

Chapter 7 Diagnostics 189
LED Indicators for the 140 NOC 771 00 EtherNet/IP Communication Module. 190
Diagnostic Testing Using the Unity Pro EtherNet/IP Software 192
Ping a Network Device 194
Viewing Output Messages in the Unity Pro EtherNet/IP Configuration Tool. 195

Chapter 8 Replacing the EtherNet/IP Communication Module 197
Replacing the EtherNet/IP Communication Module 197

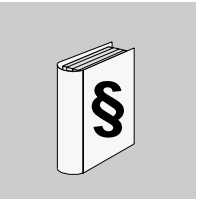
Appendices 199

Appendix A Error Codes 201
TCP/IP Ethernet Error Codes 201

Glossary 205

Index 213

Safety Information



NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger or Warning safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER

DANGER indicates an imminently hazardous situation, which, if not avoided, **will result** in death or serious injury.

⚠ WARNING

WARNING indicates a potentially hazardous situation, which, if not avoided, **can result** in death, serious injury, or equipment damage.

⚠ CAUTION

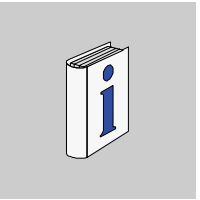
CAUTION indicates a potentially hazardous situation, which, if not avoided, **can result** in injury or equipment damage.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

© 2008 Schneider Electric. All Rights Reserved.

About the Book



At a Glance

Document Scope

This manual describes the use of the Quantum 140 NOC 771 00 EtherNet/IP communication module. This manual presents a continuing sample configuration. The features of the module are described as they are encountered in the course of this continuing sample configuration.

The specific configuration settings contained in this manual are intended to be used for instructional purposes only. The settings required for your specific EtherNet/IP configuration may—and probably will—differ from the examples presented in this manual.

Validity Note

The data and illustrations found in this book are not binding. We reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be construed as a commitment by Schneider Electric.

Related Documents

Title of Documentation	Reference Number
Advantys STB EtherNet/IP Network Interface Applications Guide	31008204

For additional information, you can also refer to the online help files for both the:

- Unity Pro software
- Unity Pro EtherNet/IP Configuration Tool software

Product Related Warnings

Schneider Electric assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When controllers are used for applications with technical safety requirements, please follow the relevant instructions.

Failure to use Schneider Electric software or approved software with our hardware products may result in improper operating results.

Failure to observe this product related warning can result in injury or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techpub@schneider-electric.com

Installation

1

At a Glance

Overview

The 140 NOC 771 00 communication module serves as the interface between a Quantum PLC (CPU) and an EtherNet/IP network. This chapter shows you how to install the module by:

- inserting it into a Quantum backplane, and
- connecting it to an EtherNet/IP network

What's in this Chapter?

This chapter contains the following topics:

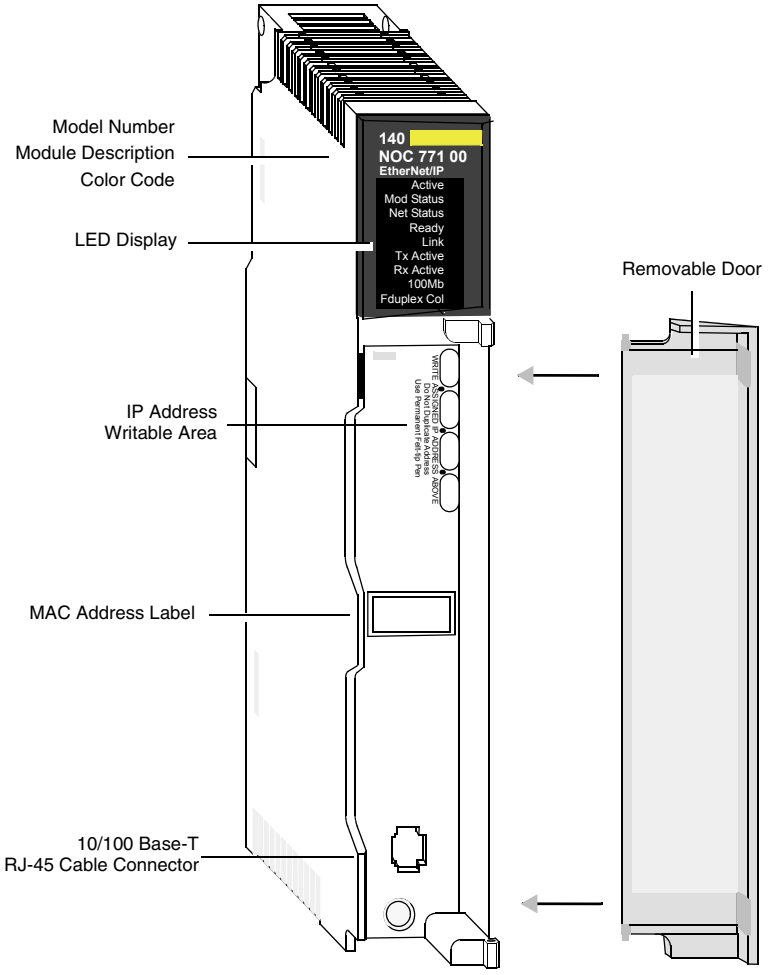
Topic	Page
Hardware Installation	12
Module Specifications	15

Hardware Installation

Overview

The following information describes how to install the 140 NOC 771 00 EtherNet/IP communication module.

External Features



LEDs

The 140 NOC 771 00 EtherNet/IP module presents the following LED indicators:

- Active
- Module Status
- Network Status
- Ready
- Link
- Transmission Activity
- Reception Activity
- 100 Mb link
- Full Duplex/Collision

For a description of these LEDs, and how to use them to diagnose the module, refer to the topic LED Indicators for the 140 NOC 771 00 Communication Module (see, *p. 190*).

Locating a Backplane Slot

The 140 NOC 771 00 EtherNet/IP module is mounted in a Quantum PLC station. It can be installed in any available position in the Quantum backplane.

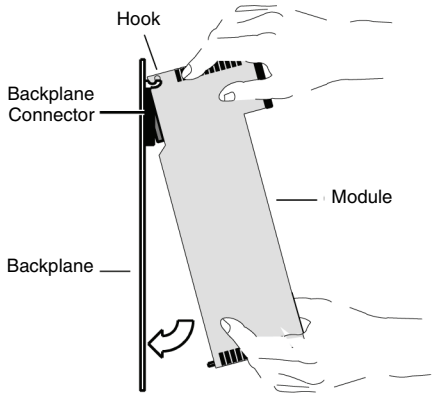
Selecting a Power Supply

When configuring the Quantum PLC station, be sure to add a power supply module that is capable of supplying power to all modules on the rack.

Tools Required

One medium-sized (size 2) Phillips-head screw driver.

Mounting the Module in the Backplane

Step	Action
1	Holding the module at an angle, mount it on the two hooks located near the top of the backplane. The following figure shows the correct way to hold the module. 
2	Swing the module down so the connector engages the backplane connector.
3	Use a Phillips-head screw driver to tighten the safety screw at the bottom of the module from 2 through 4 in-lbs or from .22 through .45 Newton meters of torque.

Wiring the Ethernet Connector

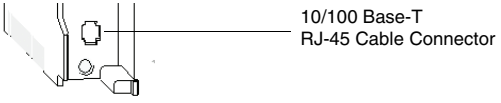
⚠ WARNING

HAZARD OF ELECTRICAL SHOCK OR BURN

Connect the ground wire to the protective earth (PE) terminal before you establish any further connections. When you remove connections, disconnect the ground wire last. For noise immunity, EMC compliance, and safety, the Ethernet cable shield must be connected to PE ground at the Ethernet switch.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The 140 NOC 771 00 module communicates over an EtherNet/IP network through a single RJ45 connector located at the bottom of the module.



Module Specifications

Related Documentation

Refer to the Hardware Installation (see, *p. 12*) section of this chapter and the *Quantum Ethernet Modules* section in the *Quantum with Unity Pro Hardware Reference Guide*, for more detailed information on the installation.

Specifications

Communication Ports	One auto-sensing 10/100Base-T shielded twisted pair (RJ-45 connector) port.
Bus Current Required	500 mA
Power Dissipation	2.5 W
Fuse	None
Operating Conditions	
Temperature	0...+60° C
Humidity	0...95% Rh non-condensing @ 60°C
Altitude	2000 m (6561.68 ft)
Vibration	10...57 Hz @ 0.0075 mm d.a
	57...150 Hz @ 1 g
Storage Conditions	
Temperature	-40...+85°C
Humidity	0...95% Rh non-condensing @ 60°C
Free Fall	1 m unpackaged
Shock	+/– 15 g, 11 ms, half sine wave

Software Compatibility

The 140 NOC 771 00 is compatible with Unity Pro XL programming software version 4.0 and higher.

Standards

The 140 NOC 771 00 module complies with the following standards:
- UL 508
- CSA 22.2-142
- CE
- C-TICK
- ODVA

Communication Modules per Rack

The maximum number of communication modules—including but not limited to 140 NOC 771 00 EtherNet/IP communication modules—you can install in a single rack is determined by the CPU serving that rack:

CPU	Maximum Number of Communication Modules per Rack
140 CPU 311 10	2
140 CPU 434 12A	6
140 CPU 534 14A	6
140 CPU 651 50	6
140 CPU 651 60	6
140 CPU 652 60	6
140 CPU 671 60	6

Configuring the 140 NOC 771 00

2

At a Glance

Overview

This chapter shows you how to use Unity Pro programming software and the Unity Pro EtherNet/IP configuration tool to select and configure the 140 NOC 771 00 EtherNet/IP communication module.

Note: The instructions presented in this chapter include specific choices made for a sample project. Your Unity Pro project may include different choices that are appropriate for your specific configuration.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Creating a Project in Unity Pro	18
2.2	Using the Unity Pro EtherNet/IP Configuration Tool	29
2.3	Configuring Network Channel Properties	36
2.4	Configuring the TCP/IP Address Settings	49
2.5	Configuring the EtherNet/IP Communication Module as an I/O Adapter	57

2.1 Creating a Project in Unity Pro

At a Glance

Overview

This section provides information about:

- selecting Quantum modules in Unity Pro
- launching the Unity Pro EtherNet/IP configuration tool

Note: For detailed information about how to use Unity Pro, refer to the online help and documentation DVD that come with the Unity Pro XL programming software.

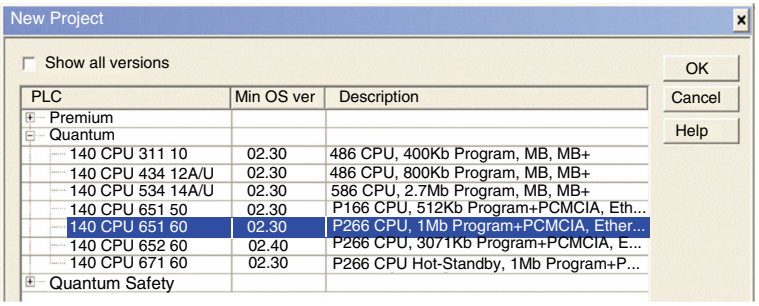
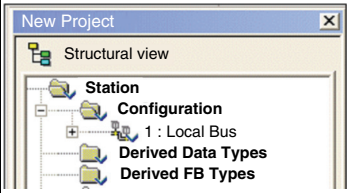
What's in this Section?

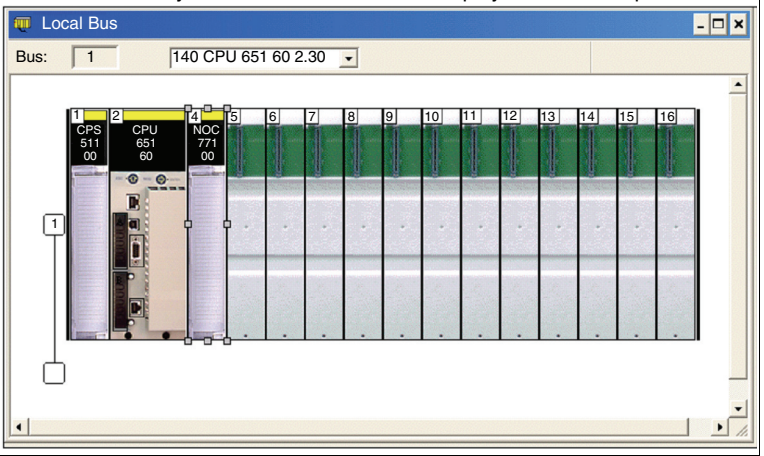
This section contains the following topics:

Topic	Page
Creating a Project in Unity Pro	19
Configuring the 140 NOC 771 00 Ethernet/IP Communication Module	21

Creating a Project in Unity Pro

Use Unity Pro to create a new project. The following steps describe a sample project created in Unity Pro:

Step	Action
1	Open Unity Pro.
2	In the Unity Pro main menu, select **File** → **New...**. The **New Project** window opens displaying a list of Schneider-Electric controller types.
3	In the **New Project** window, open the Quantum sub-list and select a controller. In this example, the 140 CPU 651 60 controller is selected: 
4	Click **OK**. The **Project Browser** opens: 
5	In the **Project Browser**, double click **Local Bus**. Unity Pro displays: ● the **Hardware catalog** and ● a **Local Bus** window with the selected CPU in the second position
6	In the **Hardware catalog**, do the following: ● In the **Supply** section, use your mouse to select then drag a 140 CPS 114 10 115/230 VAC power supply to a position in the backplane—in this example, slot 1. ● In the **Communication** section, drag a 140 NOC 771 00 EtherNet/IP communication module to a position in the backplane—in this example, slot 4.

Step	Action
7	The modules that you have selected are now displayed in the backplane. 
8	To open the configuration window for the 140 NOC 771 00, do one of the following: ● double click the left mouse button on the 140 NOC 771 00 module in the **Local Bus** window above, or ● click the right mouse button on the module, then select **Open Module...** in the popup menu The module configuration window opens, where you can configure its properties.

Configuring the 140 NOC 771 00 Ethernet/IP Communication Module

Overview

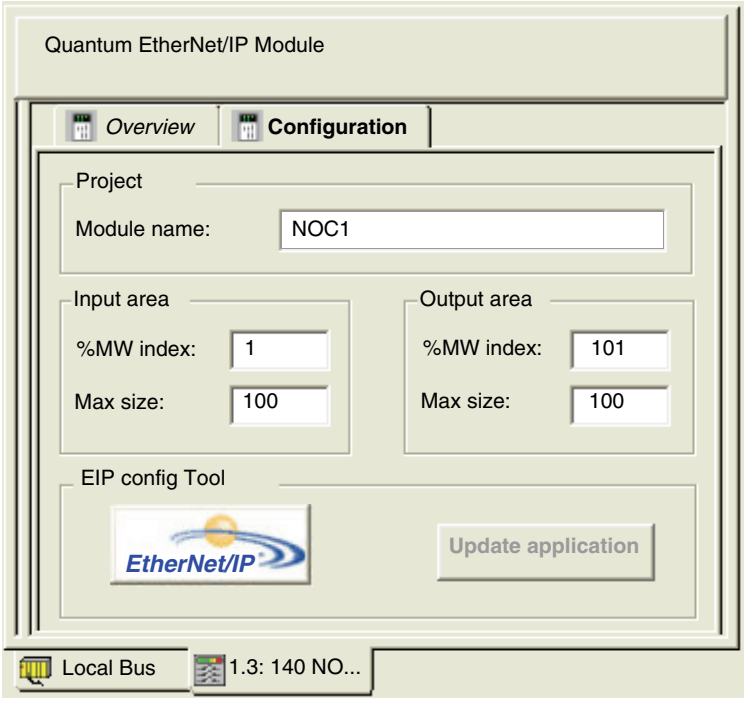
To configure properties for the 140 NOC 771 00, you need to:

- complete the Configuration page of the module properties window
- launch the Unity Pro EtherNet/IP configuration tool, where you can complete the process of editing properties for EtherNet/IP modules and devices
- add the completed EtherNet/IP module and device edits to the Unity Pro project configuration in the form of derived data types

The following steps present one example of how to configure the communication module. Your own configurations may differ.

Setting Input and Output Memory Addresses and Naming the Module

The Configuration page looks like this:



In the Configuration screen, perform the following steps to name the module, and to set addresses and sizes for both inputs and outputs:

Step	Action
1	In the **Project** section, enter a name for your module in the **Module name** field. In this example, the name **NOC1** is entered. **Note:** After the module name is entered and the EtherNet/IP configuration is validated (after clicking the Validate button), the module name cannot be edited.
2	In the **Input area and Output area** type in the size and starting position of both the inputs and outputs. These values can later be edited. For this example, the following values are entered: In the **Input area**: ● In the **%MW index** field, type in a starting address for inputs—in this example: **1**. ● In the **Max size** field, type in the maximum number of 16-bit words dedicated to inputs—in this example: **100**. In the **Output area**: ● In the **%MW index** field, type in a starting address for outputs—in this example: **101**. ● In the **Max size** field, type in the maximum number of 16-bit words dedicated to outputs—in this example: **100**. **Notes:** ● The inputs and outputs can be located at any available address and do not need to be located in adjacent areas. It is important only that the space allocated to inputs and outputs do not overlap. ● The specified %MW range for both inputs and outputs must be available in the CPU. For more information, refer to the Unity Pro help file topic Processor Configuration Screen.
3	In Unity Pro select **Edit** → **Validate** (or click the **Validate** button) to: ● save the EtherNet/IP module name—which becomes a non-editable, read-only value ● save the address and size settings for inputs and outputs, and ● start up the EtherNet/IP configuration tool

Launching the EtherNet/IP Configuration Tool

After you have saved both the EtherNet/IP module name and the input and output settings, launch the EtherNet/IP configuration tool by clicking on the EtherNet/IP button:

The EtherNet/IP configuration tool opens for editing. If EtherNet/IP device configurations have previously been edited and saved, they will be displayed.

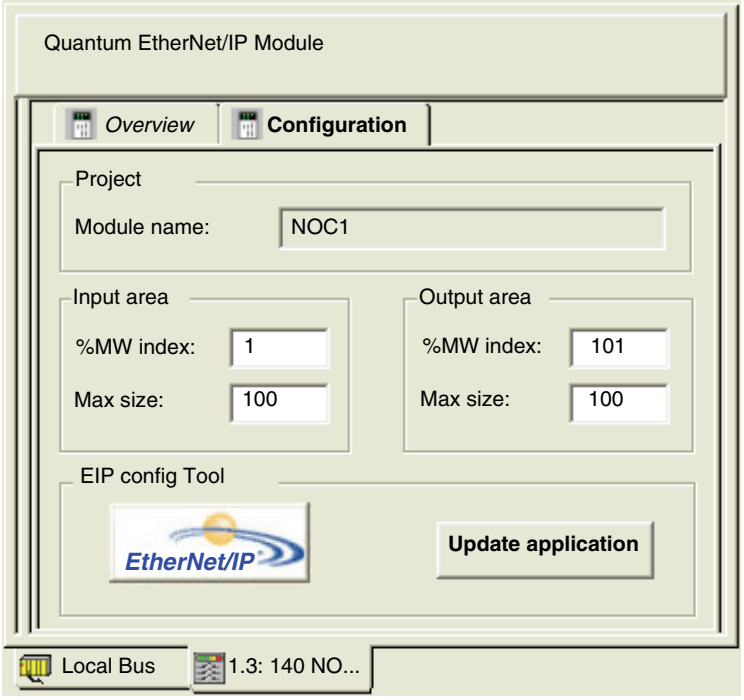
Use the EtherNet/IP configuration tool to configure:

- the EtherNet/IP module's:
  - channel properties (see *p. 36*)
  - TCP/IP settings (see *p. 49*)
  - local slave function (see *p. 57*)
- remote EtherNet/IP devices, including:
  - the 140 NIC 2212 network interface module (see *p. 84*)
  - third-party remote devices (see *p. 112*)

Note: Only a single instance of the configuration tool can be open at any time.

Creating or Updating Derived Data Types

After all EtherNet/IP module edits have been saved in the EtherNet/IP configuration tool, add these edits to the Unity Pro project, as follows:

Step	Action
1	Return to the main screen in Unity Pro and select the **Configuration** page of the EtherNet/IP configurable server module, below. Note that the **Update application** button is now enabled. 
2	Click the **Update application** button. **Note:** Every time you use the EtherNet/IP configuration tool to make edits, be sure to return to this screen and click the **Update application** button to save your edits to the Unity Pro project.

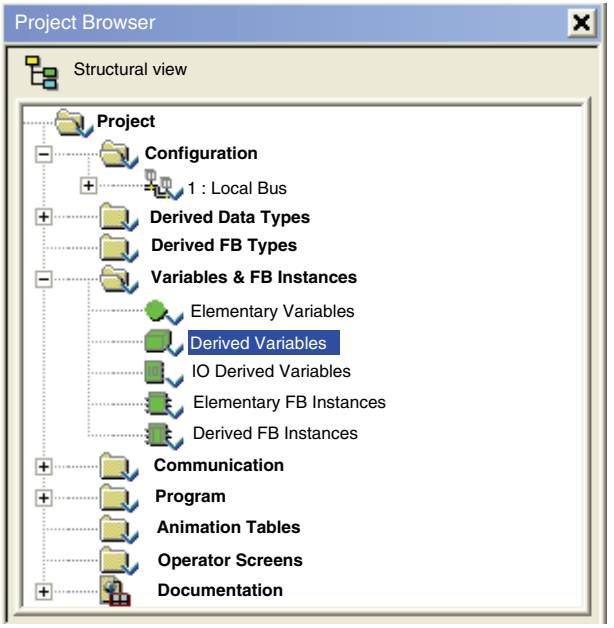
Step	Action
3	The Unity Pro software converts the EtherNet/IP configuration data to variables and derived data types, then prepares to import them into the Unity Pro project. If any conflict exists between a newly created item and an existing one, Unity Pro displays those conflicts in the **Import Trouble Report**, below: Import Trouble Report Type / Name / New Name / Keep / Replace / Rename Duplicate DTT / ST_NOC1_IN / ST_NOC1_IN_0 / / / X Duplicate DTT / ST_NOC1_IN_D... / ST_NOC1_IN_D... / / / X Duplicate DTT / ST_NOC1_IN_D... / ST_NOC1_IN_D... / / / X Duplicate DTT / ST_NOC1_OUT / ST_NOC1_OUT_0 / / / X Duplicate DTT / ST_NOC1_OUT... / ST_NOC1_OUT... / / / X Duplicate DTT / ST_NOC1_OUT... / ST_NOC1_OUT... / / / X The variable exist... / NOC1_IN / NOC1_IN_0 / / / X The variable exist... / NOC1_OUT / NOC1_OUT_0 / / / X OK / Cancel / Keep All / Replace All In this example, the Unity Pro project configuration already includes the listed variables and derived data types.
4	If the **Import Trouble Report** opens, indicate how you want to treat each item. You can specify the following selections for each item, or for all items: ● **Keep:** Keeps the component of the current project. ● **Replace:** Replaces the project component with the one from the import. ● **Rename:** Renames the imported component, allowing you to keep both components.
5	After you have determined how to treat each imported item, click **OK**.
6	After you click **OK**, the Project Browser displays the new or edited derived data types, below: Project Browser Structural view **Project** **Configuration** 1 : Local Bus **Derived Data Types** ST_NOC1_IN ST_NOC1_IN_DEVICE_A ST_NOC1_IN_DEVICE_B ST_NOC1_OUT ST_NOC1_OUT_DEVICE_A ST_NOC1_OUT_DEVICE_B

Viewing Derived Data Type Variables

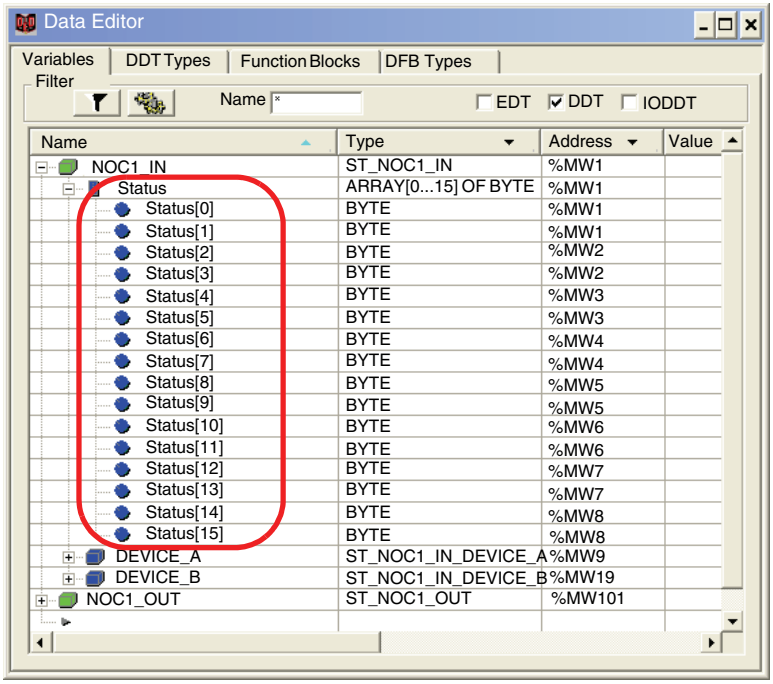
When you clicked on the **Update application** button, Unity Pro created a collection of derived data type variables. Use these variables to view the:

- status of all connections from the communication module to each remote device, where:
  - the status of all connections is displayed in an array of 16 bytes
  - each connection is represented by a single bit
  - a bit value of 1 indicates the connection is healthy
  - a bit value of 0 indicates the connection is lost, or the communication module can no longer communicate with the remote device
- value of input and output items you created using the Unity Pro EtherNet/IP configuration tool
- value of attributes defined by the EDS file of a remote device
- amount of padding, representing the reserved input or output memory space for a remote device

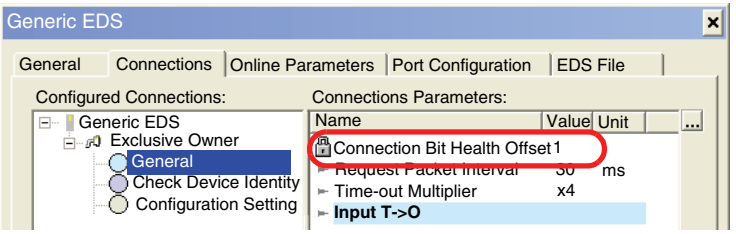
To view these derived data type variables:

Step	Description
1	Return to the main screen in Unity Pro.
2	In the **Project Browser** open the branch **Variables & FB Instances** and double-click on the **Derived Variables** sub-branch.  The **Data Editor** window opens, displaying the **Variables** tab. A check mark appears in the **DDT** checkbox. (If not, select the **DDT** checkbox.)

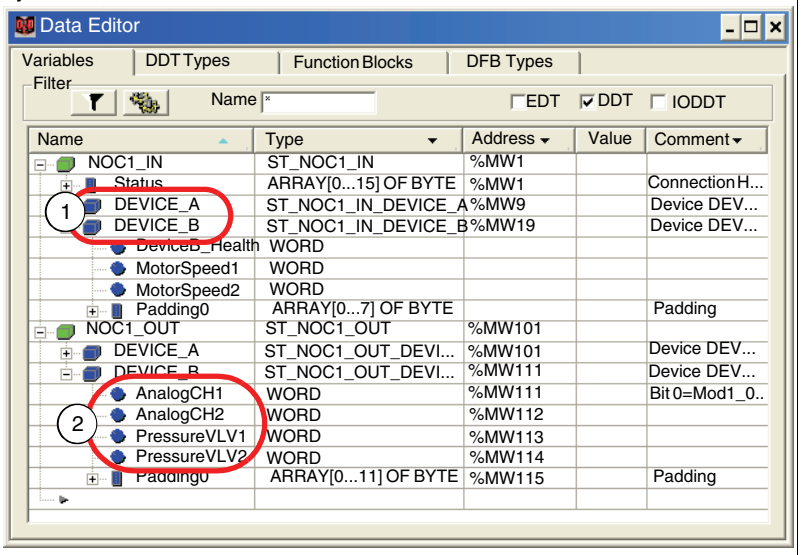
Step	Description
3	The **Status** variable—describing the status of all connections—is found beneath the first device in the first position. To display the Status variable's 16-byte array, expand the first device as depicted below



4	To determine which **Status** bit is mapped to a specific remote device:	
	a	In the Unity Pro EtherNet/IP configuration tool, open the **Properties** window for a remote device.
	b	Open the **Connections** page, and click on the General node, below:



	c	In the above example, the **Connection Bit Health Offset** value of 1 maps to the first bit in the first byte of the **Status** variable, which can be represented as `Status[0].1`.

Step	Description
5	You can also use the **Data Editor** to display DDT variables. DDT variables are either: ● input and output items you created using the Unity Pro EtherNet/IP configuration tool, or ● attributes defined by the remote device's EDS file, or ● padding, representing the reserved but unused input or output memory space for a remote device The **Data Editor** presents DDT variables in separate input and output groups, sorted by device, as shown below:  **1** device names: user-created in the Unity Pro EtherNet/IP configuration tool **2** variable names: user-created as I/O items in the Unity Pro EtherNet/IP configuration tool, or defined as a property by the EDS file of the remote device

2.2 Using the Unity Pro EtherNet/IP Configuration Tool

At a Glance

Overview

This section describes the Unity Pro EtherNet/IP configuration tool user interface. Use the configuration tool to enter settings for the EtherNet/IP communication module and for other devices connected to your EtherNet/IP network.

What's in this Section?

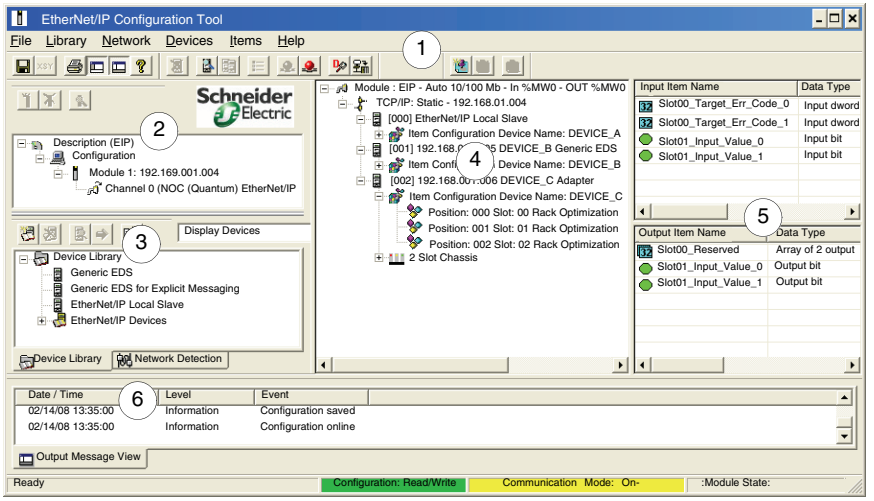
This section contains the following topics:

Topic	Page
EtherNet/IP Configuration Tool User Interface	30
Devices Window	33
Configuring Properties in the Devices Window	34

EtherNet/IP Configuration Tool User Interface

Overview

The Unity Pro EtherNet/IP configuration tool user interface presents the following parts:



1 Main menu
2 Description area
3 Workspace area
4 Devices window
5 I/O area
6 Output Message window

The parts of the EtherNet/IP user interface are briefly described below.

Main Menu

The Main menu area consists of:

- A main menubar with the following menu items and commands:

Menu item	Contains commands for...
File	• file management and printing • GUI display selections • online / offline operations
Library	managing EDS files in the **Device Library**

Network	• automatic detection of EtherNet/IP network devices
	• online actions, including:
	  • explicit messaging
	  • pinging network devices
	• commissioning devices via port configuration settings
	• working with automatically detected devices in the **Network Detection** area

Devices	working with devices in the **Devices** window, including: • displaying devices in the Devices window tree control • commissioning devices via port configuration settings • creating and configuring CIP connections for devices • diagnosing devices
Items	adding, deleting, and renaming inputs and outputs in the I/O area
Help	• displaying versioning information of the Unity Pro EtherNet/IP configuration tool • online help

- 3 toolbars:

Toolbar	Contains commands that relate to...
Main toolbar	• file management and printing • GUI display selections
Devices toolbar	working with devices in the Devices window, including: • displaying devices in the Devices window tree control • commissioning devices • creating and configuring CIP connections for devices • diagnosing devices • online / offline operations
Items toolbar	adding, deleting, and renaming inputs and outputs in the I/O area

Description Area

The **Description** area describes the EtherNet/IP communication module and its IP address.

Workspace Area

The **Workspace** area consists of two tabs, containing the:

- Device Library, where you can:
  - view properties and EDS files for all available EtherNet/IP devices
  - add a new device and its EDS file to the Device Library
  - delete a device from the **Device Library**
  - manage the display of devices in the **Device Library** list
  - insert a selected device into the configuration in the **Devices** window
- Network Detection area, where you can:
  - automatically detect EtherNet/IP devices on the network
  - take online actions, including sending explicit messages and pinging network devices
  - view properties and EDS files for all available EtherNet/IP devices
  - insert a single selected device into the configuration in the **Devices** window
  - insert all detected device into the configuration in the **Devices** window, replacing all devices in the configuration

You can show or hide the workspace area using the **File** → **Preferences** → **Workspace** command.

Devices Window

The Devices window contains a tree control, containing all devices that have been added to your EtherNet/IP network configuration. In the **Devices** window, you can:

- display and edit the properties of selected EtherNet/IP devices, including:
  - EtherNet/IP communication modules
  - local slaves
  - remote devices
  - I/O modules
- commission devices
- create and configure CIP connections for devices
- open the I/O area and display individual inputs and outputs
- diagnose device connections

I/O Area

The I/O Area displays the configuration data for each input and output, including the:

- name
- data type
- offset within the device
- offset within the connection
- address where the I/O data is sent to, or sent from

The I/O area is displayed only when a device I/O connection is selected in the Devices window configuration.

Output Message Window

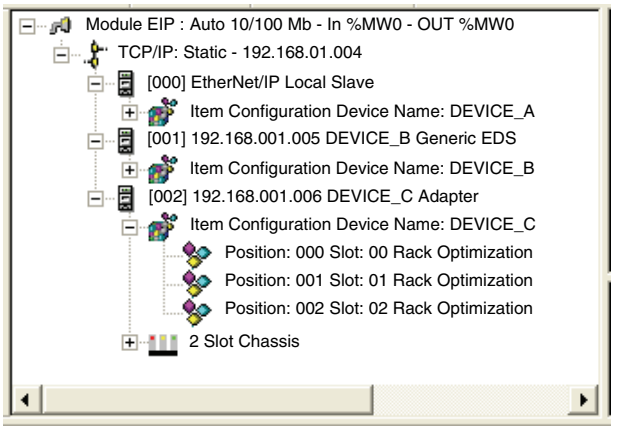
The **Output Message** window displays a sortable list of events. The Output Message window can be configured to show or hide each item's:

- date and time
- event level:
  - informational
  - warning
  - error

Devices Window

Overview

The **Devices** window is located in the center of the EtherNet/IP configuration tool's user interface and displays a node for each device in your network configuration. A example of the **Devices** window appears below:



Configurable Properties

The **Devices** window displays a node for each device—and for each device's configurable property group— in your network configuration. Each node is identified by an icon, as follows:

Node	Icon	This node is used to configure...
Channel		The properties of the EtherNet/IP module's communication channel.
TCP/IP		The EtherNet/IP communication module's IP addressing, SNMP and DHCP server settings.
Local slave		Properties related to the module's role as an I/O adapter to a remote device acting in the role of I/O scanner.
Device		The properties of any EtherNet/IP network device with an IP address, including both modular and non-modular devices.
Items collection		The name assigned to a group of I/O items.
Item		The properties of a CIP connection between the EtherNet/IP communication module and individual I/O items. If the type of connection is: ● **rack optimized**: click on the connection in the first position to display all rack optimized I/O items. ● **direct**: click on a connection for any position to display the I/O items for that connection.
Chassis		The properties of a chassis that is part of a modular device.
Module		The parameters of an I/O module that is part of a modular device.

Configuring Properties in the Devices Window

Overview

Use the **Devices** window, in the Unity Pro EtherNet/IP configuration tool, to display and configure properties for the EtherNet/IP communication module and other devices on your EtherNet/IP network.

To configure properties, double-click on the Devices window node associated with the properties (see *p. 33*) you want to configure.

For example, to configure the EtherNet/IP communication module network channel properties, double-click on the channel icon to display the **Channel Properties** window. When the window first opens, it displays 2 tabbed pages:

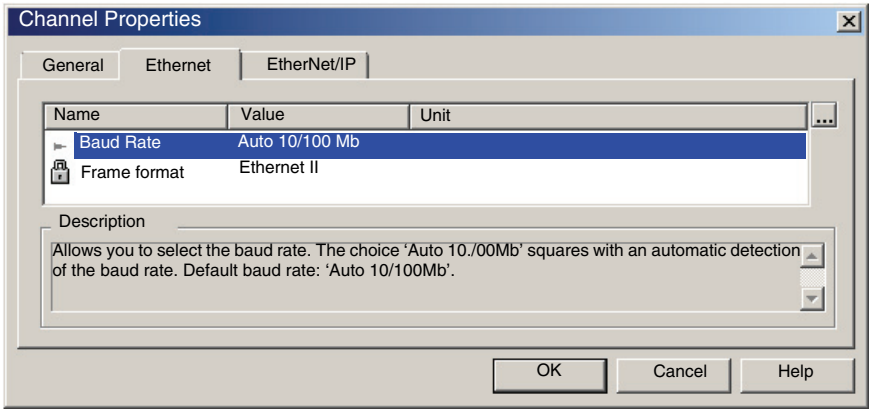
- General (the default tab)
- Ethernet

Note that the Channel Properties window can also display 2 additional pages:

- Ethernet/IP, by operating in Advanced mode (**File** → **Preferences** → **Advanced**)
- Module Information, by operating online (**File** → **Go Online**)

Displaying Property Values

Most property windows let you display a description of a selected property. Select a property in the **Name** column to display a brief description of the selected property in the **Description** area at the bottom of the window:



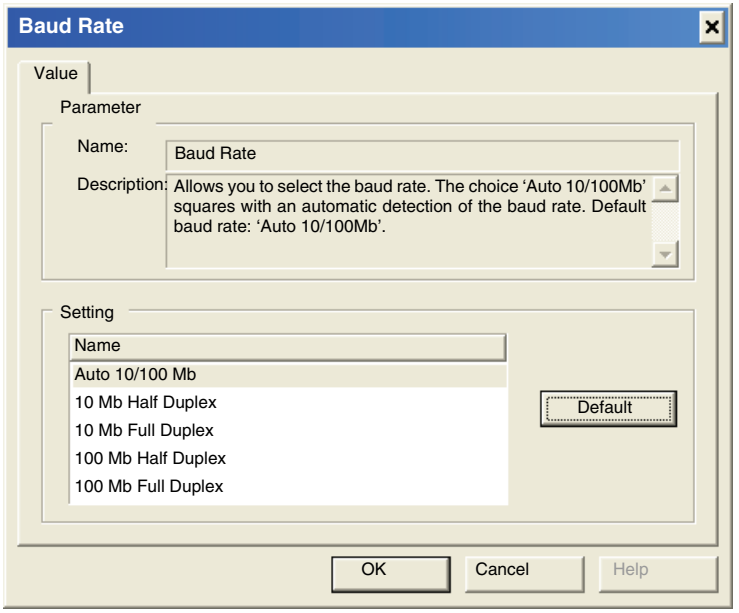
Property Types

Properties can be either read/write or read-only, as indicated by its icon:

This icon...	Indicates the property is...
	Read-only. This property value is locked and cannot be edited.
	Read-write. This property value can be edited.

Editing Property Values

To edit a read-write property value, follow these steps:

Step	Action
1	Select a read-write property.
2	There are two ways to perform an edit: ● Double-click the left mouse button on the property name. ● Click the ellipsis (...) button located at the right of the list header bar, then select **Properties** in the popup menu. A window opens—in this case for the **Baud Rate** property—where you can edit the parameter value:  **Note:** Some other properties are editable by typing in a value within a stated range.
3	After completing your edits click **OK** to close the **Properties** window and save your edits.

2.3 Configuring Network Channel Properties

At a Glance

Overview

This section describes how to configure network channel properties with the EtherNet/IP configuration tool.

What's in this Section?

This section contains the following topics:

Topic	Page
Configuring Channel Properties: The General page	37
Configuring Channel Properties: The Ethernet page	39
Configuring Channel Properties: The EtherNet/IP page	40
Configuring Channel Properties: The Module Information page	42

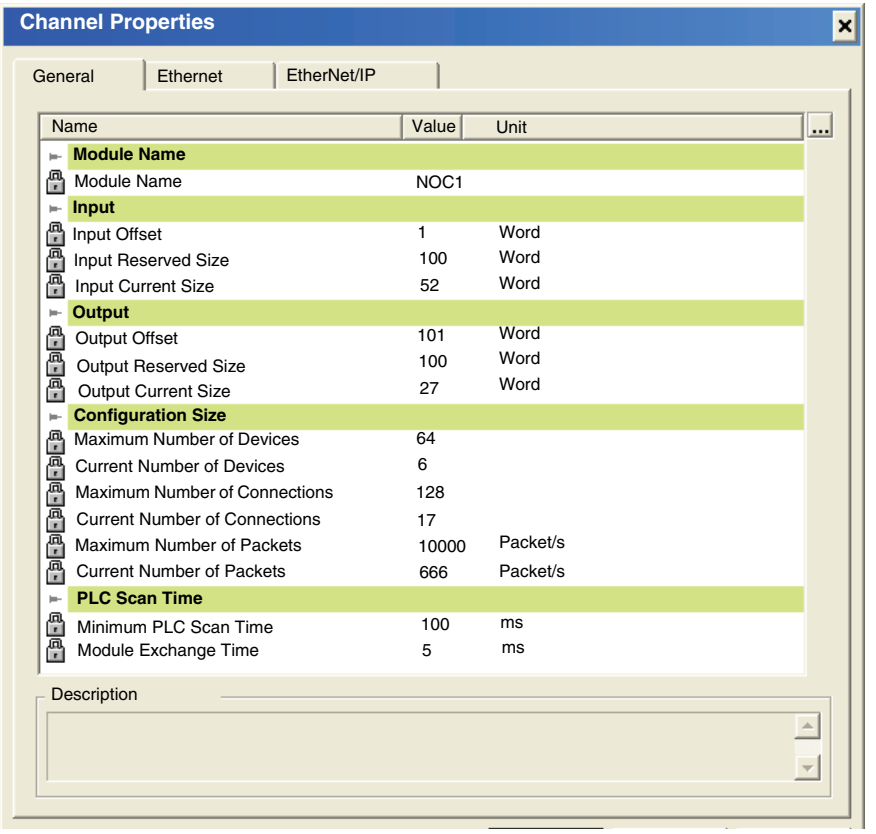
Configuring Channel Properties: The General page

The General Page

The **General** page of the **Channel Properties** window displays read-only properties that describe the:

- EtherNet/IP communication module name
- size and location of inputs and outputs
- size of the EtherNet/IP configuration

The module's property values are set by the communication module's EDS file, the configuration design, and settings entered in the **Configuration** page of Unity Pro for the EtherNet/IP communications module.



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Properties

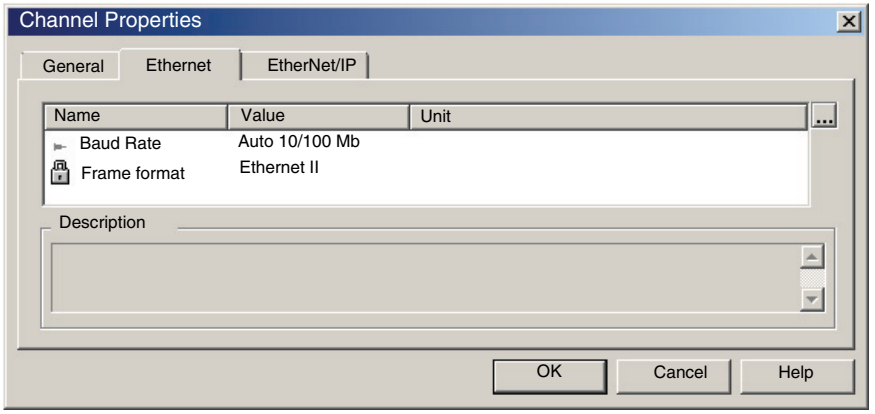
Name	Description	Value set by...
Module Name		
Module Name	The name of the EtherNet/IP module	**Configuration** page in Unity Pro
Input		
Input Offset	The starting address for inputs (%MW index)	**Configuration** page in Unity Pro
Input Reserved Size	The total number of words configured for inputs (Max size)	**Configuration** page in Unity Pro
Input Current Size	The actual number of inputs used in the application	network design in the configuration tool's **Devices window**
Output		
Output Offset	The starting address for outputs (%MW index)	**Configuration** page in Unity Pro
Output Reserved Size	The total number of words configured for outputs (Max size)	**Configuration** page in Unity Pro
Output Current Size	The actual number of outputs used in the application	network design in the configuration tool's **Devices window**
Note: When configuring an offset and a reserved size for both inputs and outputs, be sure that inputs and outputs do not overlap.		
Configuration Size		
Maximum Number of Devices	The maximum number of devices that can be added to the configuration.	predefined
Current Number of Devices	The number of devices currently in the configuration.	network design in the configuration tool's **Devices window**
Maximum Number of Connections	The maximum number of connections that can be managed by the module.	predefined
Current Number of Connections	The number of connections in the configuration.	network design in the configuration tool's **Devices window**
Maximum Number of Packets	The maximum number of packets per second the module is able to manage.	predefined
Current Number of Packets	The number of packet/s that will be generated by the current configuration.	network design in the configuration tool's **Devices window**
PLC Scan Time		
Minimum PLC Scan Time	The estimated cycle time to process inputs and outputs, equal to the sum of estimates for communication over both the backplane and the network.	predefined
Module Exchange Time	The estimated additional time contributed by the EtherNet/IP module to perform the I/O management. This value is included in the "minimum PLC scan time" value.	predefined

Configuring Channel Properties: The Ethernet page

The Ethernet Page

Use the **Ethernet** page of the **Channel Properties** window to:

- view and edit the **Baud Rate**
- view the **Frame format**



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Properties

Name	Description	Type
Baud Rate	The transmission speed and duplex mode for the configuration. To change these settings, double-click on the field name and select one of the following: ● Auto 10/100 Mb (the default) ● 10 Mb Half duplex ● 10 Mb Full duplex ● 100 Mb Half duplex ● 100 Mb Full duplex **Note:** The default setting—**Auto 10/100 Mb**—is recommended. It causes the connected devices to perform auto-negotiation and thereby determine the fastest common transmission rate and duplex mode.	Read-Write
Frame Format	Ethernet II is the only frame format available for this module.	Read-Only

Configuring Channel Properties: The EtherNet/IP page

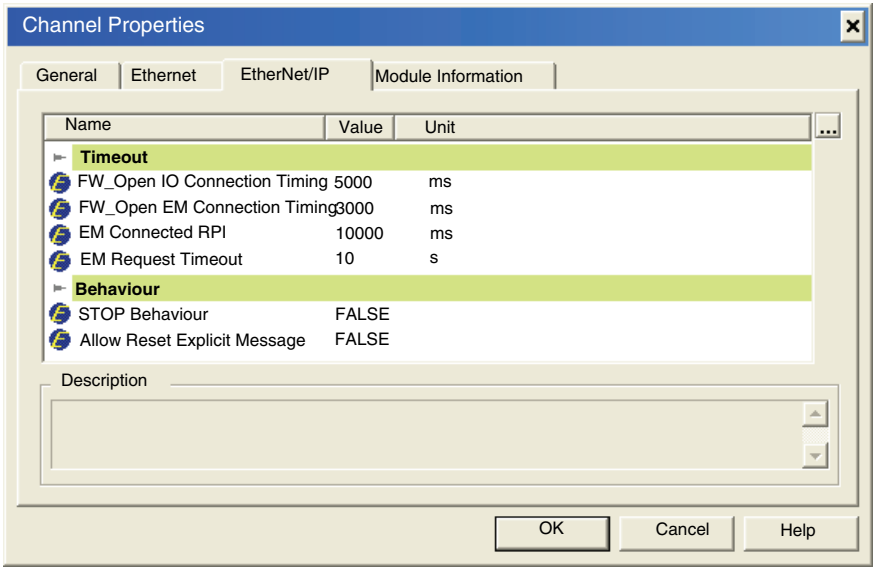
The EtherNet/IP Page

Use the **EtherNet/IP** page of the **Channel Properties** window to configure:

- properties that determine how the EtherNet/IP communication module, in its role as an I/O scanner, opens both implicit and explicit connections
- the frequency for transmitting produced data over implicit connections
- the timeout period for explicit connections
- the behavior of the module, in its role as an I/O scanner, when:
  - the application is stopped, or
  - the EtherNet/IP module receives a reset service request

Note: This page is displayed only when you are using Advanced Mode. Advanced mode properties are indicated by the icon.

To turn on Advanced Mode, select: **File** → **Preferences** → **Advanced**



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Configuring EtherNet/IP Properties

Note: Only an experienced developer of EtherNet/IP networks should edit any of the following read-write properties.

Name	Description
Timeout	
FW_Open IO Connection Timing	The amount of time the EtherNet/IP module waits for the Forward_Open IO messaging transaction to open an implicit messaging connection. Default = 5000 ms
FW_Open EM Connection Timing	The amount of time the EtherNet/IP module waits for the Forward_Open IO messaging transaction to open an explicit messaging connection. Default = 3000 ms
EM Connected RPI	The value used to set the T->O (target to originator) and O->T (originator to target) requested packet interval (RPI) for all explicit message connections. This value is used to calculate the lifetime of a connection. Default = 10000 ms.
EM Request Timeout	The amount of time the EtherNet/IP module waits between a request and reply of an explicit message. Default =10 s.
Output	
STOP Behavior	The state of the EtherNet/IP module when the CPU application goes into a STOP state: ● TRUE indicates that the module enters STOP state (implicit connections are closed). ● FALSE indicates that the module enters IDLE state (implicit connections are not closed). Default = FALSE
Allow Reset Explicit Message	The behavior of the EtherNet/IP module—as I/O scanner—when it receives a reset service request: ● TRUE indicates the module resets itself. ● FALSE indicates the module ignores the reset service request and continues uninterrupted operations. Default = FALSE

Configuring Channel Properties: The Module Information page

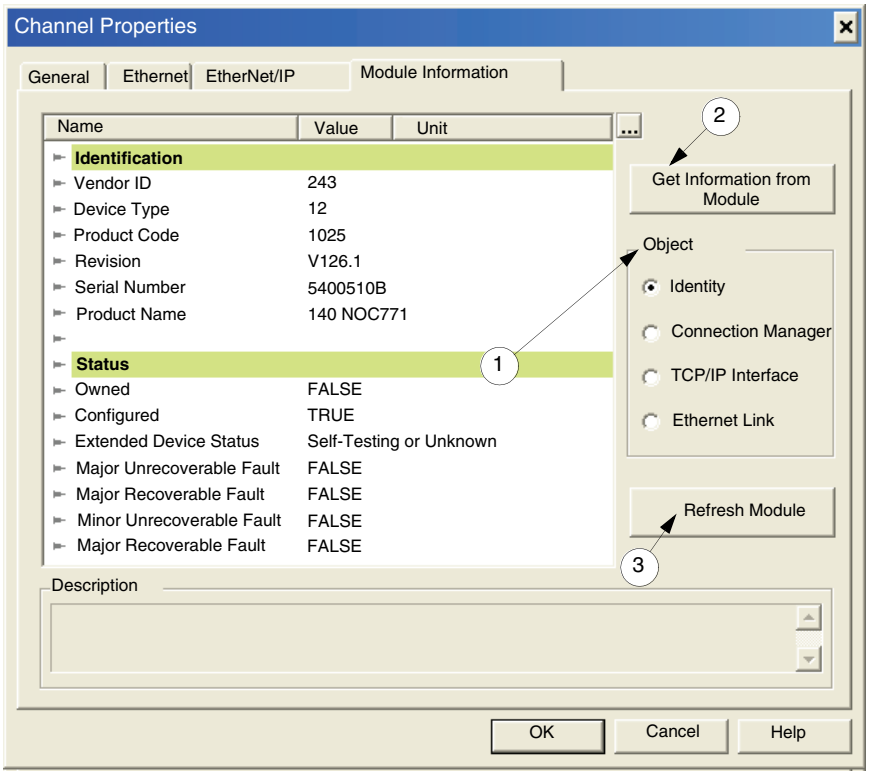
The Module Information Page

Use the **Module Information** page of the **Channel Properties** window to display properties obtained from the EtherNet/IP communication module. In this page you can:

- retrieve data from the module's EDS file
- display retrieved module data by a selected object group, including data relating to the module's:
  - Identity
  - Connection Manager
  - TCP/IP Interface
  - Ethernet Link
- refresh data

Note:
- This page is displayed only when the Unity Pro EtherNet/IP configuration tool is operating online. To operate online, select **File** → **Go Online**.
- All object groups are displayed only when you are operating in Advance mode. To operate in Advance mode, select **File** → **Preferences** → **Advance**.

Displaying module information is a 3-step process, as described below:



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Step 1	Select a property type in the **Object** list: ● identity ● Connection Manager ● TCP/IP Interface ● Ethernet Link
Step 2	Click the **Get Information from Module** button to populate property data.
Step 3	Periodically click the **Refresh Module** button to update property data.

Identity Properties and Status

After selecting **Identity**, the following information is displayed.

Property	Description
Identification	
Vendor ID	243
Device Type	12
Product Code	1025
Revision	The revision number of the device
Serial Number	The serial number of the device.
Product Name	140 NOC77100
Status	
Owned	A TRUE setting indicates that the device (or an object within the device) has an owner. The setting of this bit means that the Predefined Master/Slave Connection Set has been allocated to a master.
Configured	A TRUE setting indicates that the application of the device has been configured to do something different than the out-of-the-box default. This does not include configuration of the communications.
Extended Device Status	The vendor-specific or already defined status.
Major Unrecoverable Fault	A TRUE setting indicates the device detected a problem with itself, which caused the device to go into the Major Unrecoverable Fault state.
Major Recoverable Fault	A TRUE setting indicates the device detected a problem with itself, which caused the device to go into the Major Recoverable Fault state.
Minor Unrecoverable Fault	A TRUE setting indicates the device detected a problem with itself, which is thought to be unrecoverable. The problem does not cause the device to go into one of the faulted states.
Minor Recoverable Fault	A TRUE setting indicates the device detected a problem with itself, which is thought to be recoverable. The problem does not cause the device to go into one of the faulted states.

Connection Manager Properties

After selecting **Connection Manager**, the following information is displayed.

Property	Description
Open Counters	
Open Requests	The number of forward open service requests received.
Format Rejects	The number of forward open service requests which were rejected due to bad format.
Resource Rejects	The number of forward open service requests which were rejected due to lack of resources.
Other Rejects	The number of forward open service requests which were rejected for reasons other than bad format or lack of resources.
Close Counters	
Close Requests	The number of forward close service requests received.
Format Rejects	The number of forward close service requests which were rejected due to bad format.
Other Rejects	The number of forward close service requests which were rejected for reasons other than bad format.
Other Counters	
Connection TimeOuts	The total number of connection timeouts that have occurred in connections controlled by this Connection Manager
Numbers of Connection	The number of connections.

TCP/IP Interface Properties

After selecting **TCP/IP Interface**, the following information is displayed. Not all properties apply to the module.

Property	Description
Status	Indicates the status of the configuration: ● 0 = not configured ● 1 = a valid configuration acquired from BOOTP or nonvolatile storage
Configuration Capability	● BOOTP Client Indicates that the device is capable of acquiring its network configuration via BOOTP. ● Configuration Settable Indicates that the configuration is settable.
Startup Configuration	Determines how the device acquires its initial configuration at startup. **Note:** If the device was previously configured, it uses the previously stored interface configuration values.
IP Address	The device IP address. A 0.0.0.0 address indicates an IP address has not been configured.
Network Mask	The device network mask. A 0.0.0.0 address indicates a network mask address has not been configured.
Gateway Address	The default gateway address. A 0.0.0.0 address indicates a gateway address has not been configured.
Primary Name Server Address	(not applicable)
Secondary Name Server Address	(not applicable)
Domain Name	(not applicable)
Host Name	(not applicable)
Safety Network Number	(not applicable)
TTL Value	The value that the device uses for the IP header's Time-to-Live field when sending packets via IP an multicast.
Multicast Address Allocation Control	This determines how the device shall allocate IP multicast addresses. If set to: **0 -** Multicast addresses are generated using the default allocation algorithm. **1 -** Multicast addresses are allocated according to the values specified in the two following parameters.
Number of IP Multicast Addresses Allocated	The number of IP multicast addresses that are allocated.
Starting Multicast IP Address	The starting multicast address from which allocation begins.

Ethernet Link Properties

After selecting **Ethernet Link**, the following information is displayed.

Property	Description
General	
Interface Speed	The interface speed currently in use. A 0 is shown if the speed has not been determined.
Link Status	Indicates whether or not the Ethernet communication interface is connected to an active network.
Duplex Mode	Indicates that duplex mode currently in use.
Negotiation Status	Indicates the status of link auto-negotiation. If set to: **0 -** Auto-negotiation in progress. **1 -** Auto-negotiation and speed detection has failed. Default values for speed and duplex are being used. **2 -** Auto negotiation has failed but the speed has been detected. Duplex was defaulted. The default value is product-dependent; recommended default is half duplex. **3 -** Successfully negotiated speed and duplex. **4 -** Auto-negotiation was not attempted. Speed and duplex has been forced.
Manual Setting Requires Reset	If set to: **0 -** The interface can activate changes to link parameters (auto-negotiate, duplex mode, interface speed) automatically. **1 -** The device requires a reset service be issued to its Identity Object in order for the changes to take effect.
Local hardware Fault	A local hardware fault.
Physical Address	The MAC layer address.
Input	
Octets	The number of octets received on the interface.
Ucast Packets	The number of subnetwork-unicast packets delivered to a higher-layer protocol.
NUcast Packets	The number of non-unicast packets delivered to a higher-layer protocol.
Discards	The number of inbound packets which were chosen to be discarded even though no errors had been detected to prevent their being deliverable to a higher-layer protocol.
Errors	The number of inbound packets that contained errors preventing them from being deliverable to a higher-layer protocol.
In Unknown Protocols	The number of packets received via the interface which were discarded because of an unknown or unsupported protocol.
Output	
Octets	The number of octets sent on the interface.

Property	Description
Ucast Packets	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address.
NUcast Packets	The total number of packets that higher-level protocols requested be transmitted to a non-unicast address.
Discards	The number of outbound packets which were chosen to be discarded even though no errors had been detected to prevent their being transmitted.
Errors	The number of outbound packets that could not be transmitted because of errors.
Error Counters	
Alignment Errors	The number of frames received on this interface that are not an integral number of octets in length and do not pass the FCS check.
FCS Errors	The number of frames received on this interface that are an integral number of octets in length but do not pass the FCS check.
Single Collisions	The number of successfully-transmitted frames on this interface for which transmission is inhibited by exactly one collision.
Multiple Collisions	The number of successfully-transmitted frames on this interface for which transmission is inhibited by more than one collision.
SQE Test Errors	The number of times a SQE test error message has been generated.
Deferred Transmissions	The number of frames for which the first transmission attempt on this interface has been delayed because the medium is busy.
Late Collisions	The number of times a collision is detected later than 512 bit-times into the transmission of a packet.
Excessive Collisions	The number of frames for which transmission on this interface has failed due to excessive collisions.
MAC Transmit Errors	The number of frames for which transmission on this interface has failed due to an internal MAC sublayer transmit error.
Carrier Sense Errors	The number of times that the carrier sense condition was lost or never asserted when attempting to transmit a frame on this interface.
Frame Too Long	The number of frames received on this interface that exceeded the maximum permitted frame size.
MAC Receive Errors	The number of frames for which reception on the interface has failed due to an internal MAC sublayer receive error.

2.4 Configuring the TCP/IP Address Settings

At a Glance

Overview

This section provides information about how to configure the TCP/IP address settings for the EtherNet/IP communication module.

What's in this Section?

This section contains the following topics:

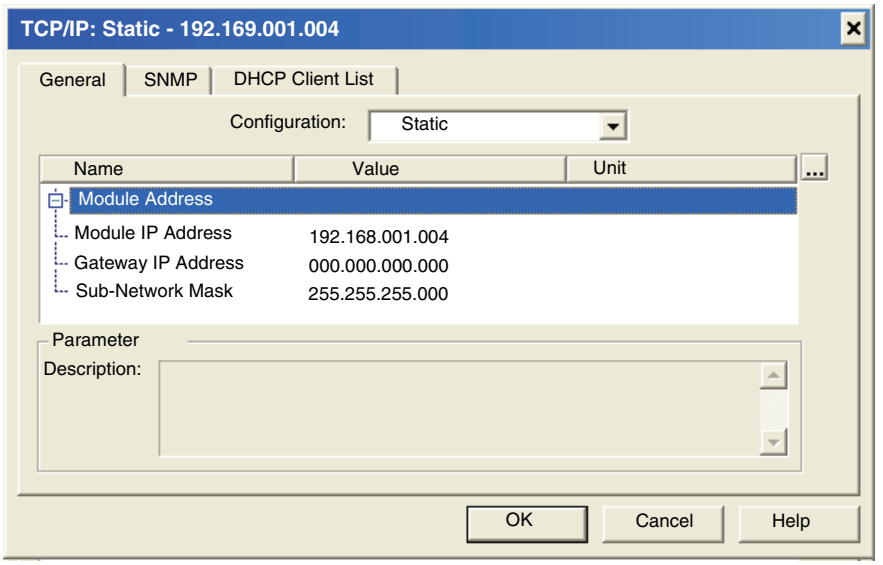
Topic	Page
TCP/IP Properties: The General Page	50
TCP/IP Properties: Configuring the SNMP Agent	52
TCP/IP Properties: Configuring the DHCP Server	54

TCP/IP Properties: The General Page

The General Page

Use the **General** page of the **TCP/IP** properties window to configure the IP address of the EtherNet/IP communication module.

Open the **TCP/IP** properties window by clicking on the TCP/IP icon in the **Devices** window.



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Selecting a Configuration Mode

Use the **Configuration** list to specify a configuration mode. The configuration mode setting determines how the module obtains its IP address at startup. Choices are:

Configuration Mode	Description
Static	The module uses the module IP address, gateway IP address, and sub-network mask configured in this page.
Flash Memory	The module uses the IP address configured via the TCP/IP object and stored flash memory. An IP address configured by this process survives a warm re-start (during which power to the device is continuously maintained), but is lost in the case of a cold re-start (where power to the device is turned off for a time).
BOOTP	The module uses an IP address assigned by a BOOTP server.

Setting the Module Addresses in Static Mode

Three IP address properties need to be configured for the EtherNet/IP communication module in static configuration mode:

Property	Description
Module IP Address	The 32-bit identifier—consisting of both a network address and a host address—assigned to a device connected to a TCP/IP Internet network using the Internet Protocol (IP).
Gateway Address	The address of a device, if any, that serves as a gateway to the EtherNet/IP module.
Sub-Net Mask	The 32-bit value used to hide (or mask) the network portion of the IP address and thereby reveal the host address of a device on a network using the IP protocol.

Default Address Configurations

The module uses a default address configuration when it is not configured or when a duplicate IP address is detected. The default address is based on the MAC address of the module and makes it possible for several Schneider devices to use their default network configuration on the same network.

The module uses the following default address configurations.

- Default IP Address
  This default address starts with 10.10 and uses the last two bytes of the MAC address. As an example, a device with the MAC address of 00:00:54:10:8A:05 has a default IP address of 10.10.138.5 (0x8A=138, 0x05=5).
- Default Subnet Mask
  The default address is 255.0.0.0 (a class A mask).
- Default Gateway Address
  The default gateway address is identical to the default IP address.

Duplicate Address Checking

Before going online, the module sends out at least four ARP (Address Resolution Protocol) messages with a proposed IP address.

- If an answer is returned
  - There is a device already using the IP address.
  - The module will not use the proposed IP address and uses the default IP address.
- If an answer is not returned
  - The module uses the IP address (along with the associated network parameters.)

TCP/IP Properties: Configuring the SNMP Agent

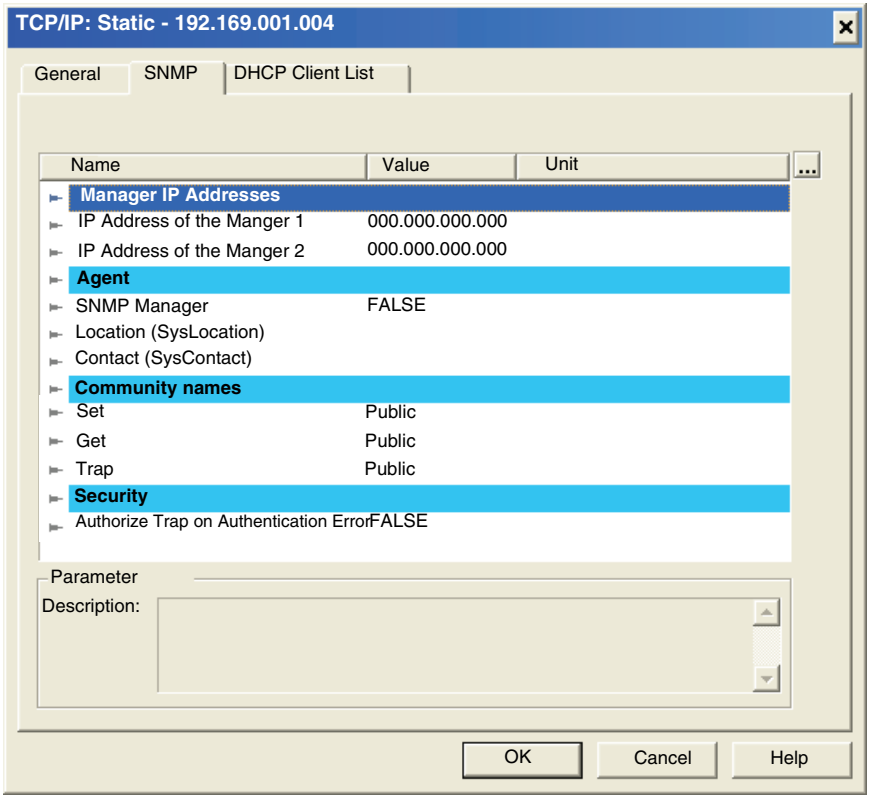
The SNMP page

Use the SNMP page of the TCP/IP properties window to configure the SNMP agent (see *p. 129*) in the EtherNet/IP communication module. An SNMP agent is a software component that reports management data about the module to another device acting as an SNMP manager.

The SNMP agent can connect to and communicate with up to 2 SNMP managers as part of an SNMP service. The SNMP service includes:

- authentication checking, by the EtherNet/IP communication module, of any SNMP manager that sends SNMP requests
- management of event, or trap, reporting by the module

Click on the SNMP tab to access the SNMP agent window:



Note: Refer to the topic Configuring Properties in the Devices Window (see *p. 34*) for information on how to display property descriptions and edit property values.

Viewing and Configuring SNMP Properties

The following properties can be viewed and edited in the SNMP page:

Property	Description
Manager IP Addresses:	
IP Address of the Manager 1	The IP address of the first SNMP manager to which the EtherNet/IP module's SNMP agent sends notices of traps.
IP Address of the Manager 2	The IP address of the second SNMP manager to which the module's SNMP agent sends notices of traps.
Agent:	
SNMP Manager	Select either: • **TRUE**: the Location and Contact information is provided by a network management tool • **FALSE**: Location and Contact settings are made in this window
Location	The device location (32 characters maximum)
Contact	Information describing the person to contact for device maintenance (32 characters maximum)
Community Names:	
Get	Password required by a MIB-II SNMP agent authorizing read commands from an SNMP manager. Default = **Public**.
Set	Password required by a MIB-II SNMP agent authorized write commands from an SNMP manager. Default = **Public**
Trap	Password a MIB-II SNMP manager requires from an SNMP agent causing the SNMP manager to accept trap notices from the SNMP agent. Default = **Public**
Security:	
Authorize Trap on Authentication Error	Causes the SNMP agent to send a trap notice to the SNMP manager if an unauthorized manager sends a Get or Set command to the agent. Default = **FALSE**.

TCP/IP Properties: Configuring the DHCP Server

The DHCP Client List Page

The EtherNet/IP communication module can be configured to perform the function of DHCP server. Connected network devices can subscribe to this DHCP service and obtain their IP parameters from the module.

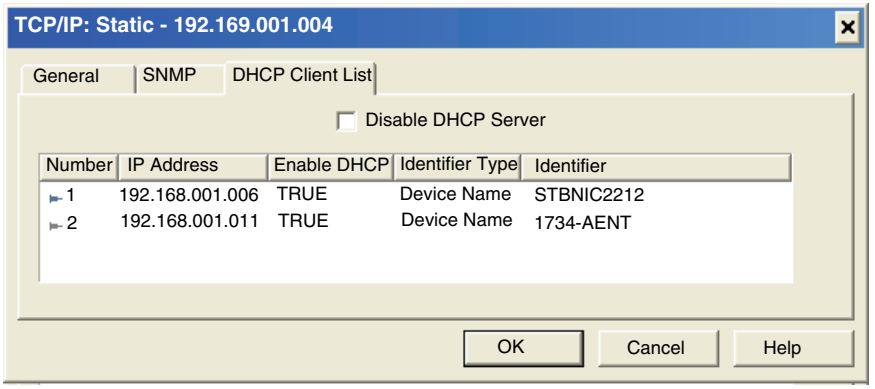
Use this page to:

- enable and disable the DHCP service, and
- view a list of all network devices indicating whether each connected network device does—or does not—subscribe to the DHCP service

Note: The DHCP service is not enabled or disabled for a specific network device in this page. See the topic Enabling the DHCP Service (see *p. 55*), below, for information on how to enable the DHCP service for a specific device.

Viewing the DHCP Client List

The **DHCP Client List** includes a row for each networked EtherNet/IP device, identifying the devices that have subscribed to the DHCP service:

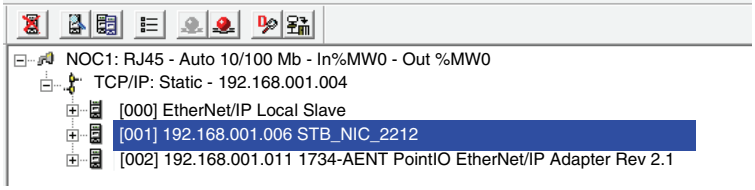
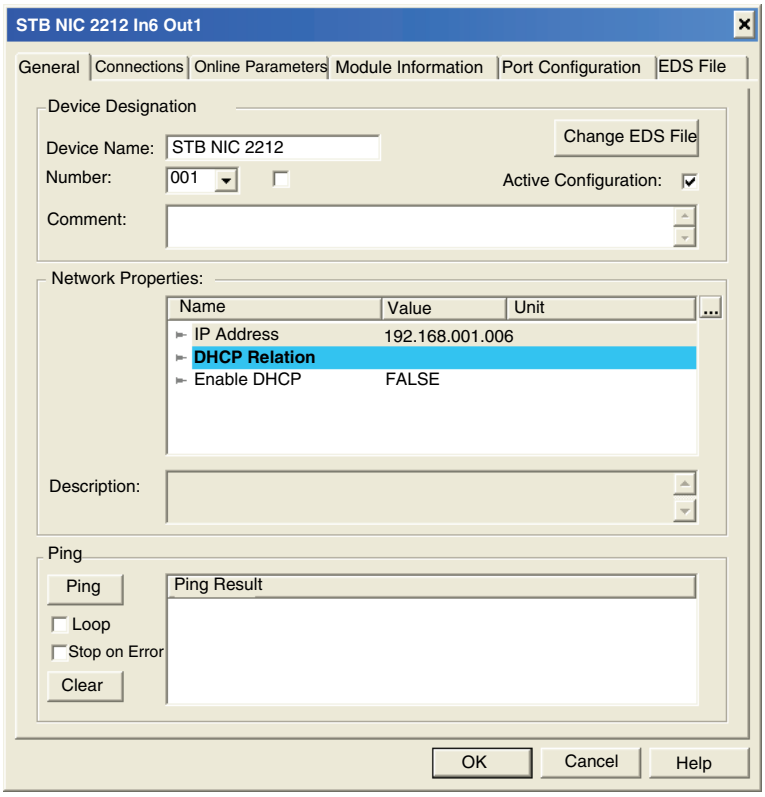


The list contains the following information for each networked device:

Property	Description
Number	The number assigned to the device in the EtherNet/IP configuration tool.
IP Address	The device IP address associated with the device.
Enable DHCP	TRUE indicates that the device subscribes to the DHCP service.
Identifier Type	Indicates the mechanism used by the server to recognize the client (MAC address or DHCP device name).
Identifier	The actual MAC address or DHCP device name.

Enabling the DHCP Service

The DHCP service for an EtherNet/IP device is not enabled in this page. Instead it is enabled and disabled in the remote EtherNet/IP device configuration. To turn on the DHCP service for a specific device, follow these steps:

Step	Action
1	n the Unity Pro EtherNet/IP configuration tool, select the DHCP client device in the **Devices** window. In this example, the selected client is an STB_NIC_2212: 
2	Select **Devices** → **Properties**. The **General** page of the **Properties** window opens for the selected device, indicating the DHCP client service is disabled (the default setting). 

Step	Action	
3	In the **Network Properties** area, under the heading **DHCP Relation**, configure the following properties:	
	Property:	**Action:**
	Enable DHCP	Select **TRUE**.
	DHCP Client Identifier	Select either: • **MAC Address**, or • **Device Name**
	Mac Address/Device Name	Enter a value for either the device name or the MAC Address.
4	Click **OK** to close the device's **Properties** window and save your edits.	

2.5 Configuring the EtherNet/IP Communication Module as an I/O Adapter

At a Glance

Overview

This section describes how to configure the EtherNet/IP communication module as an I/O adapter (local slave). In this role, the module initiates no messages. Instead, it responds to:

- implicit messaging requests from a remote device for periodic data, at the established RPI rate, and
- explicit messaging requests from other EtherNet/IP devices on the network

What's in this Section?

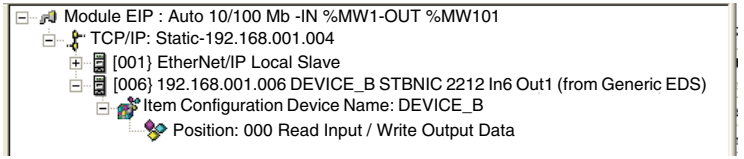
This section contains the following topics:

Topic	Page
Identifying the Local Slave	58
Local Slave Inputs and Outputs	59
Configuring Local Slave Properties: The General page	61

Identifying the Local Slave

Overview

When the Unity Pro EtherNet/IP configuration tool first opens, it automatically includes a Local Slave node in the **Devices** window:



Key Features

Features	Description
Types of connection	• Multicast Point to point is supported in both directions: O->T (Originator to Target) and T->O (Target to Originator) • Real time format 32 bit run/idle header, zero data length, none and heartbeat • Trigger T->O (Target to Originator) cyclic
Sizes	• Input sizes From 1 to 505 bytes • Output sizes From 1 to 509 bytes • Configuration size 0 words (read-only)

Local Slave Inputs and Outputs

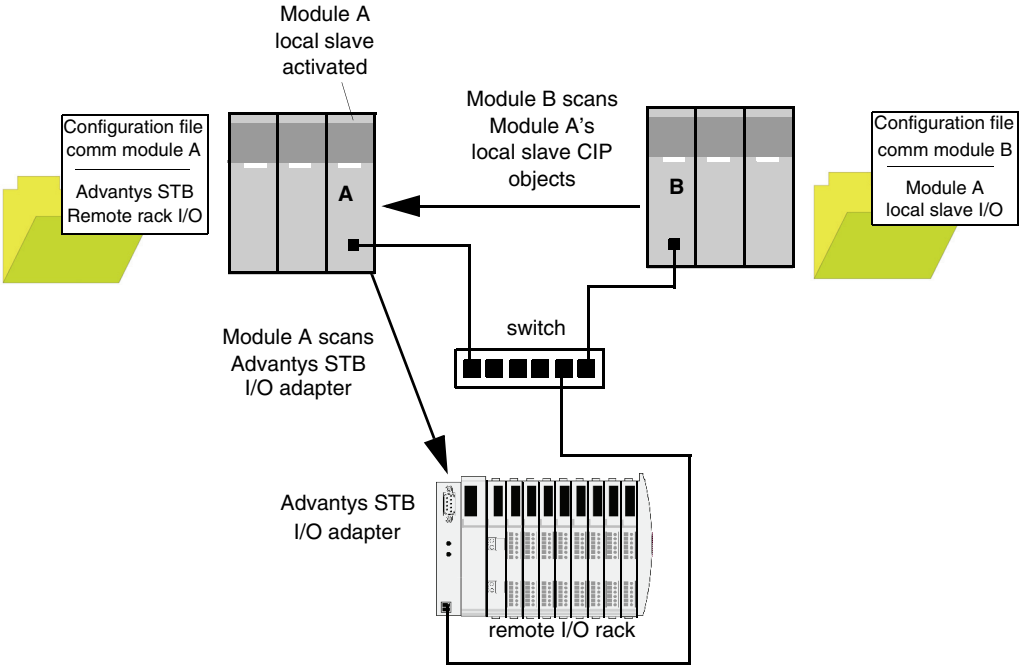
The EtherNet/IP communication module can be used as an I/O adapter. To enable this functionality, select **Active Configuration** in the **Local Slave** properties window (see *p. 61*).

When the local slave function of an EtherNet/IP communication module is enabled, the module's CIP objects (see *p. 167*) are exposed to, and can be accessed by, other EtherNet/IP devices.

The I/O data exchange, between the remote device and the local slave, is configured as part of the remote scanning module's configuration settings.

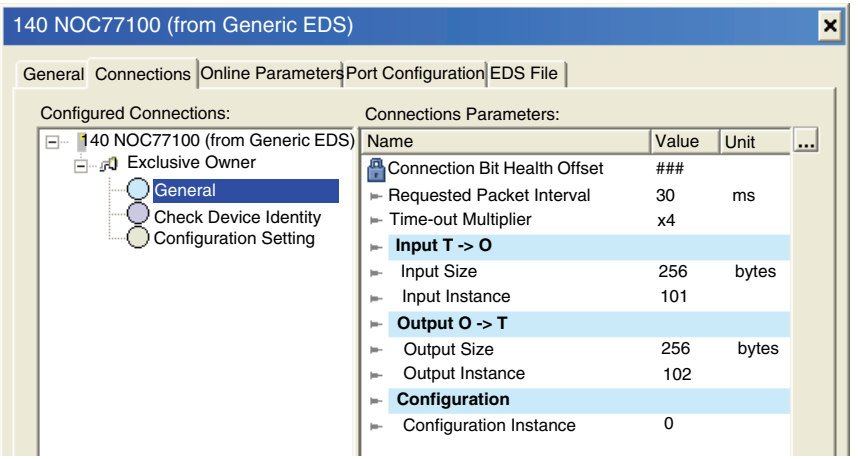
In the following example:

- module A acts as both:
  - an I/O scanner of the Advantys STB I/O adapter, and
  - an I/O adapter, with its CIP objects accessible to remote EtherNet/IP devices
- module B acts as an I/O scanner of the local slave function of module A. Module B can access the exposed CIP objects of module A. The I/O data exchange between module B and module A is configured in the settings for module B.



Configuring the Connection

The I/O data exchange between module B (in its role as an I/O scanner) and module A (in its role as an I/O adapter) is configured in the settings for module B. Do this in the **Connections** page of the remote EtherNet/IP communication module—here, module B—**Properties window**:



Configuring the I/O Items

You can configure input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit dwords, or 32-bit IEEE floating values. The number of items you create depends upon the data type and size of each item.

The process for creating and defining I/O items for the local slave are the same as for any I/O adapter, and depend upon the type of item you wish to create.

For an I/O configuration example, see the how the following I/O items were configured for the STB NIC 2212 network interface module:

- discrete input items (see *p. 100*)
- numeric input items (see *p. 106*)
- discrete output items (see *p. 103*)
- numeric output items (see *p. 109*)

Configuring Local Slave Properties: The General page

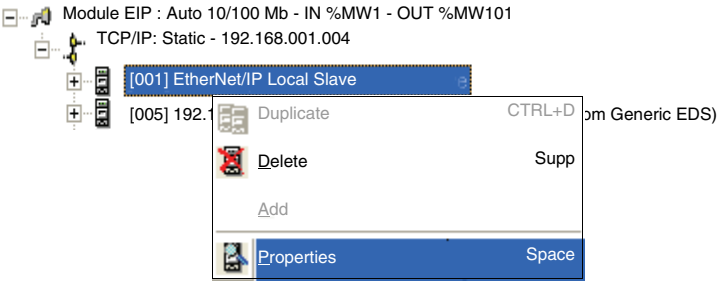
The General Page

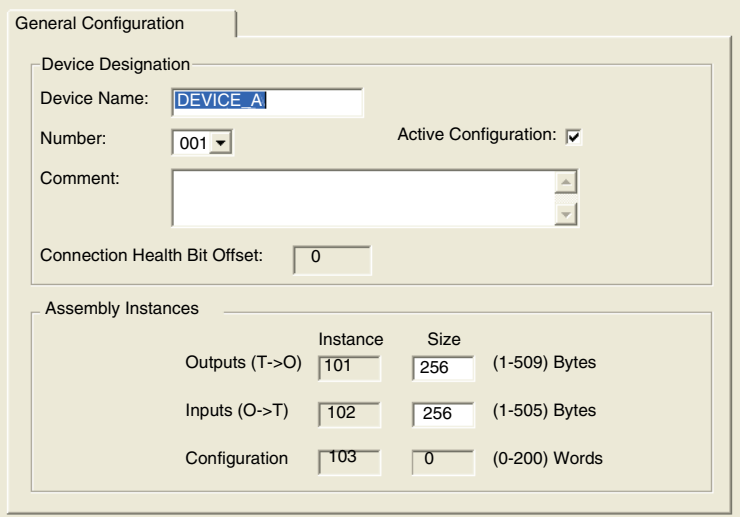
Use the **General** page to configure the EtherNet/IP communication module to serve as an I/O adapter to a remote device.

The following steps describe a sample configuration of the local slave function. Your configuration may be different.

Configuring the Local Slave

To configure the local slave function, follow these steps:

Step	Action
1	In the **Devices** window, right click on the **EtherNet/IP Local Slave** icon, then select **Properties** in the popup menu.  The **General** page of the **Local Slave** properties window opens.

Step	Action
2	Enter settings (refer to the table below) for the following parameters to configure local slave functionality for the EtherNet/IP module.  **Note:** When using explicit messaging to read the EtherNet/IP module's assembly object, be sure to allocate sufficient room for the response, because the size of the response will equal the sum of: the size of the assembly + Reply service (1 byte) + General Status (1 byte) Local slave properties are described, below.
3	The next task is to configure slave inputs and outputs.

Local Slave Properties

The following property settings have been made in this example:.

Setting	Description
Device Designation section:	
Active Configuration	● A **selected** checkbox indicates the local slave service is enabled. ● A **de-selected** checkbox indicates the local slave service is disabled and the current local slave service settings are saved. In this example, this setting is **selected**.
Device Name	Assign the local slave a unique name, consisting of up to 32 characters, including numbers, letters, and the underscore character. In this example, the auto-generated name **DEVICE_A** is accepted.
Number	The unique number—or identifier—assigned to the device. In this example, select the number **001**.
Comment	User-defined free text comment area. 80 characters maximum. In this example, leave blank.
Connection Health Bit Offset	Auto-generated integer (0...127) indicating the offset of the connection's health bit in the status byte array of the input area. **Note:** This setting is auto-generated only when the local slave settings are input and the network configuration is saved.
Assembly Instances section: ● O indicates the originator—or I/O scanner—device ● T indicates the target—or I/O adapter—device	
Outputs T -> O Instance	A read-only value always set to 101.
Outputs T -> O Size	The maximum size reserved for local slave outputs, in bytes. An integer from 0...509. In this example, accept the default of **256**.
Inputs O -> T Instance	A read-only value always set to 102.
Inputs O -> T Size	The maximum size reserved for local slave inputs, in bytes. An integer from 0...509. In this example, accept the default of **256**.
Configuration Instance	A read-only value always set to **103**.
Configuration Size	A read-only value always set to **0**.

Adding Devices to an EtherNet/IP Network

3

At a Glance

Overview

This chapter presents examples of how to add devices to, and how to configure these device for operations on, your EtherNet/IP network.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
3.1	Adding Devices to an EtherNet/IP Network	66
3.2	Adding and Configuring Remote Devices	71
3.3	Configuring the STB NIC 2212	84
3.4	Connecting to Third Party Devices	112

3.1 Adding Devices to an EtherNet/IP Network

Effect of Device Position on Input and Output %MW Memory Addresses

Introduction

The Unity Pro EtherNet/IP configuration tool assigns a %MW memory address to the Inputs and outputs of a remote device, or a local slave, when it is activated.

By default:

- a remote EtherNet/IP device is activated when it is added to an EtherNet/IP network, but
- the EtherNet/IP communication module's local slave function is not activated when it is automatically added to a newly created network—instead, it must be manually activated

This topic describes:

- the effect of activating the local slave on the %MW memory address assignment for inputs and outputs of previously configured EtherNet/IP network
- recommended practices to follow for consistent %MW memory address assignment to remote device inputs and outputs

Activating the Local Slave

When a new network is created, the Unity Pro EtherNet/IP configuration tool adds a local slave node and—by default—assigns it the device **Number** of 000. Because the local slave function is not yet activated, the local slave's inputs and outputs are not initially assigned a %MW memory address.

The following example describes the effect of activating the EtherNet/IP communication module's local slave function after another remote device has already been configured and added to the network.

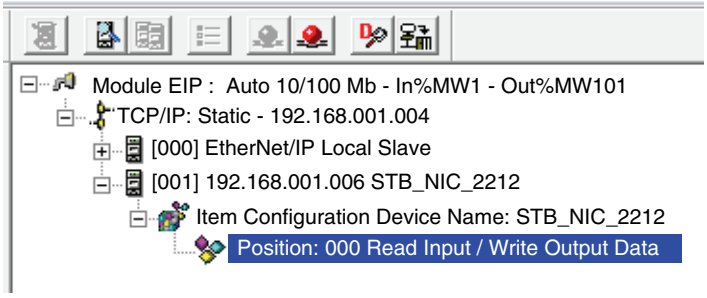
The sample EtherNet/IP network consists only of two nodes:

- the de-activated local slave at position 000
- a single, activated remote device at position 000

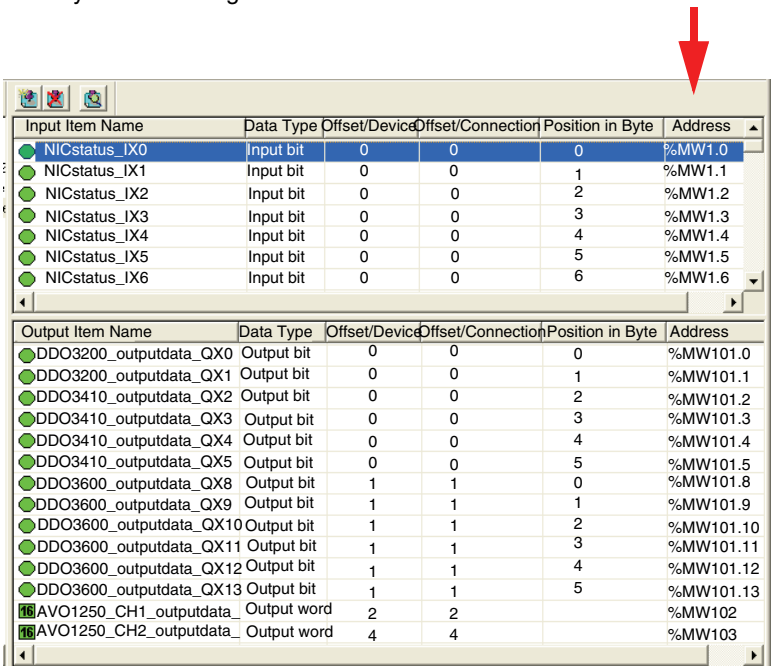
The sample EtherNet/IP network has been configured as follows:

- Total EtherNet/IP network inputs and outputs are set in the **Configuration** page of the EtherNet/IP communication module in Unity Pro:
  - 100 input words are reserved, beginning at %MW01
  - 100 output words are reserved, beginning at %MW101
- Local Slave inputs and outputs:
  - 130 input bytes (65 words) are reserved
  - 130 output bytes (65 words) are reserved
- Remote device inputs and outputs:
  - 40 input bytes (20 words) are reserved
  - 40 output bytes (20 words) are reserved

The **Devices** window of the Unity Pro EtherNet/IP configuration tool displays the network, as follows:



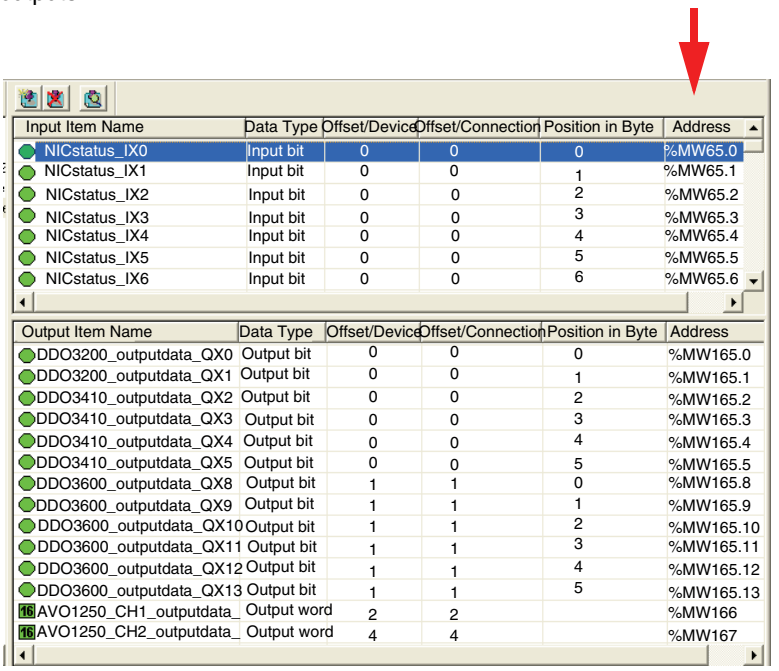
When you select the I/O Items node for the remote device, as indicated above, you display its previously configured input and output items—revealing their %MW memory address assignments:



Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW1.0
NICstatus_IX1	Input bit	0	0	1	%MW1.1
NICstatus_IX2	Input bit	0	0	2	%MW1.2
NICstatus_IX3	Input bit	0	0	3	%MW1.3
NICstatus_IX4	Input bit	0	0	4	%MW1.4
NICstatus_IX5	Input bit	0	0	5	%MW1.5
NICstatus_IX6	Input bit	0	0	6	%MW1.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW101.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW101.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW101.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW101.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW101.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW101.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW101.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW101.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW101.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW101.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW101.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW101.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW102
AVO1250_CH2_outputdata_	Output word	4	4		%MW103

If you next activate the local slave function, by selecting **Active Configuration** in the **General** page of its **Properties** window, then re-open the I/O items node for the remote device, you will see that the %MW memory address assignments have changed—because they now are located behind the local slave's inputs and outputs:



Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW65.0
NICstatus_IX1	Input bit	0	0	1	%MW65.1
NICstatus_IX2	Input bit	0	0	2	%MW65.2
NICstatus_IX3	Input bit	0	0	3	%MW65.3
NICstatus_IX4	Input bit	0	0	4	%MW65.4
NICstatus_IX5	Input bit	0	0	5	%MW65.5
NICstatus_IX6	Input bit	0	0	6	%MW65.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW165.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW165.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW165.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW165.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW165.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW165.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW165.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW165.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW165.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW165.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW165.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW165.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW166
AVO1250_CH2_outputdata_	Output word	4	4		%MW167

This shift of %MW input and output memory address assignments occurs because the assignment of a remote device's—or a local slave's—I/O to a specific %MW memory address depends upon the node's relative position among active nodes in the EtherNet/IP network.

You can avoid this shift in input and output %MW memory addresses. When you activate the local slave function, be sure to change the local slave's device **Number** from the default value of 000 to a value larger than the device number of the last device in the network.

In this example, setting the local slave's device **Number** to **002** would preserve the remote device's original %MW input and output memory address assignments.

Recommended Practices

To avoid the problem of shifting input and output %MW memory address assignments, consider the following recommended practices when developing your application:

- As described above, when activating the local slave function of an EtherNet/IP communication module, change the local slave device **Number** from its default value of 000 to a value larger than the device number for the last device in your network.
- When adding a new remote device to your EtherNet/IP network, always add it to the end of the device list and assign it a device **Number** larger than any other device number on your network.
- When configuring function blocks in Unity Pro, do not directly assign input and output pins to a specific %MW memory address. Instead, assign input and output pins to the derived data types and variables automatically created by Unity Pro.

3.2 Adding and Configuring Remote Devices

At a Glance

Overview

This section describes how to:

- add a generic device to your EtherNet/IP network
- configure properties for the generic device
- save, transfer and re-use Unity Pro project files that include EtherNet/IP module settings

What's in this Section?

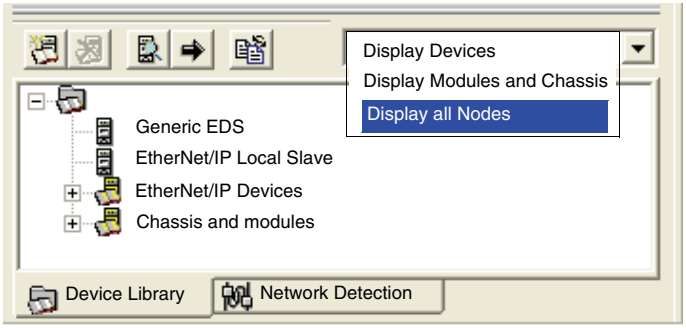
This section contains the following topics:

Topic	Page
Device Library	72
Add an EDS File to the Device Library	74
Adding A Remote Device	76
Configuring Remote Device Properties	78
Managing Project Files	82

Device Library

Overview

The Unity Pro EtherNet/IP configuration tool includes a **Device Library**, located in the lower left part of the configuration tool's main window. The **Device Library** is a repository of both generic and device-specific EDS files. Each EDS file defines a device, chassis, or module that you can add to your EtherNet/IP network configuration.



Both the Generic EDS node and the EtherNet/IP Local Slave node describe generic devices and cannot be deleted.

Click on the ⊞ icon to expand the device list and display the items of the selected type.

Functions

Use the **Device Library's** toolbar controls to perform the following tasks:

Function	Icon	Description
Add an EDS File		Opens the Add an EDS File wizard (see *p. 74*), which steps you through the process of adding a new EDS file to the **Device Library**.
Delete a device from the Device Library list		Deletes the selected device, chassis, or module from the Device Library list, but retains the associated EDS File in your PC's EDS File folder. You can use the Add an EDS File button to restore the deleted device to the list. **Notes:** • Do not delete a device that has been added to your EtherNet/IP network. • You can delete only device-specific devices; you cannot delete a generic device.
Display device properties		Opens the properties window for the selected device. In the properties window, click the **View or Print EDS File...** button to display the EDS File in a text file window. In the text file window, select **File** → **Print** to print the contents of the EDS file.
Insert a device into your EtherNet/IP configuration		Inserts the selected device to the last position in your EtherNet/IP design. **Note:** You cannot manually insert a chassis or module into the configuration. These are added during the configuration of modular devices.
Sort the Device Library list		Opens the **Sort Device Library** window, where you can select a sort order for the devices, chassis, and modules displayed in the **Device Library**.
Filter the Device Library list	List	Click inside the drop-down list to display and select one of the following filtering options: • **Display Devices**: displays only devices—module and chassis entries are filtered out • **Display Modules and Chassis**: displays both chassis and for modules—devices are filtered out • **Display all Nodes**: displays devices, modules and chassis.

Add an EDS File to the Device Library

Overview

The Unity Pro EtherNet/IP configuration tool includes an **EDS Management** wizard that you can use to add one or more EDS files to the **Device Library**. The wizard presents a series of instruction screens that:

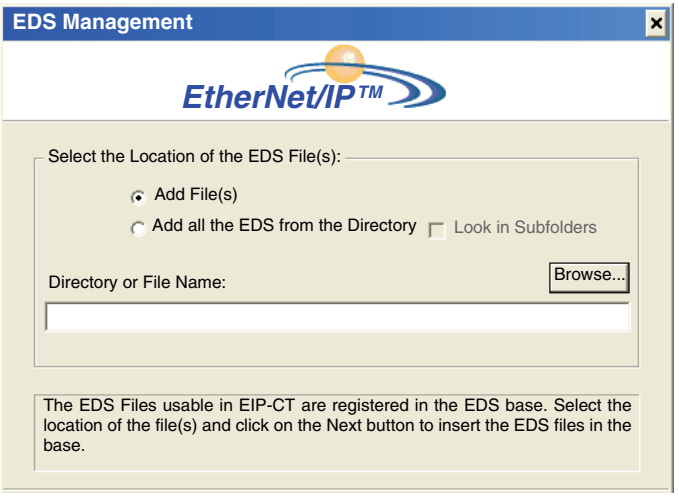
- simplify the process of adding EDS files to the **Device Library**, and
- provide a redundancy check in case you attempt to add duplicate EDS files to the **Device Library**

Select **Devices** → **Options...** to open the **Display Options** window, where you can enable/disable messages indicating the EDS file you are adding is a duplicate—or a different version—of an existing EDS file.

> **Note:** The Unity Pro EtherNet/IP configuration tool contains a library of EDS files registered with the ODVA. This library includes EDS files for products not manufactured or sold by Schneider Electric. The non-Schneider Electric EDS files are identified in the Unity Pro EtherNet/IP Configuration Tool library. Please contact the identified device's manufacturer for inquiries regarding the corresponding non-Schneider Electric EDS files.

Adding EDS Files

To add one or more EDS files to the **Device Library**:

Step	Action
1	Do one of the following: ● in the **Device Library**, click the Add button , or ● select **Library** → **Add** Page 1 of the wizard opens.
2	Click **Next**. Page 2 of the wizard opens: 

Step	Action
3	In the **Select the Location of the EDS File(s)** section, select either: ● **Add File(s)**, to add one or more EDS files you will individually select, or ● **Add all the EDS Files from the Directory**, to add all files from a folder you will select. ● Select **Look in Subfolders** to also add EDS files in subfolders beneath the folder you select
4	Click the **Browse** button. The **Open** dialog opens.
5	Use the **Open** dialog to navigate to and select: ● one or more EDS files, or ● a folder containing EDS files
6	After you have made your selection9s), click **Open**. The dialog closes and your selection appears in the **Directory or File Name** field.
7	Click **Next**. The wizard compares the selected EDS files against existing files in the **Device Library**.
8	(Conditional) If one or more selected EDS files are duplicates and if notice of redundant files is enabled in the **Display Options** dialog, a **File Already Exists** message displays. **Close** the message.
9	Page 3 of the wizard opens indicating the Status of each device you attempted to add: ● a green check mark indicates the EDS file can be added ● a blue informational icon indicates a redundant file ● a red check mark indicates an invalid EDS file (Optional) Select a file in the list, then click **View Selected File** to open it.
10	Click **Next** to add the non-duplicate files. Page 4 of the wizard opens, indicating the action is complete.
11	Click **Finish** to close the wizard.

Adding A Remote Device

Overview

The Device Library consists of two types of entries:

Entry	Defined by
generic	A device without an associated EDS File. In the Device Library, generic devices include: ● Generic EDS ● EtherNet/IP Local Slave
EDS File specific	A device, module, or chassis defined by a unique vendor-created EDS File. In the Device Library, these devices appear beneath the branches: ● EtherNet/IP Devices ● Chassis and modules

You can add both generic devices or devices with a specific EDS File to your EtherNet/IP network.

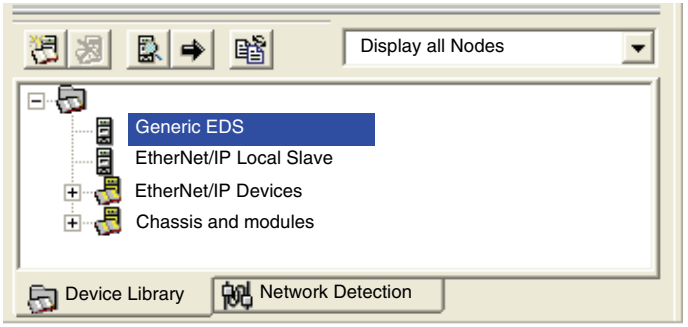
When you add:

- a device that is associated with a specific EDS File, the Unity Pro EtherNet/IP configuration tool recognizes the device and automatically performs much of the device configuration for you
- a generic device, you need to manually perform all device configuration

In the following example, a generic device is added to an EtherNet/IP network.

Adding a Generic Remote Device

To add a generic remote device to your EtherNet/IP network, follow these steps:

Step	Action
1	In the **Device Library**, select Generic EDS (see below): 
2	Click the Insert button. Two things occur simultaneously: ● a new generic device is added to the end of the EtherNet/IP network configuration, and ● the Generic EDS properties window opens for editing.
3	Refer to the topic Configuring a Generic Remote Device (see *p. 78*) for additional instructions on configuring the generic device.

Configuring Remote Device Properties

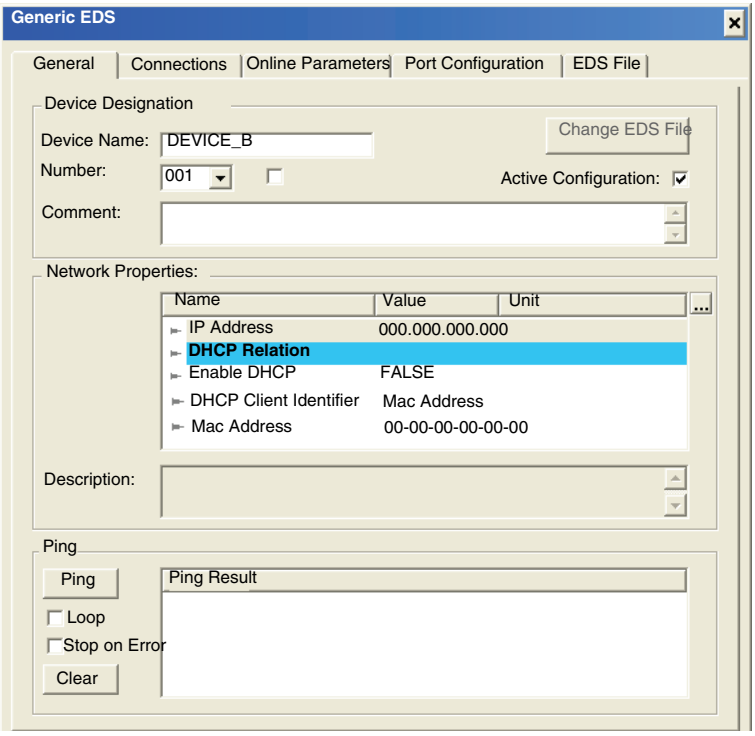
Overview

When a generic device is added to an EtherNet/IP network, the Unity Pro EtherNet/IP configuration tool automatically opens its properties window for immediate configuration. When operating offline, the properties window consists of the following 5 pages. Only the first two of these pages need to be configured:

In this page...	Do the following...
General	Enter configuration settings, as described below.
Connections	Enter configuration settings, as described below.
Online Parameters	Not accessible offline. No configuration required.
Port Configuration	Not accessible offline. No configuration required.
EDS File	(Read-only page - no configuration required)

Configuring the General Page

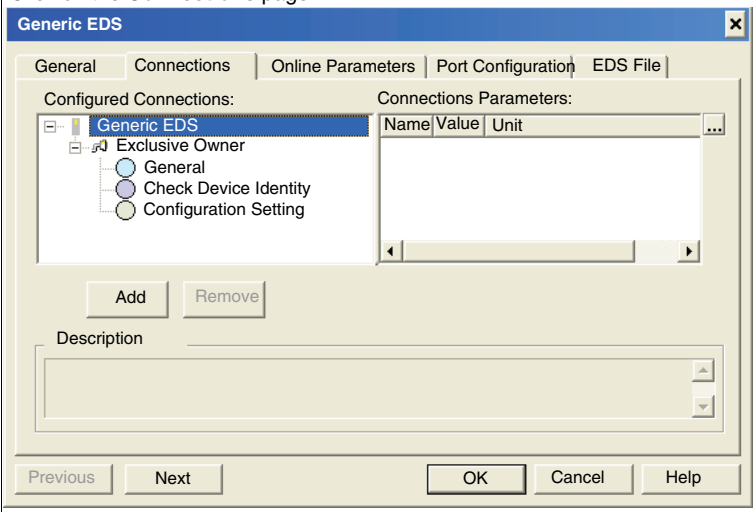
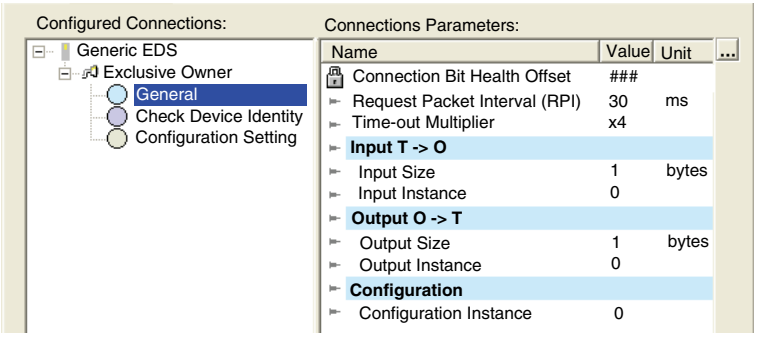
To configure the **General** page:

Step	Action
1	Click on the **General** page: 

Step	Action	
2	In the **General** page, edit the following settings:	
	Device Name	The label for the remote device in the EtherNet/IP device list. Either: ● type in a unique name using letters, numbers and the underscore character (_), or ● accept the auto-generated name (DEVICE_N}
	Number	The relative position in the EtherNet/IP device list. Either: ● accept the default (i.e. the next available number), or ● select a different number from the drop-down list
	Link Parameters	Select this setting to: ● lock the **IP Address** setting, and ● set the last octet of the **IP Address** equal to the value selected in the **Number** parameter De-select this setting to unlock the **IP Address** setting.
	Active Configuration	Select this setting to include this remote device in EtherNet/IP network communications. De-select this setting to exclude this device from network communications, but save the device's configuration settings.
	IP Address	The IP Address of this remote device. This setting is: ● editable, when the **Link Parameters** field is de-selected ● locked, when the **Link Parameters** field is selected By default: ● the first 3 octet values equal the first 3 octet values of the EtherNet/IP module's IP address ● when the **Link Parameters** field is selected, the last octet value equals the value selected in the **Number** parameter
	Enable DHCP	TRUE activates the DHCP client in this remote device. On startup, this device requests its IP address from a DHCP server. **Note**: the EtherNet/IP module can be configured to act as a DHCP server.
	DHCP Client Identifier	If the DHCP client is enabled, select the identifier the DHCP server will use to recognize this remote device: ● **MAC Address** ● **Device Name**
	Mac Address/ Device Name	Type in the value of the DHCP client identifier. **Note:** The **Device Name** referenced here is not the same as the Device Name described in the first row of this table.

Configuring the Connections Page

To configure the Connections page:

Step	Action
1	Click on the **Connections** page: 
2	In the **Configured Connections** list, click on **General** to display the general connection settings in the **Connection Parameters** list, shown below: 

Step	Action	
3	In the **Connections** page, edit the following general connection settings:	
	Connection Health Bit Offset	(read-only)
	Request Packet Interval (RPI)	The refresh period for this I/O connection. Value range: 2...65535 ms Default = 30 ms
	Time-out Multiplier	The value, multiplied against the RPI rate, which triggers an inactivity timeout. Value list: 4, 8, 16, 32, 64, 128, 256, 512 Default: 4
	Input Size (in bytes)	The number of bytes reserved for input data, in bytes. Value range: 1...509 Default: 1
	Input Instance	The instance identifier for inputs: **101**.
	Output Size (in bytes)	The number of bytes reserved for output data, in bytes. Value range: 1...505 Default: 1
	Output Instance	The instance identifier for outputs: **102**.
	Configuration Instance	The instance identifier for configuration data: **103**.
	Note: The **Input Size** and **Output Size** parameter settings are determined by the size—in bytes—of the input data and output data sections of your specific application.	
4	Click **OK** to save your settings and close the **Properties** window. The next step is to configure I/O settings. For an example of I/O configuration for a generic remote device, see how the following I/O items were configured: ● discrete input items (see *p. 100*) ● discrete output items (see *p. 103*) ● numeric input items (see *p. 106*) ● numeric output items (see *p. 109*)	

Managing Project Files

Overview

Managing Unity Pro project files that contain EtherNet/IP module settings includes:

- saving project files as either:
  - Unity Pro Archived Application Files (*.STA)
  - Unity Pro project files (*.STU)
- opening saved project files
- transferring files

Note: To transfer Unity Pro project files, follow the steps set forth below.
Do not use the following Unity Pro commands to transfer a Unity Pro project file that contains EtherNet/IP settings:
- project transfer command: **PLC** → **Transfer Project from PLC**
- export project command: **File** → **Export Project...**

Creating Unity Pro Archive (*.STA) Files

Unity Pro project files, containing EtherNet/IP module settings, can be transferred within the Unity Pro application only as Unity Pro Archived Application Files (*.STA). To save a Unity Pro project file as a Unity Pro Archived Application File (*.STA) suitable for transfer and reuse, follow these steps:

Step	Action
1	Build the Unity Pro project. Select: **Build** → **Rebuild All Project**.
2	Download the rebuilt Unity Pro project file to the PLC. Select: **PLC** → **Transfer Project to PLC**. The taskbar should indicate **EQUAL**.
3	Go offline. Select: **PLC** → **Disconnect**.
4	Select **File** → **Save Archive...** The **Save Archive** window opens.
5	In the **Save Archive** window: • type a **File name** • navigate to a location to store the archived project file • click **Save**. Unity Pro creates a Unity Pro Archived Application File (*.STA).

Opening a Unity Pro Archive (*.STA) File

After a Unity Pro Archived Application File has been saved, you can transfer it (like any file), then re-open it in the same version of Unity Pro. To re-open an archived project file:

Step	Action
1	Select **File** → **Open**. The **Open** dialog opens.
2	In the **Open** dialog, select **Unity Pro Archived Application Files (*.STA)** as the **Files of type**.
3	In the **Look in** drop down box, navigate to the location of the archived Unity Pro archive file that you want to open.
4	Select the file and click **Open**. Unity Pro opens the archived Unity Pro project file.

Transferring Unity Pro Project (.STU) Files

You can copy, paste, and transfer a Unity Pro project (*.STU) file as you would any file, using the tools and commands available in Windows Explorer.

A saved Unity Pro project (*.STU) file can be re-opened only by the same version of Unity Pro software that saved it.

3.3 Configuring the STB NIC 2212

At a Glance

Overview

This section presents a sample configuration of an STB NIC 2212 EtherNet/IP network interface module, and adds it to a Unity Pro project.

Note: The instructions in this chapter describe a single, specific device configuration example. Refer to the Unity Pro EtherNet/IP configuration tool help file for additional information about alternative configuration choices.

The following example extends the sample configuration of the EtherNet/IP communications network—described in the previous chapter—where you:

- created a project
- added a power supply module, CPU and EtherNet/IP communication module to the project
- configured the EtherNet/IP communication module

What's in this Section?

This section contains the following topics:

Topic	Page
Setting Up Your Network	85
Automatically Detect and Add the STB NIC 2212	87
Configuring STB NIC 2212 Properties	88
Connecting to the Advantys STB Island	92
Configuring I/O Items	97

Setting Up Your Network

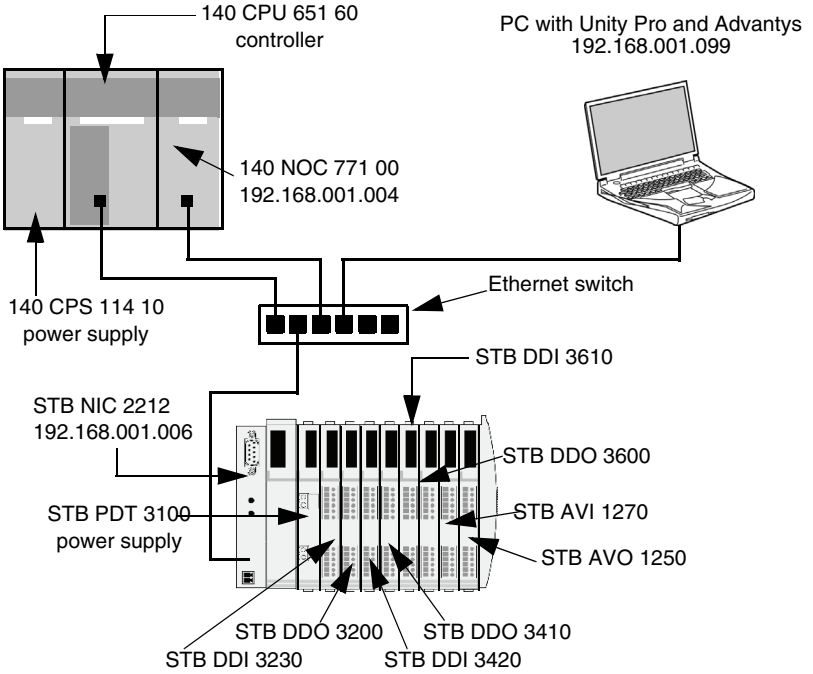
Overview

This sample network includes the following hardware and software:

- a controller rack with:
  - 140 CPS 114 10, 115/230 VAC power supply
  - 140 CPU 651 60 controller
  - 140 NOC 771 00, EtherNet/IP communication module
- a remote STB Advantys island with:
  - STB NIC 2212 Ethernet network interface module
  - STB PDT 3100 power distribution module
  - STB DDI 3230 2 pt digital input module
  - STB DDO 3200 2 pt digital output module
  - STB DDI 3420 4 pt digital input module
  - STB DDO 3410 4 pt digital output module
  - STB DDI 3610 6 pt digital input module
  - STB DDO 3600 6 pt digital output module
  - STB AVI 1270 2 pt analog input module
  - STB AVO 1250 2 pt analog output module
- a PC running both Unity Pro (version 4.0 or higher) and Advantys configuration software (version 4.0 or higher)
- an Ethernet switch connected to the above EtherNet/IP devices with twisted pair Ethernet cable and RJ45 connectors (It is strongly recommended that you use a managed switch that supports the IGMP protocol.)

Network Topology

The network example topology looks like this:



To re-create this example, be sure to:

- use the IP addresses for your own configuration's:
  - PC
  - 140 NOC 771 00 EtherNet/IP communication module
  - STB NIC 2212 network interface module
- check all wiring

Note: Unity Pro software running in the PC is used to configure the CPU 651 60 controller. In this example, the PC is indirectly wired to the CPU's Ethernet port via the Ethernet switch. Alternatively, you could bypass the switch and directly wire the PC to either the CPU's Modbus or USB ports.

Automatically Detect and Add the STB NIC 2212

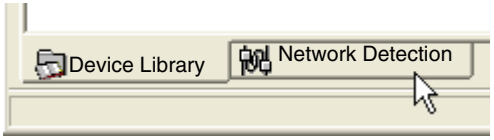
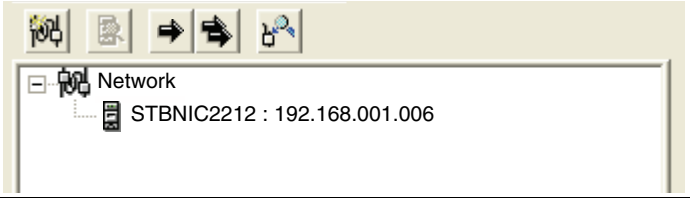
Overview

Use the Unity Pro EtherNet/IP configuration tool to automatically detect the STB NIC 2212 module, then add it to your project.

Note: The STB NIC 2212 module must be active online with a valid IP address before you can detect it then add it to your project. You can assign an IP address using a DHCP or BOOTP server, or use the MAC-generated (default) IP address.

Detecting and Adding Network Devices

To automatically detect the STB NIC 2212, then add it to your project, follow these steps:

Step	Action
1	Launch the configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.
2	In the configuration tool, begin on-line operations by clicking the **Go Online** button .
3	Click on the **Network Detection** tab to enable automatic network detection: 
4	Click the **Read Network Configuration** toolbar button . The configuration tool searches the network for EtherNet/IP devices, classifies them using the device EDS file, then lists the EtherNet/IP devices it detects. 
5	Select the **STB NIC 2212** in **Network Detection** window.
6	Click the **Insert in Configuration** button . The properties window opens, where you can configure the STB NIC 2212.

Configuring STB NIC 2212 Properties

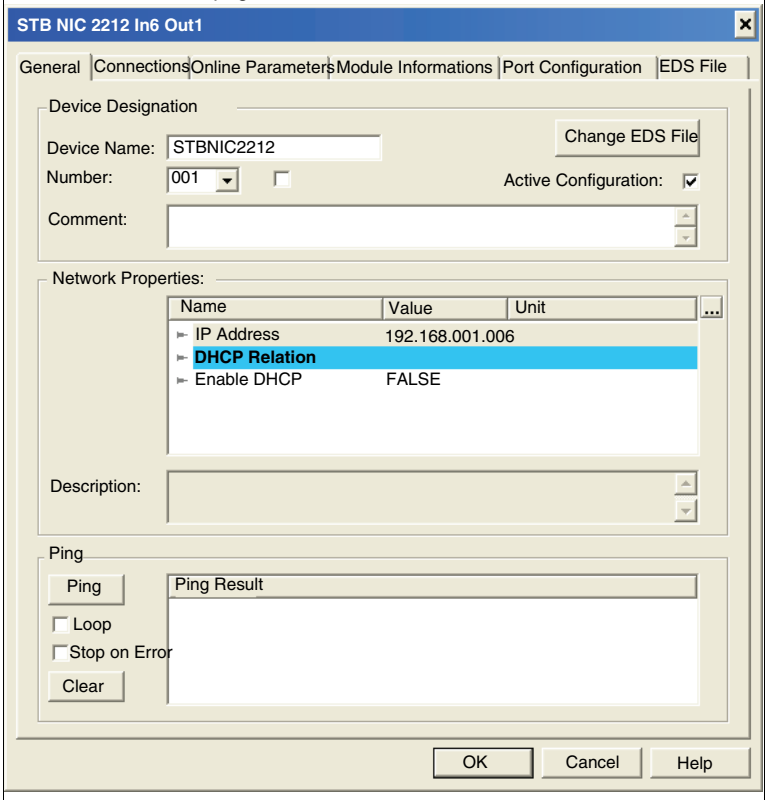
Overview

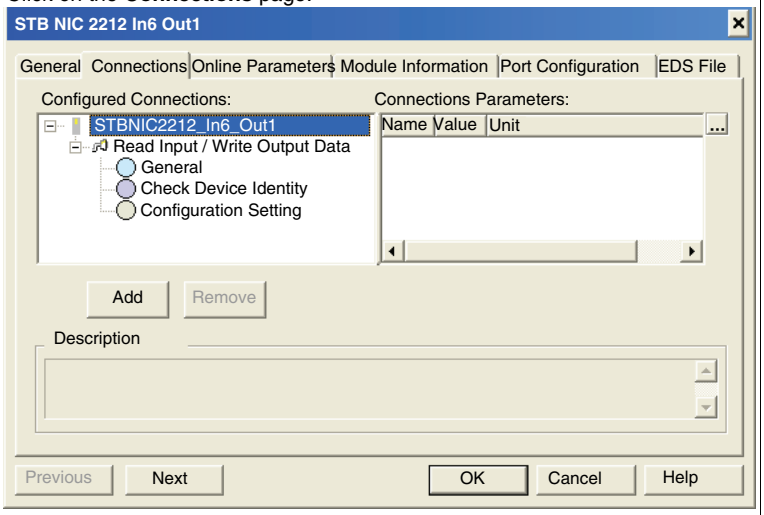
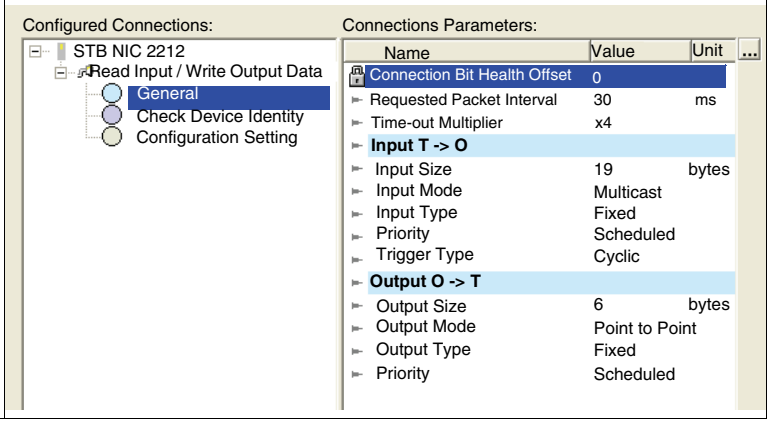
The STB NIC 2212 network interface module properties window presents the following tabbed pages. Only some of these pages need to be edited for this example:

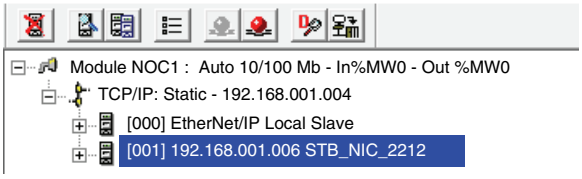
In this page...	Do the following...
General	• input device name • configure IP address • add the device to the project configuration
Connections	• configure the requested packet interval (RPI) • specify the size and location of inputs and outputs
Online Parameters	Accept the default settings, if any.
Module Informations	(Read-only page - no configuration required)
Port Configuration	(Read-only page - no configuration required)
EDS File	(Read-only page - no configuration required)

Configuring the STB NIC 2212

The following settings are used in this sample configuration. Be sure to use settings that are appropriate for your actual application:

Step	Action	
1	Click on the **General** page:	
		
2	In the **General** page, edit the following settings:	
	Device Name	**STBNIC2212**
	Number	The relative position in the EtherNet/IP device list. For this example, select **001**.
	Active Configuration	Be sure this checkbox is selected.
	IP Address	**192.168.001.006**

Step	Action
3	Click on the **Connections** page: 
4	In the **Configured Connections** list, click on **General** to display the general connection settings in the **Connection Parameters** list, shown below: 

Step	Action	
5	In the **Connections** page, edit the following general connection settings:	
	Request Packet Interval	**30 ms**
	Input Size (in bytes)	**19 bytes**
	Input Instance	**101**
	Output Size (in bytes)	**6 bytes**
	Output Instance	**102**
	Note: The **Input Size** and **Output Size** parameter settings are determined by the size—in bytes—of the input data and output data sections of the Advantys island's Fieldbus Image.	
6	Click **OK** to save your settings and close the **Properties** window. A node is added to the project configuration in the **Devices** window, below:  The next step is to configure I/O settings.	

Connecting to the Advantys STB Island

Overview

In this example, you will use the Advantys configuration software running on your PC to:

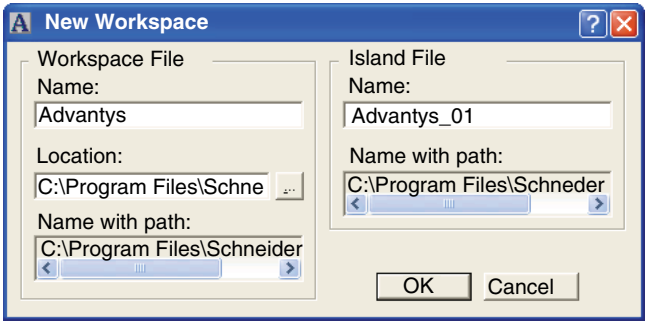
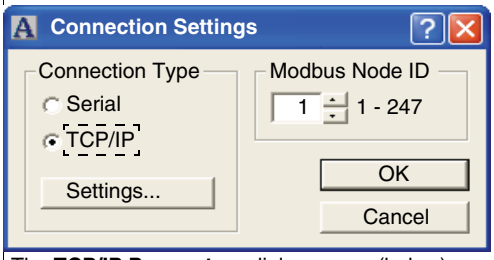
- connect the Advantys configuration software to the STB NIC 2212 and the 8 I/O modules that comprise the Advantys STB island
- upload Advantys STB island configuration to the Advantys configuration software in your PC
- display a fieldbus image for the Advantys STB island showing the relative location of:
  - status information
  - input data
  - output data

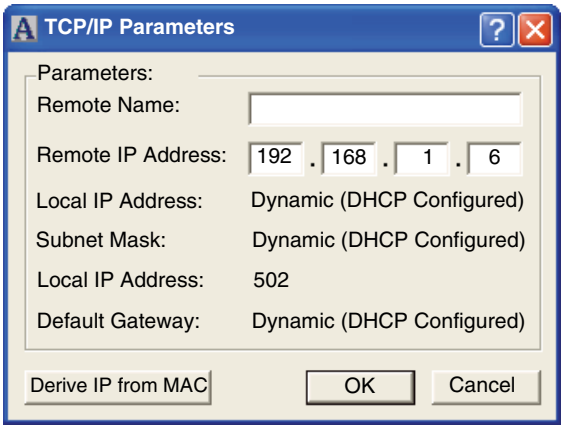
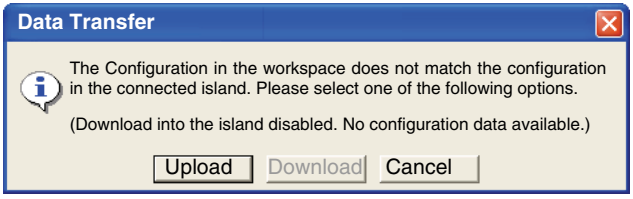
Using the data presented in the fieldbus image, you can use the Unity Pro EtherNet/IP configuration tool to create input and output items that map to specific status, input, output, and output echo data.

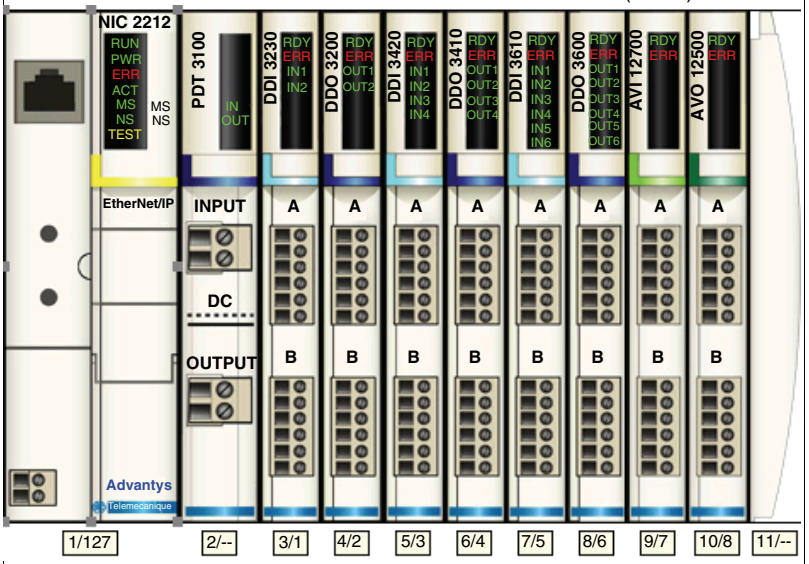
Note: Before proceeding with the following instructions, be sure you have auto-configured the Advantys STB island by pressing the **RST** button on the front of the STB NIC 2212 module.

Making the Connection

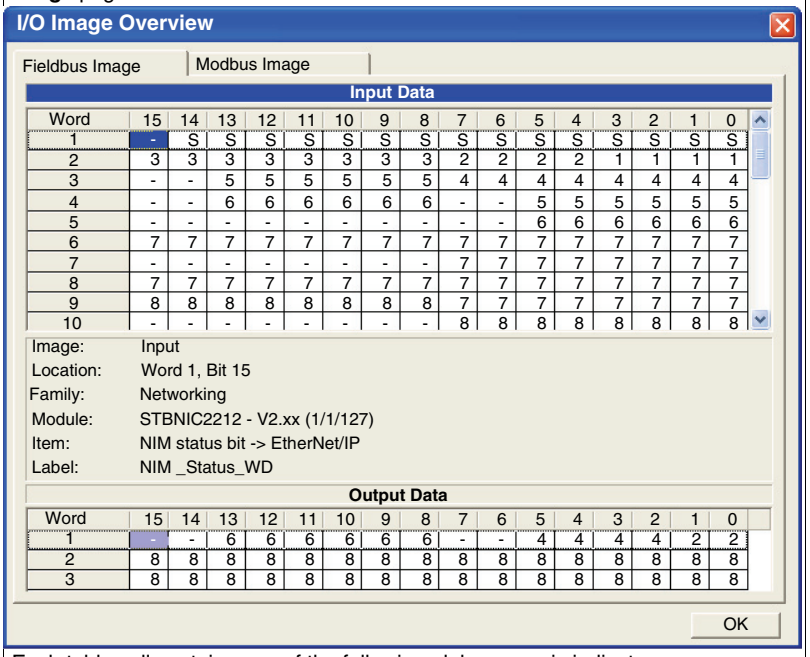
To connect to the STB NIC 2212 and I/O modules using the Advantys configuration software:

Step	Action
1	Startup the Advantys configuration software on your PC. A dialog opens displaying available project types.
2	Select **STB**. A choice of language dialog opens.
3	Select your choice of language.
4	Select **File** → **New Workspace**. The **New Workspace** window opens (below).
5	For this example, type in the following field values: ● for the field **Workspace File** type in **Advantys** ● for the field **Island File** type in **Advantys_01** 
6	Click **OK**. The Advantys configuration software displays an empty DIN rail in the center of the screen.
7	Select **Online** → **Connection Settings**. The **Connection Settings** window opens (below).
8	In the **Connection Settings** window, accept the **Modbus Node ID** default setting of **1**, select **TCP/IP**, and click the **Settings...** button:  The **TCP/IP Parameters** dialog opens (below).

Step	Action
9	In the **Remote IP Address** field, type in the IP address for the STB NIC 2212, in this example: **192.168.1.6**.  **Note:** Use the mouse to move between octets, and do not type in any leading zeroes. For example, do not type in 192.168.001.006.
10	Click **OK** to close the **TCP/IP Parameters** dialog, and click **OK** again to close the **Connection Settings** dialog.
11	Select **Online** → **Connect**. The **Data Transfer** dialog opens (below): 

Step	Action
12	Select **Upload** in the **Data Transfer** dialog. The island workspace is populated with island data and shows the STB NIC 2212 and all island modules (below):  **Note:** A box appears beneath each module containing one or two integers—for example 3/1 . These integers serve the following purpose: ● The left-side integer (3 in this example) identifies the module's physical position—left to right—among all modules in the rack. ● The right-side integer (1 in this example) identifies the module's relative position—left to right—among only data producing/receiving modules. If the module is not a data producing/receiving module (e.g. a power supply, or end of segment module) no right-side integer appears.

Step	Action
13	Select **Island** → **I/O Image Overview**. The **I/O Image** window opens to the **Fieldbus Image** page:



Each table cell contains one of the following alpha-numeric indicators:

- **S** indicates a status bit for the STB NIC 2212 network interface module
- **an integer** identifies the relative position—from left to right—of a data producing/receiving module with input or output data in that cell. For example:
  - the STB DDI 3230 input module is the first data producing or receiving module in the rack; its data is designated by the integer 1 in bits 0 - 3 of word 2 in the **Input Data** table
  - the STB DDO 3600 output module is the sixth data producing module in the rack; its status and output echo data is designated by the integer 6 in bits 8 - 13 of word 4 and in bits 0 - 5 of word 5 in the **Input Data** table; its output data is designated by the integer 6 in bits 8 - 13 of word 1 in the **Output Data** table

Notes:

Select a cell in either the **Input Data** or **Output Data** tables to display—in the middle of the page—a description of the cell data and its source module.

Convert the size of the **Input Data** table and the **Output Data** table from words to bytes (i.e. divide by 2), then use that data as the values for the **Input Size** (19) and **Output Size** (6) parameters when configuring the remote device's general connection properties (see *p. 89*).

Configuring I/O Items

Overview

The final task in this example is to add I/O items to the configuration of the STB NIC 2212 and its 8 I/O modules. To accomplish this:

- use the Advantys configuration software to identify the relative position of each I/O module's inputs and outputs
- use the Unity Pro EtherNet/IP configuration tool to create input and output items, defining each item's:
  - name
  - data type
- identity the address assigned to each new input and output item using the Unity Pro EtherNet/IP configuration software

I/O Item Types and Sizes

The goal is to create a collection of input items and output items that equal the input size and output size specified in the STB NIC 2212 Connection properties page. In this example, items need to be created for:

- 19 bytes of inputs
- 6 bytes of outputs

The Unity Pro EtherNet/IP configuration tool provides great flexibility in creating input and output items. You can create input and output items in groups of 1 or more single bits, 8-bit bytes, 16-bit words, 32-bit dwords, or 32-bit IEEE floating values. The number of items you create depends upon the data type and size of each item.

In the sample project, the following items were created:

- discrete bits for digital inputs and outputs
- 8-bit bytes or 16-bit words for analog inputs and outputs

Mapping Input and Output Items

Use the **Fieldbus Image** page of the **I/O Image Overview** window in the Advantys configuration software to identify the number and type of I/O items you need to create, as follows:

Step	Action
1	In the Advantys configuration software, select **Island** → **I/O Image Overview**. The **I/O Image** window opens to the **Fieldbus Image** page.
2	Select the first cell (word 1, cell 0) in the **Input Data** table to display—in the middle of the page—a description of the cell data and its source module.
3	Make a note of the word, bit(s), module and item information for that cell.
4	Repeat steps 2 and 3 for each cell containing either an S or an integer.

Note: The Fieldbus Image presents input and output data in the form of 16-bit words (starting with word 1). You need to rearrange this data for the Unity Pro EtherNet/IP configuration tool, which presents the same data in the form of 8-bit bytes (starting with byte 0).

This process yields the following tables of input and output data:

Input Data:

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	Bit(s)	Byte	Bit(s)		
1	0-15	0	0-7	NIC 2212	NIC status
		1	0-7		
2	0-1	2	0-1	DDI 3230	input data
	2-3		2-3	DDI 3230	input status
	4-5		4-5	DDO 3200	output data echo
	6-7		6-7	DDO 3200	output status
	8-11	3	0-3	DDI 3420	input data
	12-15		4-7	DDI 3420	input status
3	0-3	4	0-3	DDO 3410	output data echo
	4-7		4-7	DDO 3410	output status
	8-13	5	0-5	DDI 3610	input data
	14-15		6-7	NA	not used
4	0-5	6	0-5	DDI 3610	input status
	6-7		6-7	NA	not used
	8-13	7	0-5	DDO 3600	output data echo
	14-15		6-7	NA	not used

Advantys Fieldbus Image		Unity Pro EIP Items		STB Module	Description
Word	Bit(s)	Byte	Bit(s)		
5	0-5	8	0-5	DDO 3600	output status
	6-15	8	6-7	NA	not used
		9	0-7		
6	0-15	10	0-7	AVI 1270	input data ch 1
		11	0-7		
7	0-7	12	0-7	AVI 1270	input status ch 1
	8-15	13	0-7	NA	not used
8	0-15	14	0-7	AVI 1270	input data ch 2
		15	0-7		
9	0-7	16	0-7	AVI 1270	input status ch 2
	8-15	17	0-7	AVO 1250	output status ch 1
10	0-7	18	0-7	AVO 1250	output status ch 2
	8-15	NA	NA	NA	not used

Output Data:

Advantys Fieldbus Image		Unity Pro EIP Items		Module	Description
Word	Bit(s)	Byte	Bit(s)		
1	0-1	0	0-1	DDO 3200	output data
	2-5		2-5	DDO 3410	output data
	6-7		6-7	NA	not used
	8-13	1	0-5	DDO 3600	output data
	14-15		6-7	NA	not used
2	0-15	2	0-7	AVO 1250	output data ch 1
		3	0-7		
3	0-15	4	0-7	AVO 1250	output data ch 2
		5	0-7		

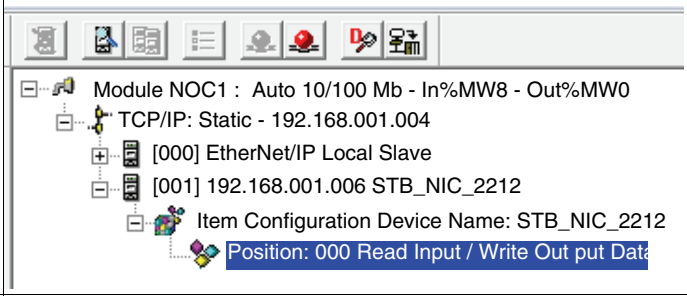
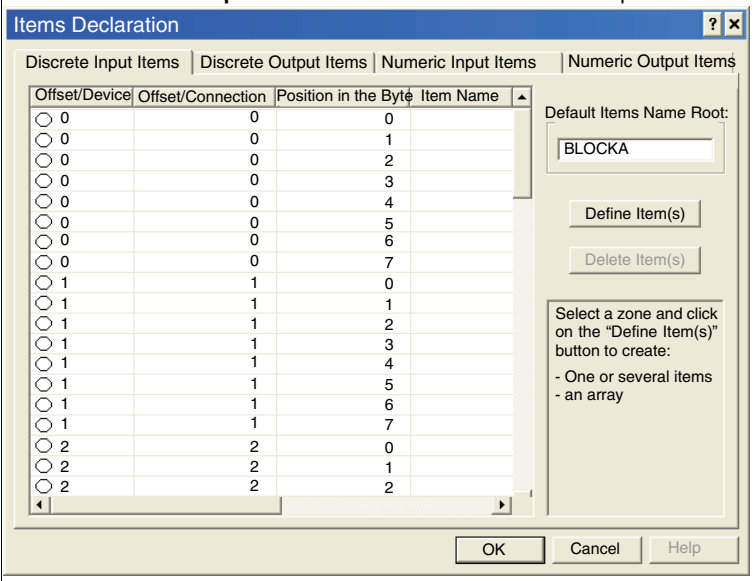
For this example, you need to create input items for the 19 input bytes and output items for the 6 output bytes using the Unity Pro EtherNet/IP configuration tool. These input and output items include:

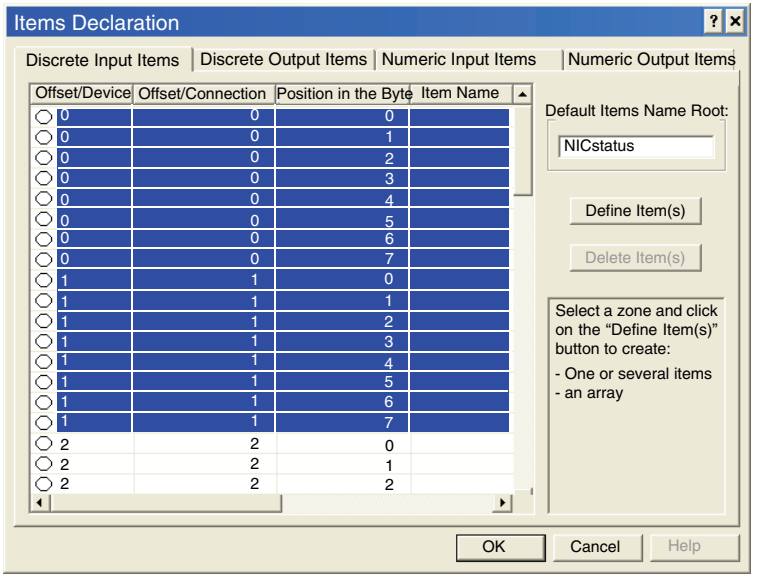
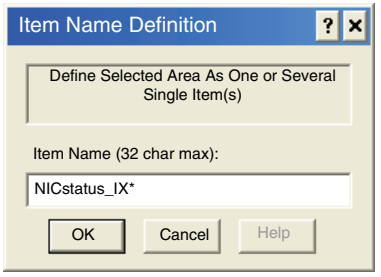
- discrete input and output items—made up of 1 or more bits—for the digital I/O modules, and
- numeric input and output items—made up of either an 8-bit byte or a 16-bit word—for the analog I/O modules

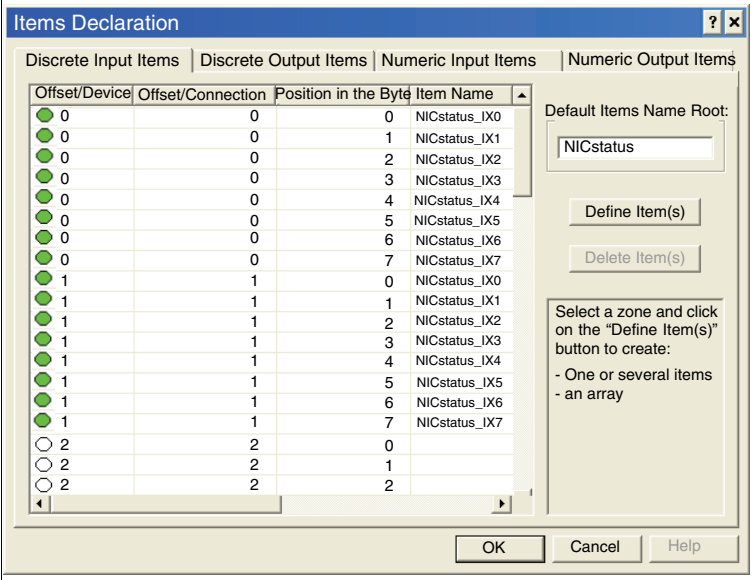
The following examples show you how to create each kind of item.

Creating Discrete Input Items

To create discrete input items for the STB NIC 2212 example, beginning with 16 discrete inputs for NIC status:

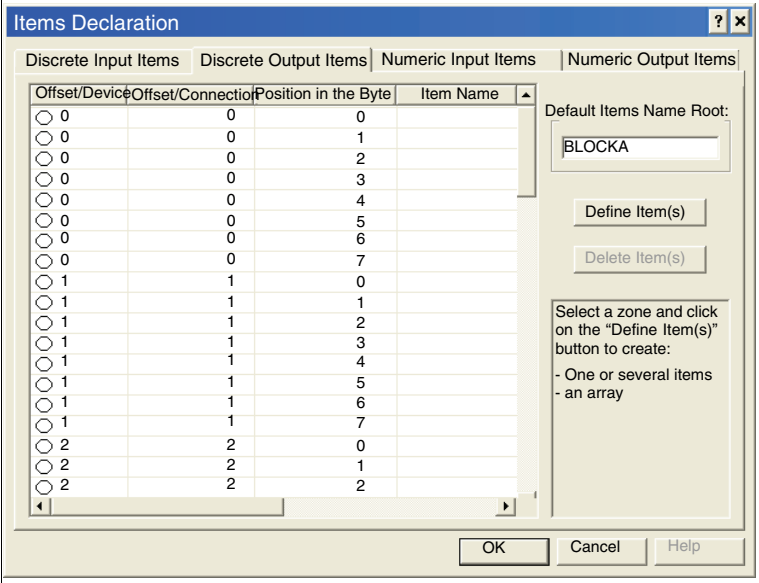
Step	Action
1	In the **Devices** window of the Unity Pro EtherNet/IP configuration tool, navigate to and select the connection item at Position 000, as shown below: 
2	Select **Devices** → **Properties**. The **Items Declaration** window opens: 
3	In the **Default Items Named Root** input box type: **NICstatus**.

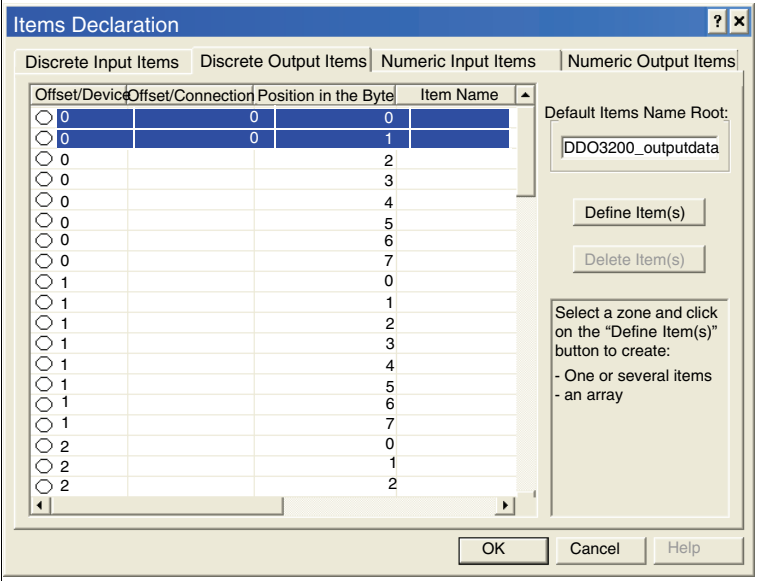
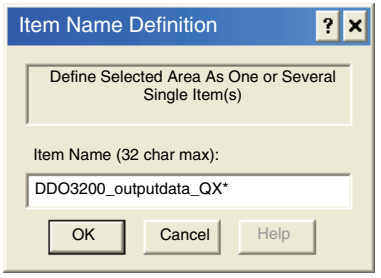
Step	Action
4	In the **Items List**, select the rows that correspond to bits 0-7 in bytes 0 and 1—i.e., the first 16 rows: 
5	Click the **Define Item(s)** button.The **Item Name Definition** dialog opens:  **Note:** The asterisk (*) indicates a series of discrete items with the same root name will be created.

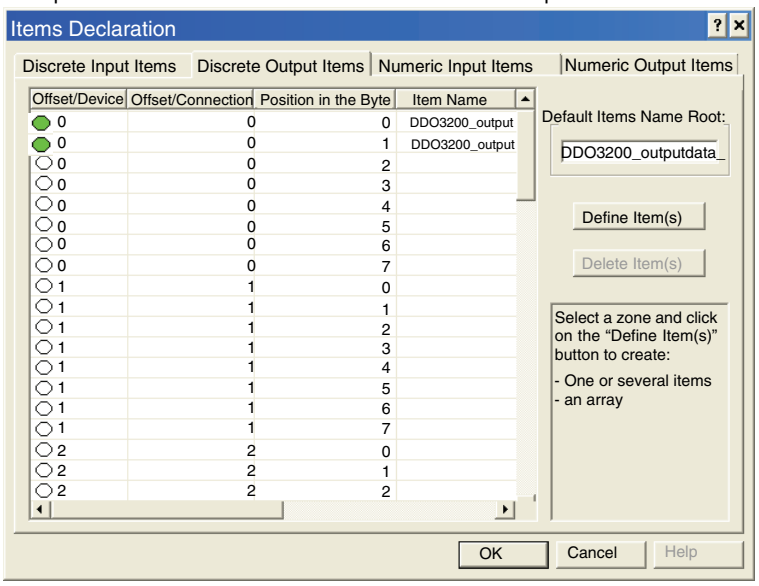
Step	Action
6	Accept the default **Item Name** and click **OK**. 16 discrete input items are created: 
7	Repeat steps 3 - 6 for each group of discrete input items you need to create. In this example, that includes items for each of the following groups: ● Byte: 2, Bits: 0-1, Default Items Name Root: DDI3230_inputdata ● Byte: 2, Bits: 2-3, Default Items Name Root: DDI3230_inputstatus ● Byte: 2, Bits: 4-5, Default Items Name Root: DDO3200_outputdataecho ● Byte: 2, Bits: 6-7, Default Items Name Root: DDO3200_outputstatus ● Byte: 3, Bits: 0-3, Default Items Name Root: DDI3420_inputdata ● Byte: 3, Bits: 4-7, Default Items Name Root: DDI3420_inputstatus ● Byte: 4, Bits: 0-3, Default Items Name Root: DDO3410_outputdataecho ● Byte: 4, Bits: 4-7, Default Items Name Root: DDO3410_outputstatus ● Byte: 5, Bits: 0-5, Default Items Name Root: DDI3610_inputdata ● Byte: 6, Bits: 0-5, Default Items Name Root: DDI3610_inputstatus ● Byte: 7, Bits: 0-5, Default Items Name Root: DDO3600_outputdataecho ● Byte: 8, Bits: 0-5, Default Items Name Root: DDO3600_outputstatus
8	Click on the **Discrete Output Items** tab to open that page.

Creating Discrete Output Items

To create discrete output items for the STB NIC 2212 example, beginning with 2 discrete outputs for the STB DDO3200 module:

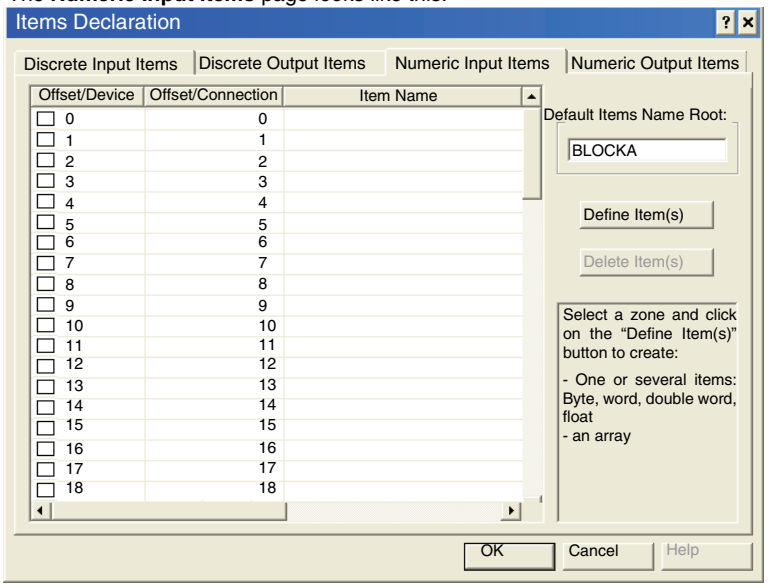
Step	Action
1	The **Discrete Output Items** page looks like this:  **Note:** Again, both the **Offset/Device** and **Offset/Connection** columns represent the byte address of the discrete output, while the **Position in the Byte** column indicates the bit position of the discrete output item.
2	In the **Default Items Name Root** input box type: **DDO3200_outputdata**.

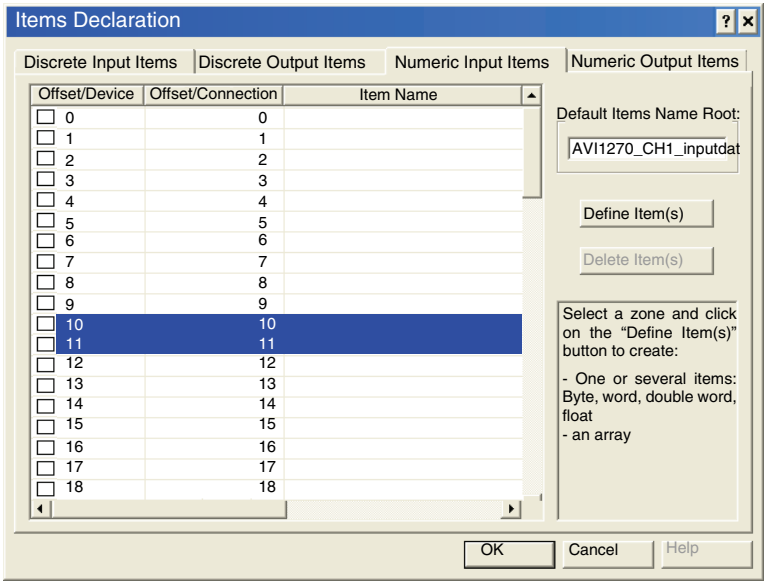
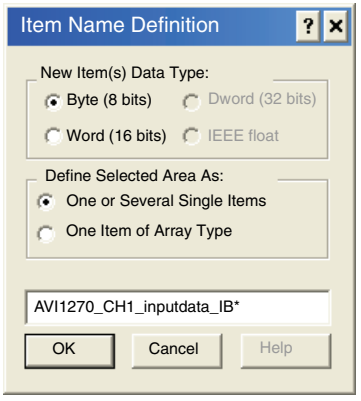
Step	Action
3	In the **Items List**, select the rows that correspond to bits 0-1 in byte 0—i.e., the first 2 rows: 
4	Click the **Define Item(s)** button.The **Item Name Definition** opens:  **Note:** The asterisk (*) indicates a series of discrete items with the same root name will be created.

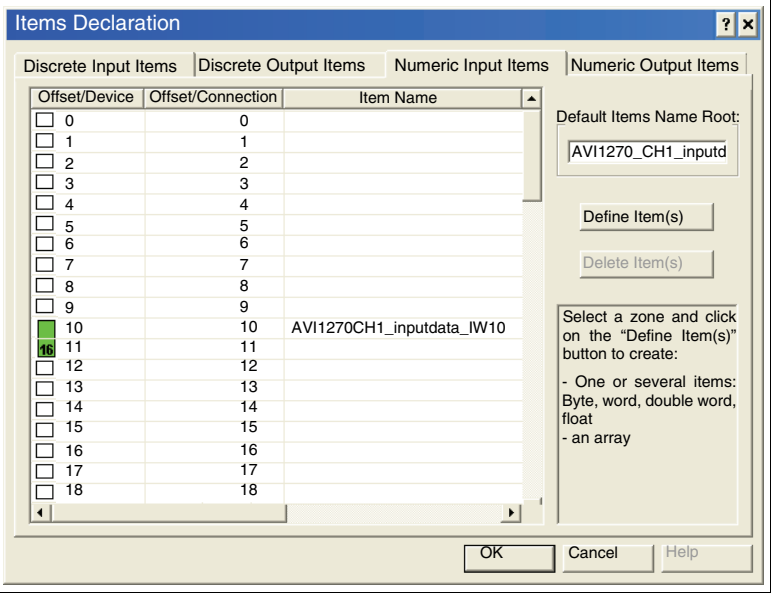
Step	Action
5	Accept the default **Item Name** and click **OK**. 2 discrete input items are created: 
6	Repeat steps 2 - 5 for each group of discrete output items you need to create. In this example, that includes items for each of the following groups: ● Byte: 0, Bits: 2-5, Default Items Name Root: DDO3410_outputdata ● Byte: 1, Bits: 0-5, Default Items Name Root: DDO3600_outputdata
7	Click on the **Numeric Input Items** tab to open that page.

Creating Numeric input Items

To create numeric input items for the STB NIC 2212 example, beginning with a channel 1 input data word for the STB AVI 1270 module:

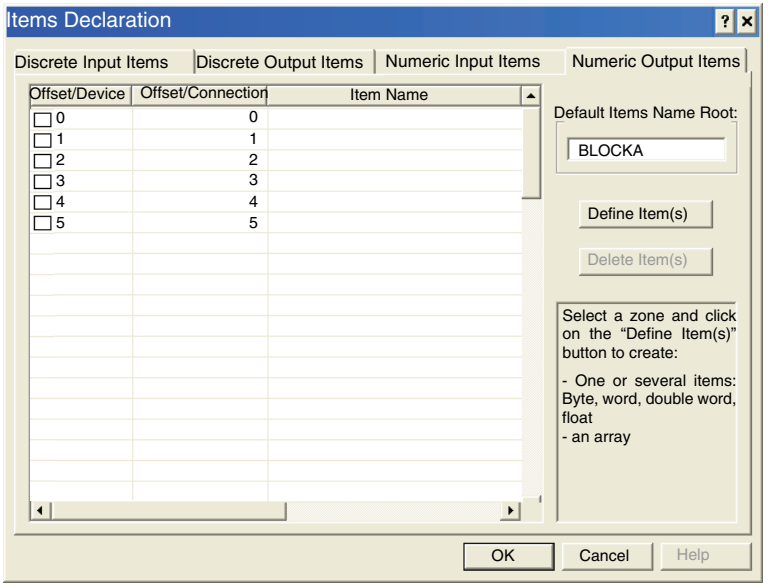
Step	Action
1	The **Numeric Input Items** page looks like this:  **Note:** In this example, both the **Offset/Device** and **Offset/Connection** columns represent the byte address. All items you create will be either an 8-bit byte or a 16-bit word.
2	In the **Default Items Name Root** input box type: **AVI1270_CH1_inputdata**.

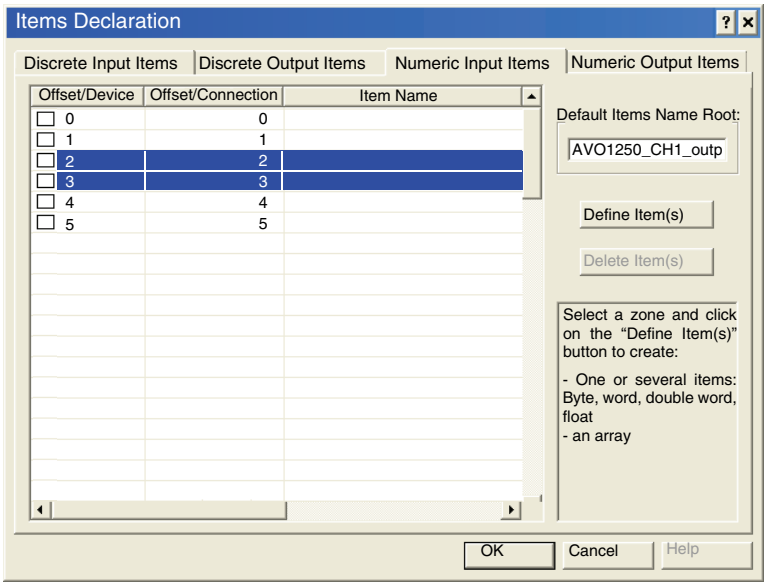
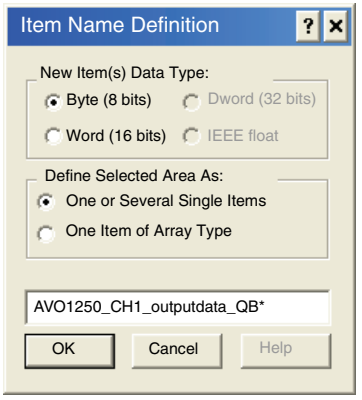
Step	Action
3	In the **Items List**, select bytes (or rows) 10 and 11: 
4	Click the **Define Item(s)** button.The **Item Name Definition** dialog opens: 

Step	Action
5	Select **Word (16 bits)** as the **New Item(s) Data Type**, then click **OK**. A new item is created: 
6	Repeat steps 2 - 5 for each group of numeric input item you need to create. In this example, that includes items for each of the following groups: ● Byte: 12, Default Items Name Root: AVI1270_CH1_inputstatus ● Word: 14-15, Default Items Name Root: AVI1270_CH2_inputdata ● Byte: 16, Default Items Name Root: AVI1270_CH2_inputstatus ● Byte: 17, Default Items Name Root: AVO1250_CH1_outputstatus ● Byte: 18, Default Items Name Root: AVO1250_CH2_outputstatus
7	Click on the **Numeric Output Items** tab to open that page.

Creating Numeric Output Items

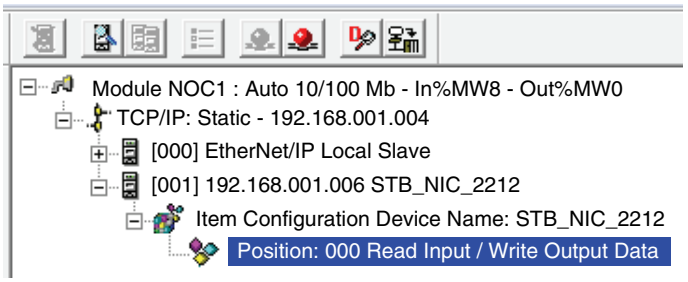
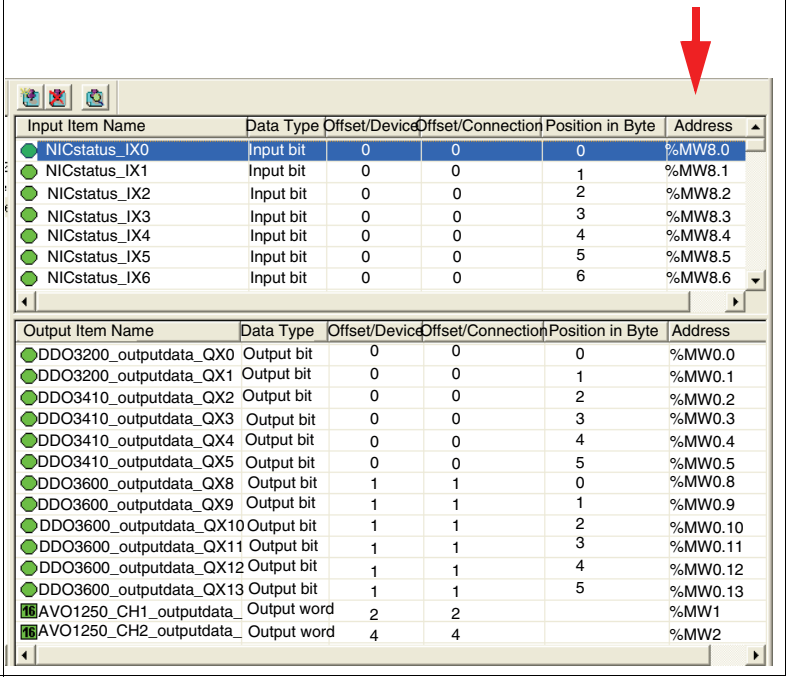
To create numeric output items for the STB NIC 2212, example, beginning with a output data word for the STB AVO 1250 module:

Step	Action
1	The **Numeric Output Items** opens:  **Note:** In this example, both the **Offset/Device** and **Offset/Connection** columns represent the byte address. All items you create will be 16-bit words comprising 2 bytes.
2	In the **Default Items name Root** input box type: **AVO1250_CH1_outputdata**.

Step	Action
3	In the **Items List**, select bytes (or rows) 10 and 11: 
4	Click the **Define Item(s)** button.The Item **Name Definition** dialog opens: 
5	Repeat steps 2 - 5 for the AVO 1250 channel 2 output data at bytes 4 and 5.
6	Click **OK** to close the **Items Declaration** window.
7	Select **File** → **Save** to save your edits.

Viewing Input and Output Item Addresses

The final step in this example is to view the address the Unity Pro EtherNet/IP configuration tool assigns to each newly created input and output item. To do this:

Step	Action
1	In the **Devices** window of the Unity Pro EtherNet/IP configuration tool, the item at Position 000 should still be selected. If not, select it, as shown below:  The automatically configured input and output items appear on the right side of the screen in the I/O area (as depicted in the next step, below).
2	If necessary, use the horizontal scroll bar to scroll to the far right of the input or output area and display the **Address** column, which identifies the location of the input or output in the 140 NOC 771 00: 

Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
NICstatus_IX0	Input bit	0	0	0	%MW8.0
NICstatus_IX1	Input bit	0	0	1	%MW8.1
NICstatus_IX2	Input bit	0	0	2	%MW8.2
NICstatus_IX3	Input bit	0	0	3	%MW8.3
NICstatus_IX4	Input bit	0	0	4	%MW8.4
NICstatus_IX5	Input bit	0	0	5	%MW8.5
NICstatus_IX6	Input bit	0	0	6	%MW8.6

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
DDO3200_outputdata_QX0	Output bit	0	0	0	%MW0.0
DDO3200_outputdata_QX1	Output bit	0	0	1	%MW0.1
DDO3410_outputdata_QX2	Output bit	0	0	2	%MW0.2
DDO3410_outputdata_QX3	Output bit	0	0	3	%MW0.3
DDO3410_outputdata_QX4	Output bit	0	0	4	%MW0.4
DDO3410_outputdata_QX5	Output bit	0	0	5	%MW0.5
DDO3600_outputdata_QX8	Output bit	1	1	0	%MW0.8
DDO3600_outputdata_QX9	Output bit	1	1	1	%MW0.9
DDO3600_outputdata_QX10	Output bit	1	1	2	%MW0.10
DDO3600_outputdata_QX11	Output bit	1	1	3	%MW0.11
DDO3600_outputdata_QX12	Output bit	1	1	4	%MW0.12
DDO3600_outputdata_QX13	Output bit	1	1	5	%MW0.13
AVO1250_CH1_outputdata_	Output word	2	2		%MW1
AVO1250_CH2_outputdata_	Output word	4	4		%MW2

3.4 Connecting to Third Party Devices

At a Glance

Overview

The EtherNet/IP communication module can connect to and communicate with EtherNet/IP devices made by third party manufacturers. This section describes how to set up communications with the Rockwell Automation 1734-AENT remote device and its I/O.

What's in this Section?

This section contains the following topics:

Topic	Page
Adding a Third Party Device to the Sample Network	113
Add an EDS File	114
Automatically Detect and Add the 1734-AENT PointIO Adapter	116
Configuring 1734-AENT PointIO Adapter Properties	117
Viewing 1734-AENT PointIO Adapter I/O Addresses	121

Adding a Third Party Device to the Sample Network

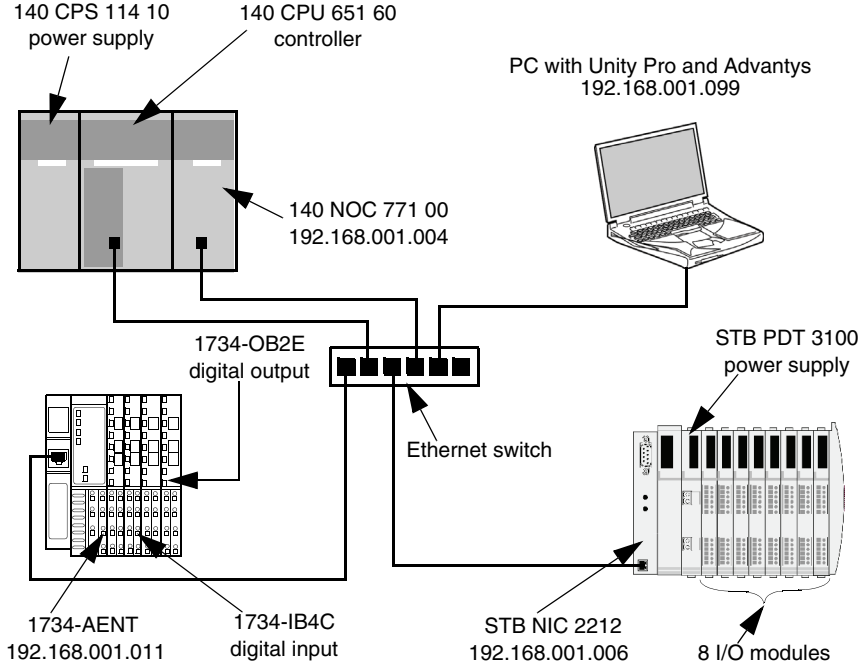
Overview

The next task is to extend the sample network by adding the following third party devices:

- 1734-AENT PointIO adapter with IP address of 192.168.001.011
- 1734-IB4/C 4pt DC input module
- 1734-OB2E 2pt DC output module

Network Topology

The modified network topology looks like this:



To re-create this example, be sure to:

- use the IP addresses for your own configuration's:
  - PC
  - 140 NOC 771 00 EtherNet/IP communication module
  - STB NIC 2212 EtherNet/IP network interface module
  - 1734-AENT PointIO adapter
- check all wiring

Note: Unity Pro software running in the PC is used to configure the CPU 651 60 controller. In this example, the PC is indirectly wired to the CPU's Ethernet port via the Ethernet switch. Alternatively, you could bypass the switch and directly wire the PC to either the CPU's Modbus or USB ports.

Add an EDS File

Overview

Before you can add a third party device to your configuration, be sure the EDS file for that device is included in the Unity Pro EtherNet/IP configuration tool's **Device Library**.

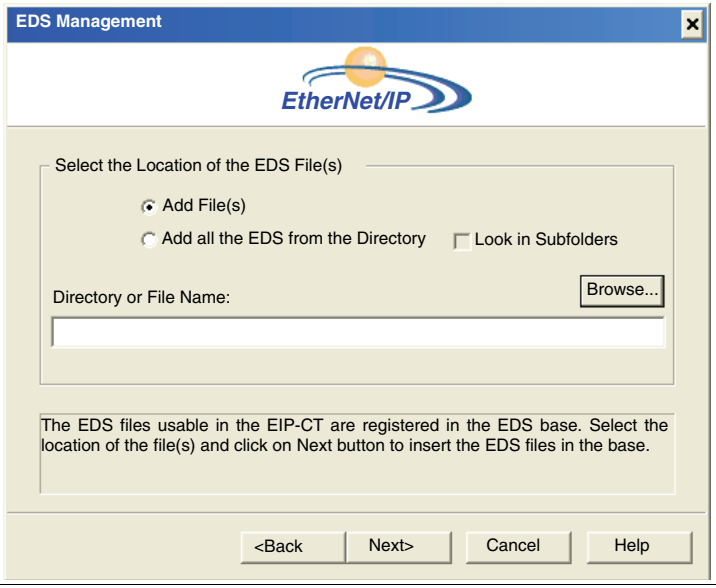
Use the EDS management wizard to add one or more EDS files to the **Device Library**. The wizard presents a series of instruction screens that:

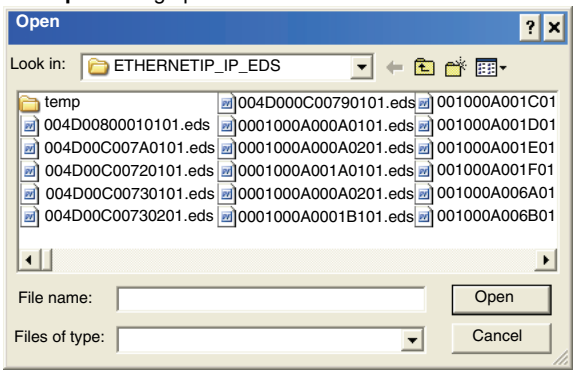
- simplify the process of adding EDS files to the **Device Library,** and
- provide a redundancy check that insures the same version of an EDS file cannot be added more than once

Note: Click **Devices** → **Options...** to open the **Display Options** window, where you can turn on or off the display of messages indicating the EDS file you are adding is a duplicate, or a different version of an EDS file already included in the **Device Library**.

Adding EDS Files

To add one or more EDS files to the **Device Library**:

Step	Action
1	Do one of the following: ● in the **Device Library**, click the Add button , or ● in the **Library** menu, click **Add** Page 1 of the wizard opens.
2	Click **Next**. Page 2 of the wizard opens: 

Step	Action
3	In the **Select the Location of the EDS File(s)** section, select either: ● **Add File(s)**, to add one or more EDS files you will individually select, or ● **Add all the EDS Files from the Directory**, to add all files from a folder you will select. ● Select **Look in Subfolders** to also add EDS files in subfolders beneath the folder you select
4	Click the **Browse** button. The **Open** dialog opens: 
5	Use the **Open** dialog to navigate to and select: ● one or more EDS files, or ● a folder containing EDS files
6	Click **Open**. The dialog closes and your selection appears in the **Directory or File Name** field.
7	Click **Next**. The wizard compares the selected EDS files against existing files in the **Device Library**.
8	(Conditional) If one or more selected EDS files are duplicates and if notice of redundant files is enabled in the **Display Options** dialog, the configuration tool displays a **File Already Exists** message. **Close** the message.
9	Page 3 of the wizard opens indicating the Status of each device you selected ● a green check mark indicates the EDS file can be added ● a blue informational icon indicates a redundant file (Optional) Select a file in the list, then click **View Selected File** to open it.
10	Click **Next** to add the non-duplicate files. Page 4 of the wizard opens, indicating the action is complete.
11	Click **Finish** to close the wizard. The device(s) you added can now be inserted into your EtherNet/IP configuration.

Automatically Detect and Add the 1734-AENT PointIO Adapter

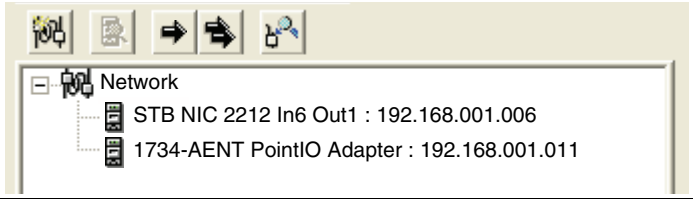
Overview

Use the Unity Pro EtherNet/IP configuration tool to automatically detect the 1734-AENT PointIO adapter. After it is detected, you can add it to your project.

Note: The 1734-AENT must be active online with a valid IP address before you can detect and add it to your project.

Detecting and Adding Network Devices

To automatically detect the 1734-AENT, then add it to your project, follow these steps:

Step	Action
1	Launch the configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.
2	In the configuration tool, begin on-line operations by clicking the **Go Online** button .
3	Click on the **Configuration** tab to enable automatic network detection:
4	Click the **Read Network Configuration** toolbar button . The configuration tool searches the network for EtherNet/IP devices, classifies them using the device EDS file, then lists the EtherNet/IP devices it detects. 
5	Select the **1734-AENT PointIO Adapter** in **Network Detection** window.
6	Click the Insert in Configuration button . The **Properties** window opens, where you can configure the 1734-AENT PointIO adapter.

Configuring 1734-AENT PointIO Adapter Properties

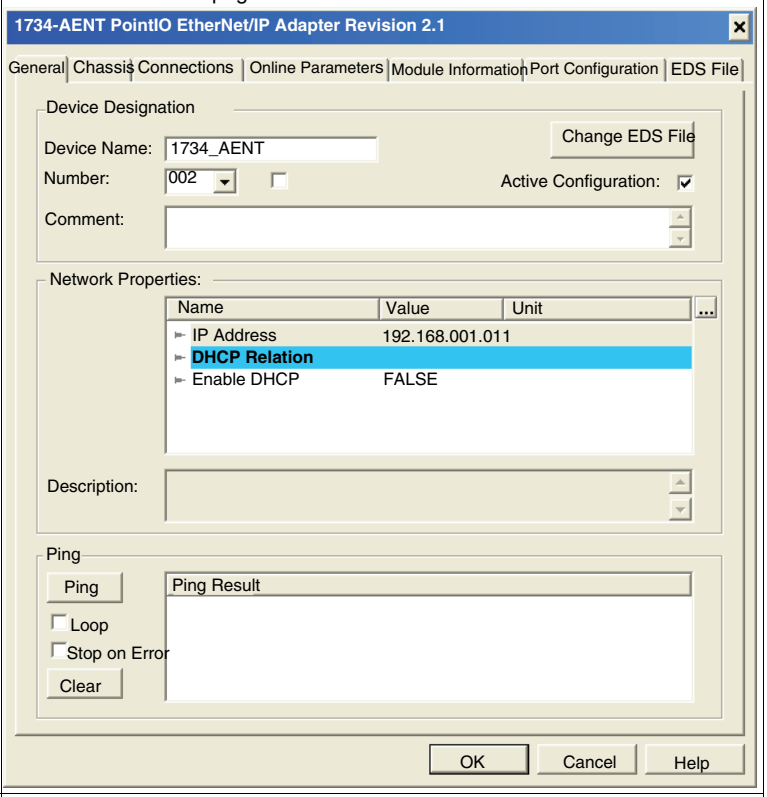
Overview

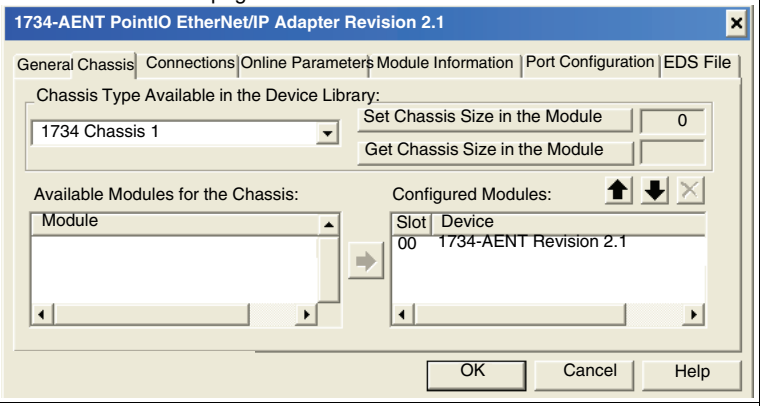
The 1734-AENT PointIO adapter module properties window presents the following tabbed pages. Only some of these pages need to be edited for this example:

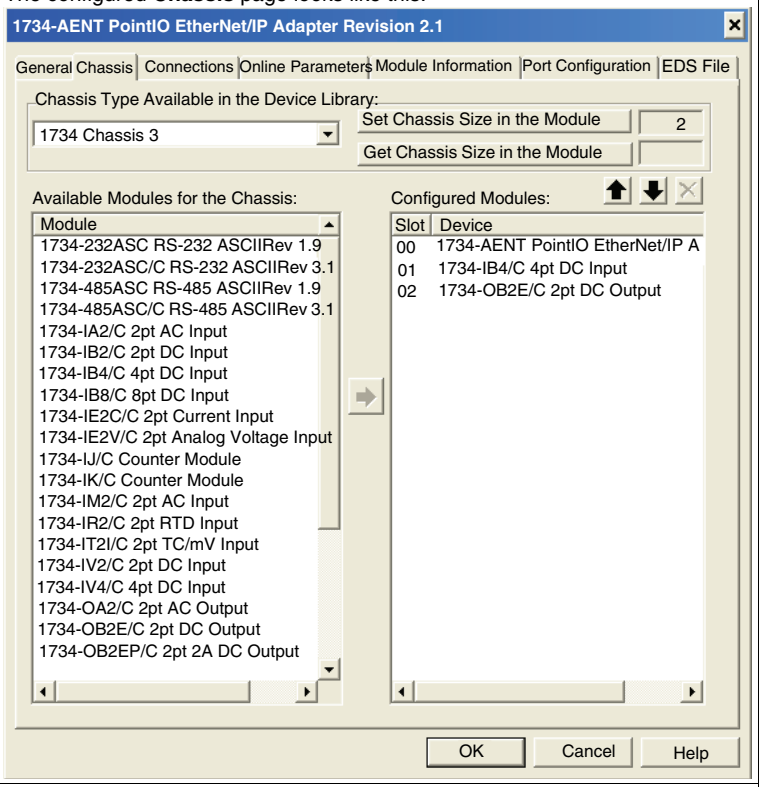
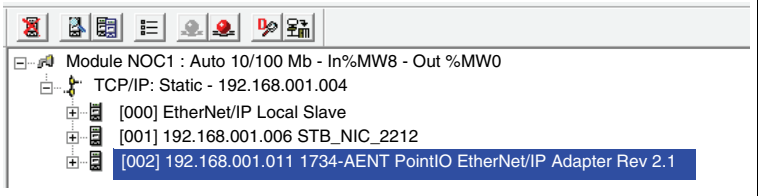
In this page...	Do the following...
General	• input device name • configure IP address • add the device to the project configuration
Chassis	Add 2 I/O modules to the chassis: • 1734-IB4/C 4pt DC input module • 1734-OB2E 2pt DC output module
Connections	Accept the default settings.
Online Parameters	Accept the default settings, if any.
Module Informations	(Read-only page - no configuration required)
Port Configuration	(Read-only page - no configuration required)
EDS File	(Read-only page - no configuration required)

Configuring the 1734-AENT

The following settings are used in the sample configuration:

Step	Action	
1	Click on the **General** page: 	
2	In the **General** page, edit the following settings:	
	Device Name	**1734_AENT**
	Number	The sequence of the device in the **Devices** window. for this example, type in **003**.
	Active Configuration	Be sure this checkbox is selected.
	IP Address	**192.168.001.011**

Step	Action	
3	Click on the **Chassis** page:  	
4	In the **Chassis** page, do the following:	
	a	Select **1734 Chassis 3** in the **Chassis Type Available in the Device Library** list. The **Available Modules for the Chassis** list is populated and two [Empty] rows appear in the **Configured Modules** list.
	b	Select **1734-IB4/C 4pt DC Input** in the **Available Modules for the Chassis** list.
	c	Click the Insert button ➡. The module appears in position 01 in the **Configured Modules** list.
	d	Select **1734-OB2E/C 2pt DC Output** in the **Available Modules for the Chassis** list.
	e	Click the Insert button ➡. The module appears in position 02 in the **Configured Modules** list.

Step	Action
5	The configured **Chassis** page looks like this: 
6	Click **OK** to save your settings and close the properties window. A node is added to the project configuration in the **Devices** window:  The next step is to view the device's inputs and outputs.

Viewing 1734-AENT PointIO Adapter I/O Addresses

Overview

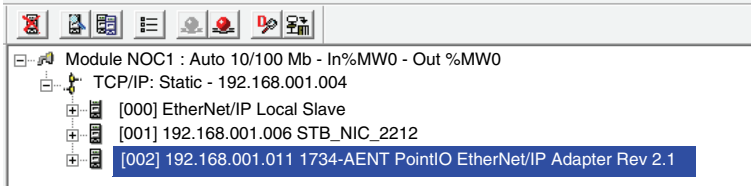
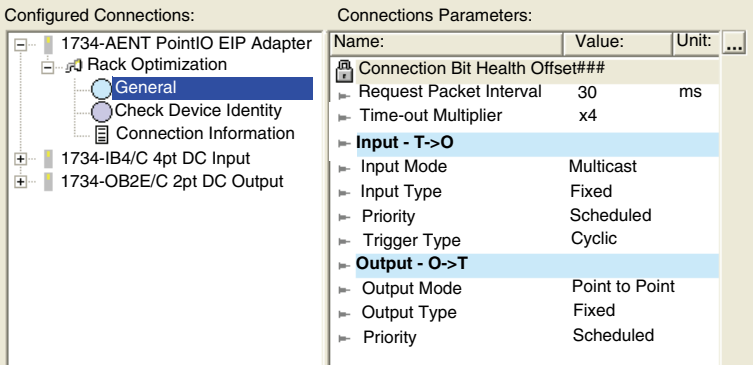
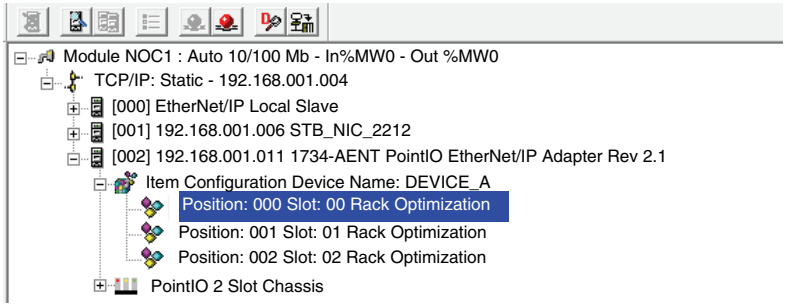
Because the Device Library includes EDS files for the 1734-AENT PointIO adapter and its discrete input and output modules, the Unity Pro EtherNet/IP configuration tool automatically:

- creates a rack optimized CIP connection from the 140 NOC 771 00 EtherNet/IP communication module to the 1734-AENT PointIO adapter, and
- configures each input and output item by assigning:
  - an item name
  - an address location
  - a size allotment based on its data type

Note: In this example, the configuration tool created a rack optimized connection, which is more efficient. A rack optimized connection can be used only with discrete (digital) I/O modules. For analog I/O modules, each analog module must be connected to the 140 NOC 771 00 using a separate connection.

Viewing the CIP Connection and I/O

To view the automatically created CIP connection and the I/O items in the Unity Pro EtherNet/IP configuration tool:

Step	Action
1	In the Devices window, select the 1734-AENT: 
2	In the Devices menu, select **Properties**. The **Properties** window opens for the 1734-AENT.
3	In the **Connections** tab, under the top-most Rack Optimization connection, select **General**. The rack optimized connection properties are displayed in the **Connection Parameters** list: 
4	In the **Devices** window, navigate to and select the first Rack Optimized connection item at Position 000, as shown below:  The automatically configured input and output items appear on the right side of the screen in the I/O area (as depicted in the next step, below).

Step	Action
5	If necessary, use the horizontal scroll bar to scroll to the far right of the input or output area and display the **Address** column, which identifies the location of the input or output in the 140 NOC 771 00:

Input Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
Slot00_Target_Error_Co	Input dword	0	0		%MW18
Slot00_Target_Error_Co	Input dword	4	4		%MW20
Slot01_Input_Value_0	Input bit	9	9	0	%MW22.8
Slot01_Input_Value_1	Input bit	9	9	1	%MW22.9
Slot01_Input_Value_2	Input bit	9	9	2	%MW22.10
Slot01_Input_Value_3	Input bit	9	9	3	%MW22.11
Slot02_Status_Output_0	Input bit	10	10	0	%MW23.0
Slot02_Status_Output_1	Input bit	10	10	1	%MW23.1

Output Item Name	Data Type	Offset/Device	Offset/Connection	Position in Byte	Address
Slot00_Reserved	Array of 2 output words	0	0		%MW10
Slot01_Input_Value_	Input bit	9	9	0	%MW15.0
Slot01_Input_Value_	Input bit	9	9		%MW15.1

Optimizing Performance

4

At a Glance

Overview

This chapter describes how to optimize performance of your EtherNet/IP network.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Selecting a Switch	126
4.2	Understanding EtherNet/IP Connections	130
4.3	Estimating EtherNet/IP System Performance	135

4.1 Selecting a Switch

At a Glance

Overview

This section describes how to select an Ethernet switch for your EtherNet/IP network.

What's in this Section?

This section contains the following topics:

Topic	Page
Role of a Switch in an EtherNet/IP Network	127
Full Duplex	127
IGMP Snooping	128
Port Mirroring	128
Virtual Local Area Network (VLAN)	129
Simple Network Management Protocol (SNMP) Agent	129

Role of a Switch in an EtherNet/IP Network

Overview

EtherNet/IP requires the use of managed switches, not hubs. Unlike a hub, a managed switch:

- inspects data packets it receives to determine both its source and destination network address
- delivers the data packet only to the intended destination switch, port and device

By sending data packets only to the intended recipient device, the switch conserves network bandwidth and provides better performance than a hub.

Recommended Switch Features

When acquiring an Ethernet switch for your EtherNet/IP network, be sure the switch offers the following features:

- full duplex support
- IGMP snooping
- port mirroring
- VLAN support
- an embedded SNMP agent

Full Duplex

EtherNet/IP supports full duplex transmissions, where each end device can simultaneously transmit at the maximum transmission speed.

Because each end device is the only device on its segment, there is no risk of collisions of data packets sent by competing devices. Ethernet's Carrier Sense Multiple Access with Collision Detect (CSMA/CD) arbitration protocol, with its inherent random delays, is never invoked.

IGMP Snooping

Overview

The switch must support v2 (or higher) of the Internet Group Management Protocol (IGMP). Switches that support this protocol can listen in on IGMP conversations between remote routers and Ethernet devices connected to the switch.

IGMP snooping is an essential feature of multicast messaging. When a switch detects a transmission sent by an Ethernet device to:

- join a multicast group, the switch records the association between the device's port number and the multicast group destination address
- leave a multicast group, the switch removes the entry associating the Ethernet device with the multicast group destination address

Multicast Messaging

Multicast messaging is the transmission of information by a device to a group of devices. By sending a transmission to a group address, the transmission is sent only once. Routers receive the transmission, and then forward it via optimal network paths to each device that is a member of the multicast group.

Multicast messaging reduces network traffic by:

- requiring that a message be sent only once
- sending the message only to devices for which the message is intended

Port Mirroring

Port mirroring lets you troubleshoot switch port transmissions, by re-directing a copy of all data packets sent through one switch port to a different port.

Without port mirroring, it is impossible to troubleshoot a single switch port's transmissions.

For example, if you send to port 2—the mirroring port—a copy of data packets sent to port 1, you can attach a packet sniffer to port 2 and thereby troubleshoot port 1.

Note: A port used for port mirroring should be dedicated to only port mirroring, and should not be used for any other purpose. Only the packet sniffer should be connected to that port.

A packet sniffer's troubleshooting features should include:

- analyzing network problems
- monitoring network activity

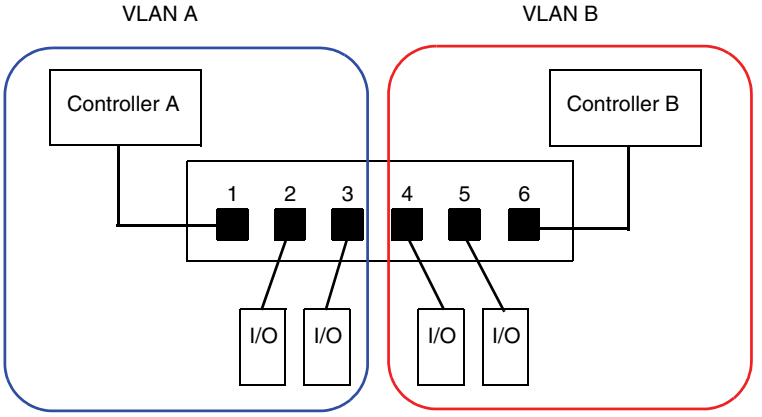
Virtual Local Area Network (VLAN)

A VLAN is a group of Ethernet devices—which may be physically located on different network segments—that are grouped together and communicate as if they were located on the same LAN segment.

In a port-based—or static—VLAN, an administrator confers VLAN membership by assigning individual switch ports to the VLAN. Any device connected to that port is effectively added to the VLAN.

VLANs permit the creation of logically separate groups of network devices, without having to physically re-wire those devices. When a switch receives a message directed to a specific VLAN, it forwards that message only to the switch ports that are members of that VLAN. The switch does not send the message to its ports that are not members of that VLAN.

In the example, below, switch ports 1, 2, and 3 are assigned to VLAN A, while switch ports 4, 5, and 6 are assigned to VLAN B:



Simple Network Management Protocol (SNMP) Agent

An *SNMP agent* is a software component that reports management data about the switch to another device acting as an *SNMP manager*. If a switch is equipped with SNMP agent software, the SNMP manager can:

- retrieve management data about the switch
- manage the switch by editing its configuration settings
- accept traps—or notices of events—affecting the state of the switch

4.2 Understanding EtherNet/IP Connections

At a Glance

Overview

This section describes EtherNet/IP network connections, networks and data packets.

What's in this Section?

This section contains the following topics:

Topic	Page
EtherNet/IP Messaging Overview	131
TCP Connections	133
CIP Connections and Messages	134
Messaging Performance	134

EtherNet/IP Messaging Overview

Overview

EtherNet/IP supports both implicit and explicit messaging:

Message Type	Data Transmitted	Includes...
Explicit	Information	● Non-time critical management data ● Read/write application data
Implicit	I/O data	● Real-time I/O data ● Real-time control data from remote devices

EtherNet/IP supports implicit and explicit messaging:

Explicit Messaging

Explicit messages transmit point-to-point, client-server data. The client initiates the transaction by sending a message containing both address and service request information. Explicit messaging uses CIP transport class 3 connections.

Explicit messages can be sent as either connected or unconnected, depending upon the frequency of your need for data and the level of service required:

connected explicit message	● Begins when an originating device initiates a connection by sending a request to a target device. ● The connection is established when the originator receives a successful response from the target. ● A CIP connected message has a high priority and better guarantee of service, but requires a greater amount of resources from both the target and originator devices.
unconnected explicit message	● Less resource intensive.

The explicit message can be triggered by the application.

Two timeout periods exist for connected explicit messages, one for the:

- length of time the CIP explicit message connection remains open (connection timeout)
- length of time the requesting device waits for a response (response timeout)

Each timeout is configurable.

Connection Timeout Calculation:

The connection timeout is calculated by multiplying the **EM Connected RPI** parameter (located in the **EtherNet/IP** page of the **Channel Properties** window) against a non-configurable, fixed multiplier value, as follows:

(EM)Connected RPI x (fixed timeout multiplier) = EM connection timeout

Using the **EM Connected RPI** default value of 10000 ms and the globally fixed multiplier of 4, this calculation yields an explicit message connection timeout of:

(10000 ms) x (4) = 40 s

Request Timeout:

The response timeout can be configured using the **EM Request Timeout** parameter (also located in the **EtherNet/IP** page of the **Channel Properties** window).

Implicit Messaging

Implicit messaging is producer/consumer messaging. The originator defines the connection parameters, such as the required data, and how often that data must be produced. After the connection is established, both the originator and the target can act as a producer or consumer. Implicit messaging uses CIP transport class 1 connections.

An implicit message is a connected CIP message. It can be cyclic, or triggered by either the application or a change of state.

The data contained in an implicit message can include, for example:

- real-time I/O data
- functional safety data
- motion control data

After the connection is established, both sides in the transmission must produce data at the rate established when the connection was opened.

Implicit connected messages are routed either point-to-point or multicast over a CIP connection using UDP/IP data packets. Each packet contains both data and a unique connection ID. Because UDP/IP data packets do not contain additional information for addressing, flow control and error recovery, the UDP/IP data packet is smaller and its transmission speed faster than TCP/IP data packets used in explicit connected messaging.

There are 3 types of connections used for implicit messaging:

Type	Description
Exclusive Owner	A connection to an output on the target device. Only one originator may control an target's outputs.
Input Only	A connection to an input on the target. No output data is sent.
Listen Only	An Input Only connection that is owned by another Exclusive Owner connection. It received data at the rate defined by the Exclusive Owner. This connection type behaves as an Input Only connection, except that if the Exclusive Owner connection times out, the Listen Only connection also closes.

TCP Connections

Overview

EtherNet/IP uses TCP connections as a pipeline for CIP connections Both connected and unconnected messaging use the TCP connection.

TCP Connection Limits

The 140 NOC 771 00 EtherNet/IP communication module can provide up to 67 TCP connections, as follows:

Connection type	Maximum number of connections
I/O adapter	64[1]
I/O scanner	
Explicit message client	
Explicit message server	3
Total TCP connections:	67

[1]64 connections can be used for any combination of:
- **I/O adapter connections**
- **I/O scanner connections**
- **explicit messages (as client)**

A single TCP connection can support multiple CIP connections.

Note: TCP connections dedicated to other services, for example FTP, are not included in the above numbers.

CIP Connections and Messages

Overview

EtherNet/IP uses CIP connections to transmit messages between objects running in connected devices. There are different types of CIP connections.

Connection Types

CIP connection types include:

CIP connection type...	Supports...
Rack optimized	The grouping of data from multiple, I/O modules in the same rack transmitted over a single connection. **Note:** A rack optimized connection: ● can transmit only device status and data ● applies only to digital I/O modules ● a CIP connection is consumed by each IO module, in addition to the rack optimized connextion
Direct	A link between a controller and a single device. **Note:** A connection to an analog I/O module must be via a direct connection.

Connection Limits

The 140 NOC 771 00 EtherNet/IP communication module can provide up to 198 concurrent CIP connections, as follows:

Connection type	Maximum number of connections
I/O adapter	128[1]
I/O scanner	
Explicit message client	64
Explicit message server	6
Total TCP connections:	198
[1]128 CIP connections can be used for any combination of: ● **I/O adapter connections** ● **I/O scanner connections**	

Note: Up to16 simultaneous explicit messaging connections can be active per scan.

Messaging Performance

Maximum Messaging Load

The 140 NOC 771 00 EtherNet/IP communication module supports an implicit messaging load of up to 7500 data packets per second (pps).

4.3 Estimating EtherNet/IP System Performance

At a Glance

Overview

This section shows you how to predict the impact of your EtherNet/IP network topology on system performance.

What's in this Section?

This section contains the following topics:

Topic	Page
Determining EtherNet/IP Network Performance	136
Network Load Calculation Example	138

Determining EtherNet/IP Network Performance

Overview

You can estimate the performance of a proposed EtherNet/IP network design using the following information:

- the total number of implicit CIP connections to be used by each EtherNet/IP device
- the frequency, in milliseconds, that an EtherNet/IP device sends or receives data packets over a connection
- the load, in data packets per second, each EtherNet/IP device can handle

Counting CIP Connections

CIP connections can be either explicit or implicit.

Explicit CIP connections are one-time transmissions of non-time-sensitive data, which exist for only a brief period of time.

Implicit CIP connections transmit time-sensitive data between an EtherNet/IP communication module and end devices—typically I/O modules.

Depending upon the features of the particular I/O adapter device used in your EtherNet/IP network, the device may support implicit connections that are either rack optimized or direct.

Because they remain open and continuously transmit data packets, implicit CIP connections comprise the most significant part of the network load. Be sure to identify and include every implicit connection when making your performance calculations.

Configuring Implicit Connections

An input implicit connection can be configured to cause the producing device to transmit messages either:

- cyclically, at a pre-configured frequency, or
- upon a change of state

An output implicit connection can only be configured to cause the originator to transmit messages to the target device cyclically.

Cyclical implicit connections are used to transmit I/O data and produce the greatest load on the network. Change of state implicit connection triggers usually reduce the number of transmissions, and thereby place a smaller load on the network.

The frequency for transmitting cyclical implicit messages is determined by the configurable requested packet interval (RPI (see *p. 210*)) setting for the producing device. Typical RPI values can range from:

- 2 to 20 ms for digital I/O devices
- 20 to 80 ms for analog I/O devices

In the following examples, all implicit connections are presumed to be cyclic.

Calculating the Load for an EtherNet/IP Device

The load on an EtherNet/IP network is the sum of the load contributed by each device. The load contributed by a single device is the sum of the load of all its implicit CIP connections.

The load calculation formula for a single device is:

Load = (number of packets per connection) x (number of connections) / RPI

The *number of packets per connection* value depends upon the capacity of the device and can be either:

- 2, for devices that support bi-directional communication, or
- 1, for devices that support uni-directional communication

In the following examples, all devices are presumed to support bi-directional communication, and the *number of packets per connection* value is always 2.

Example 1:

The load generated by an EtherNet/IP module acting as an I/O adapter with 5 implicit CIP connections—one to each of 5 analog I/O modules—each with an RPI settings of 50 ms is:

Load = 2 x 5 / 50 ms = 200 data packets per second

Example 2:

The load generated by an EtherNet/IP module—for example, a Rockwell 1734-AENT PointIO—acting as an I/O adapter with 1 rack optimized implicit CIP connection to a group of 6 digital I/O modules with an RPI setting of 10 ms is:

Load = 2 x 1 / 10 ms = 200 data packets per second

Predicting Network Performance

To predict the performance of your proposed EtherNet/IP network design, you must determine whether the I/O scanner device can handle the load contributed by every I/O adapter device. To do this, perform the following steps:

1. Calculate the implicit messaging load, in data packets per second, for each remote device.
2. Sum the load estimates for all remote devices.
3. Compare the total implicit messaging load against the maximum implicit messaging capacity—in data packets per second—of the EtherNet/IP device acting as I/O scanner.

If the projected total load for an EtherNet/IP module acting as an I/O scanner exceeds its implicit messaging load limits, consider one or more of the following corrective actions:

- If the I/O adapter supports rack optimized connections, and if a single rack of digital I/O uses multiple direct connections, replace the direct connections with a single rack optimized connection.
- Increase the RPI setting for a device where possible.
- Add another EtherNet/IP module to act as an I/O scanner, and re-design the EtherNet/IP topology in order to share the load.

Network Load Calculation Example

Overview

This example estimates EtherNet/IP system performance for a network composed of the following devices:

- a PLC that controls 3 remote I/O stations (A, B, and C)
- a 140 NOC 771 00 Quantum, or a TSX ETC 100 Premium EtherNet/IP communication module, acting as the local I/O scanner, installed in the PLC rack
- an 8-port Ethernet managed switch
- a PC running Unity Pro software, used to obtain diagnostic data via explicit messages
- 4 remote EtherNet/IP devices, acting as:
  - an I/O adapter (A) for a rack of I/O modules
  - a second I/O adapter (B) for a rack of I/O modules
  - a remote I/O drive (C)
  - a remote I/O scanner (D)

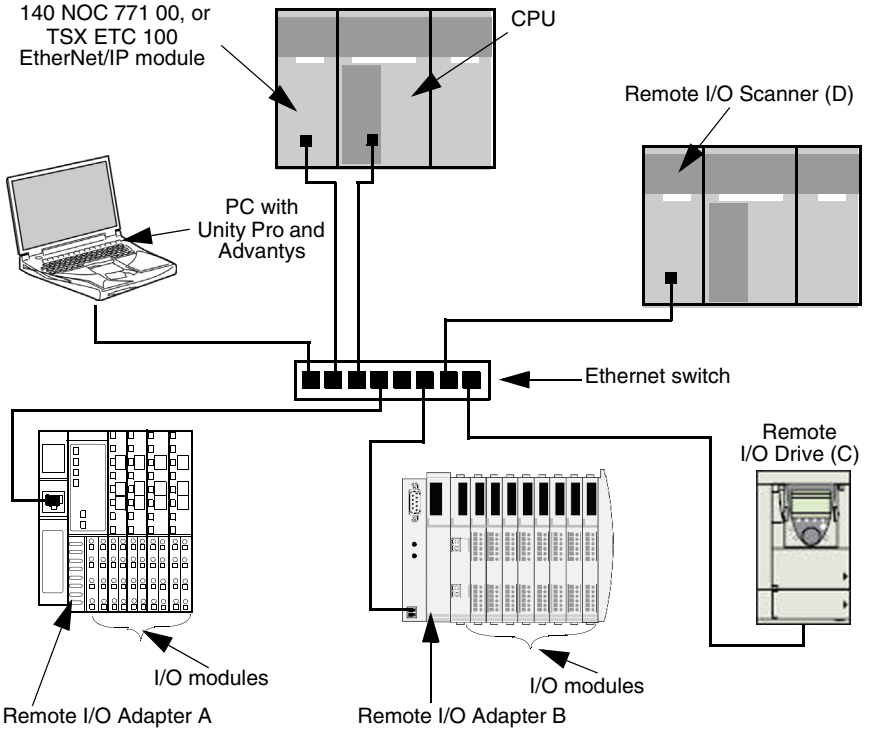
Unity Pro software running in the PC is used to configure the CPU 651 60 controller.

For programming purposes you need a connection to the PLC either through the CPU's Ethernet port or other supported programming paths.

To access the diagnostics data for the EtherNet/IP network your computer needs to be on the same subnet as the ETC module.

Network Diagram

The proposed network diagram looks like this:



EtherNet/IP Module Load Limits

When performing calculations, keep in mind that the total load of the EtherNet/IP module, acting as an I/O scanner, cannot exceed its implicit messaging limit. Those limits are:

EtherNet/IP module	Implicit messaging limit (data packets / second)
140 NOC 771 00	7500 pps
TSX ETC 100	7500 pps

Remote Device Connections and RPI

For the purpose of this example, it is assumed that the remote EtherNet/IP devices require the following numbers CIP connections, and are configured for the stated requested packet interval (RPI (see *p. 210*)) settings:

Device	CIP Connections	RPI Setting
remote I/O adapter (A)	5	20 ms
remote I/O adapter (B)	2	30 ms
remote drive (C)	2	30 ms
remote I/O scanner (D)	2	50 ms

For the purposes of this example, it is also assumed that all connections are bi-directional.

I/O Scanner Calculations

The EtherNet/IP module, acting as local I/O scanner, must be able to handle the implicit messaging load contributed by all of the remote devices. Your task is to:

1. estimate the implicit messaging load contributed by each remote device
2. sum the load values for each remote device
3. compare the total load against the maximum implicit messaging capacity—in packets per second—of the local I/O scanner

Recall that the implicit messaging load calculation formula for a single remote device is:

Load = (number of packets per connection) x (number of connections) / RPI

Because all connections are assumed to be bi-directional, the *number of packets per connection* value is always 2. Consequently, the estimated implicit messaging load contributed by each device, and the total implicit messaging load the local I/O scanner must handle can be estimated as follows:

Device	number of packets per connection	X	number of connections	÷	RPI	=	Total
A	2	X	5	÷	20 ms	=	500 pps
B	2	X	2	÷	30 ms	=	134 pps
C	2	X	2	÷	30 ms	=	134 pps
D	2	X	2	÷	50 ms	=	80 pps
					Total	=	848 pps

Conclusion

The projected total load for the module—848 pps—is within the EtherNet/IP device implicit messaging limit of 7500 data packets per second.

Note that the above calculations do not include any load contributed by explicit messaging, which is presumed to be negligible. When developing your control system application, be sure to keep any explicit messaging load within the module limits. These explicit messaging load limits are:

EtherNet/IP module	Explicit messaging limit (data packets / second)
140 NOC 771 00	100 pps
TSX ETC 100	100 pps

Explicit Messaging In Unity Pro

5

At a Glance

Overview

This chapter describes how to execute explicit messages in Unity Pro using either:

- the `MBP_MSTR` function block
- the **Online Action** window of the Unity Pro EtherNet/IP configuration tool

Unity Pro supports both connected and unconnected explicit messaging.

Note: Your Unity Pro application can contain more than 16 explicit messaging blocks, but only 16 explicit messaging blocks can be active at the same time. Also, there can be only one concurrent explicit message - connected or unconnected - from an EtherNet/IP communication module to the same remote EtherNet/IP device.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Explicit Messaging Services	142
Configuring Explicit Messaging Using MBP_MSTR	144
MBP_MSTR Example - Get_Attributes_Single	148
MBP_MSTR Example - Reset	154
Explicit Messaging Error Codes	160
Explicit Messaging - Online Action: Get_Attributes_Single	162
Explicit Messaging - Online Action: Reset	164

Explicit Messaging Services

Overview

Every explicit message performs a service. Each service is associated with a service code (or number). You will need to identify the explicit messaging service by its name, decimal number, or hexadecimal number.

You can execute explicit messages using either a Unity Pro function block or the Unity Pro EtherNet/IP configuration tool.

All services are available using the `MBP_MSTR` Unity Pro function block.

Note: All configuration edits made to an EtherNet/IP module via explicit messaging—including explicit messages executed by an MBP_MSTR block or by the Unity Pro EtherNet/IP configuration tool's Online Action window—are not included in the operating parameters the CPU sends the module on startup.

Services

You can use Unity Pro to construct a request that executes any service supported by the target device that is compliant with the EtherNet/IP protocol.

The services supported by Unity Pro include the following standard explicit messaging services:

Service Code		Description	Available in...	
Hex	Dec		Function Block	EIP config tool
1	1	Get_Attributes_All	X	X
2	2	Set_Attributes_All	X	X
3	3	Get_Attribute_List	X	—
4	4	Set_Attribute_List	X	—
5	5	Reset	X	X
6	6	Start	X	X
7	7	Stop	X	X
8	8	Create	X	X
9	9	Delete	X	X
A	10	Multiple_Service_Packet	X	—
D	13	Apply_Attributes	X	X
E	14	Get_Attribute_Single	X	X
10	16	Set_Attribute_Single	X	X
11	17	Find_Next_Object_Instance	X	X
14	20	Error Response (DeviceNet only)	—	—
15	21	Restore	X	X
16	22	Save	X	X
17	23	No Operation (NOP)	X	X
18	24	Get_Member	X	X
19	25	Set_Member	X	X
1A	26	Insert_Member	X	X
1B	27	Remove_Member	X	X
1C	28	GroupSync	X	—
"X" indicates the service is available. "—" indicates the service is not available.				

Configuring Explicit Messaging Using MBP_MSTR

Overview

Use the MBP_MSTR function block to configure EtherNet/IP connected and unconnected explicit messages. The MBP_MSTR block can send requests and receive responses up to 511 bytes long.

The operation begins when the input to the ENABLE pin is turned ON. The operation ends if the ABORT pin is turned ON, or if the ENABLE pin is turned OFF.

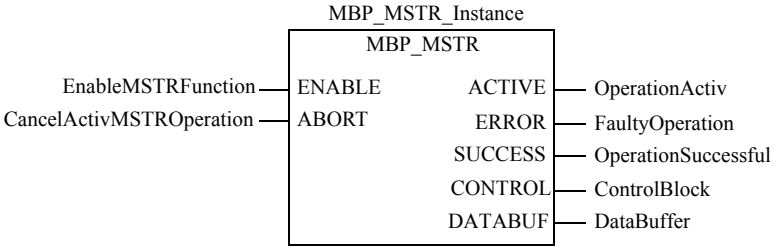
The CONTROL and DATABUF output parameters define the operation. Refer to Configuring the Control Block (see *p. 146*) and Configuring the Data Buffer (see *p. 147*), below, for details.

Note: The DATABUF parameter varies in size, depending upon its content. To avoid overwriting the request, the data buffer must be large enough to simultaneously contain both the request and response data.

The ACTIVE output turns ON during operation; the ERROR output turns ON if the operation aborts without success; the SUCCESS output turns ON upon the successful completion of the operation.

EN and ENO can be configured as additional parameters.

Representation in FBD



Input Parameters

Parameter	Data type	Description
ENABLE	BOOL	When ON, the explicit message operation (specified in the first element of the CONTROL pin) is executing.
ABORT	BOOL	When ON, the operation is aborted.

Output Parameters

Parameter	Data type	Description
ACTIVE	BOOL	ON when the operation is active. OFF at all other times..
ERROR	BOOL	ON when the operation is aborted without success. OFF before operation, during operation, and if operation succeeds.
SUCCESS	BOOL	ON when the operation concludes successfully. OFF before operation, during operation, and if operation fails.
CONTROL	WORD	This parameter contains the control block. See Configuring the Control Block, below, for a description of this parameter. **Note:** This parameter must be assigned to a located variable.
DATABUF	WORD	This parameter contains the data buffer. See Configuring the Data Buffer, below, for a description of this parameter. **Note:** This parameter must be assigned to a located variable.

Configuring the Control Block

The Control Block parameter consists of 9 contiguous words, as described below:

Register	Function	Description
CONTROL[0]	Operation	Low byte = 14 (CIP Explicit Message)
		High byte = ● 0: unconnected (16#000E) ● 1: connected (16#010E)
CONTROL[1]	Error status	Holds the error code (read-only).
CONTROL[2]	Data buffer length	Data buffer length, in words
CONTROL[3]	Response offset	Offset for the beginning of the response in the data buffer, in 16-bit words **Note:** To avoid overwriting the request, the response offset value must be greater than the request length CONTROL[6].
CONTROL[4]	Slot	Low byte = 0 (not used)
		High byte = slot location on backplane
CONTROL[5]	Device ID	The number assigned to the device in the Devices window of the Unity Pro EtherNet/IP configuration tool
CONTROL[6]	Request length	Length of the CIP request, in bytes
CONTROL[7]	Response length	Length of the response received, in bytes Read only—set after completion
CONTROL[8]	(Reserved)	—

Configuring the Data Buffer

The data buffer varies in size. It consists of contiguous registers that include—in sequence—both the CIP request and the CIP response.

<table>
<tr><td rowspan="2">Data Buffer:
Variable size: set in CONTROL[2]</td><td>CIP Request:
Request size: set in CONTROL[6]</td></tr>
<tr><td>CIP Response:
Starting position: set in CONTROL[3]
Response size: reported in CONTROL[7]
Note: If the response offset is smaller than the request size, response data will overwrite part of the request.</td></tr>
</table>

The format of the data buffer's CIP request and CIP response is described, below.

Note: Both the request and response must be structured in little endian order.

Request:

Byte offset	Field	Data type	Description
0	Service	Byte	Service of the explicit message
1	Request_Path_Size	Byte	The number of words in the Request_Path field
2	Request_Path	Padded EPATH	This byte array describes the path of the request—including class ID, instance ID, etc.—for this transaction
...	Request_Data	Byte array	Service specific data to be delivered in the explicit message request—if none, this field is empty

Response:

Byte offset	Field	Data type	Description
0	Reply Service	Byte	Service of the explicit message + 16#80
1	Reserved	Byte	0
2	General Status	Byte	EtherNet/IP General Status
3	Size of Additional Status	Byte	Additional Status array size—in words
4	Additional Status	Word array	Additional status
...	Response Data	Byte array	Response data from request, or additional error data if General Status indicates an error

MBP_MSTR Example - Get_Attributes_Single

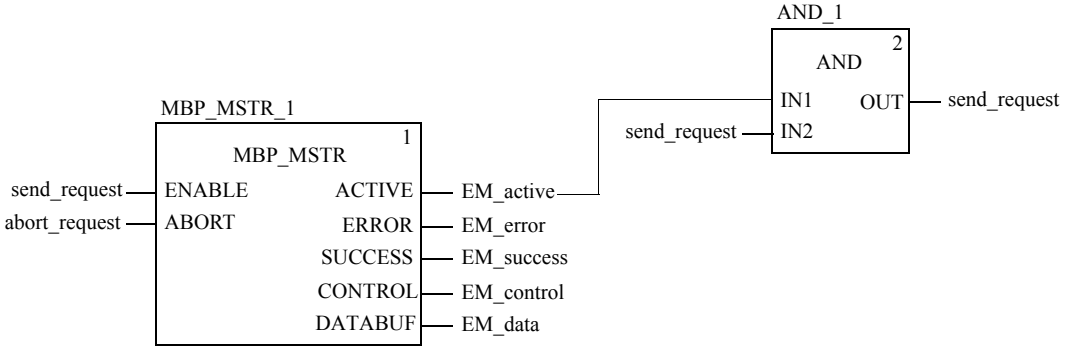
Overview

The following unconnected explicit messaging example shows you how to use the `MBP_MSTR` function block to retrieve diagnostic information for an Advantys STB island from an STB NIC 2212 EtherNet/IP network interface module, using the Get_Attributes_Single service.

You can perform the same explicit messaging service using the *Online Action* window of the Unity Pro EtherNet/IP configuration tool (see *p. 162*).

Implementing the MBP_MSTR Function Block

To implement the `MBP_MSTR` function block, you need to create and assign variables, and connect it to an `AND` block, as follows:



Input Variables

Variables need to be created and assigned to input pins. For the purpose of this example, variables have been created—and named—as described below. (You can, of course, use different variable names in your explicit messaging configurations.)

Input pin	Variable	Data type
`ENABLE`	send_request	`BOOL`
`ABORT`	abort_request	`BOOL`

Output Variables

Variables also need to be created and assigned to output pins. (The names assigned to output variables apply only to this example, and can be changed in your explicit messaging configurations.)

Output pin	Variable	Data type
ACTIVE	EM_active	BOOL
ERROR	EM_error	BOOL
SUCCESS	EM_success	BOOL
CONTROL	EM_control	Array of 9 WORDs
DATABUF	EM_data	Array of 100 WORDs

Note: To simplify configuration, you can assign the CONTROL and DATABUF output pins to a byte array consisting of located variables. When configured in this manner, you will not need to be aware of the location of data within a word (for example, high versus low byte, and big or little endian format).

Control Array

The control array parameter (EM_control) consists of 9 contiguous words. You need to configure only some control words; other control words are read-only and are written to by the operation. In this example, the control array defines the operation as an unconnected explicit message, and identifies the target device:

Register	Description	Configure	Setting (hex)
`CONTROL[0]`	Operation: Low byte = 0E (CIP explicit message) High byte = ● 00 (unconnected), or ● 01 (connected)	Yes	16#000E (unconnected)
`CONTROL[1]`	Error status: read-only (written by operation)	No	—
`CONTROL[2]`	Data buffer length = 100 words	Yes	16#0064
`CONTROL[3]`	Response offset: offset—in words—for the beginning of the explicit message response in the databuffer	Yes	16#0004
`CONTROL[4]`	Slot of the 140 NOC 771 00 module: Low byte = 0 (not used) High byte = slot location	Yes	16#0400
`CONTROL[5]`	Device number: from the **Devices** window of the Unity Pro EtherNet/IP configuration tool	Yes	16#0001
`CONTROL[6]`	CIP request length (in bytes)	Yes	16#0008
`CONTROL[7]`	Length of received response (written by operation)	No	—
`CONTROL[8]`	(*Reserved*)	No	—

CIP Request

The CIP request is located at the beginning of the databuffer and is followed by the CIP response. In this example, the CIP request calls for the return of a single attribute value (diagnostic data), and describes the request path through the target device's object structure leading to the target attribute:

Request word	High byte		Low byte	
	Description	**Value (hex)**	**Description**	**Value (hex)**
1	Request path size (in words)	16#03	EM Service: Get_Attributes_Single	16#0E
2	Request path: class assembly object	16#04	Request path: logical class segment	16#20
3	Request path: instance	16#64	Request path: logical instance segment	16#24
4	Request path: attribute	16#03	Request path: logical attribute segment	16#30

Combining the high and low bytes, above, the CIP request would look like this:

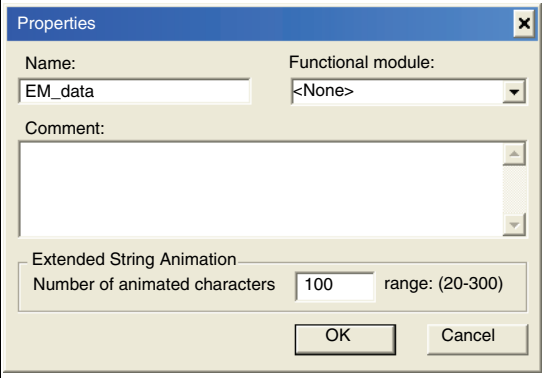
Request word	Value
1	16#030E
2	16#0420
3	16#6424
4	16#0330

Viewing the Response

Use a Unity Pro Animation table to display the EM_data variable array. Note that the EM_data variable array consists of the entire data buffer, which includes the:

- CIP request (4 words) located in EM_data(1-4)
- CIP service type (1 word) located in EM_data(5)
- CIP request status (1 word) located in EM_data(6)
- CIP response (in this case, 10 words) located in EM_data(7-16)

To display the CIP response, follow these steps:

Step	Action	
1	In Unity Pro, select **Tools** → **Project Browser** to open the Project Browser.	
2	In the Project Browser, select the **Animation Tables** folder, then click the right mouse button. A pop-up menu appears.	
3	Select **New Animation Table** in the pop-up menu. A new animation table and its properties dialog both open.	
4	In the Properties dialog, edit the following values:	
	Name	Type in a table name. For this example: **EM_data**.
	Functional module	Accept the default **<None>**.
	Comment	Leave blank.
	Number of animated characters	Type in **100**, representing the size of the data buffer in words.
5	The completed **Properties** dialog looks like this:  Click **OK** to close the dialog.	
6	In the animation table's **Name** column, type in the name of the variable assigned to the databuffer: **EM_data** and hit **Enter**. The animation table displays the EM_data variable.	

Step	Action
7	Expand the EM_data variable to display its word array, where you can view the CIP response at words EM_data(7-16): EM_data Modification Force Extended Strings Name Value Type Comment EM_data ARRAY(1...100) EM_data[1] 16#030E WORD EM_data[2] 16#0420 WORD EM_data[3] 16#6424 WORD EM_data[4] 16#0330 WORD EM_data[5] 16#008E WORD EM_data[6] 16#0000 WORD EM_data[7] 16#10A0 WORD EM_data[8] 16#0000 WORD EM_data[9] 16#00FF WORD EM_data[10] 16#0000 WORD EM_data[11] 16#0000 WORD EM_data[12] 16#0000 WORD EM_data[13] 16#0000 WORD EM_data[14] 16#0000 WORD EM_data[15] 16#00FF WORD EM_data[16] 16#0000 WORD **Note:** Each word presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, ‘0E’ in word[1] is the lower byte, and ‘03’ is the upper byte.

MBP_MSTR Example - Reset

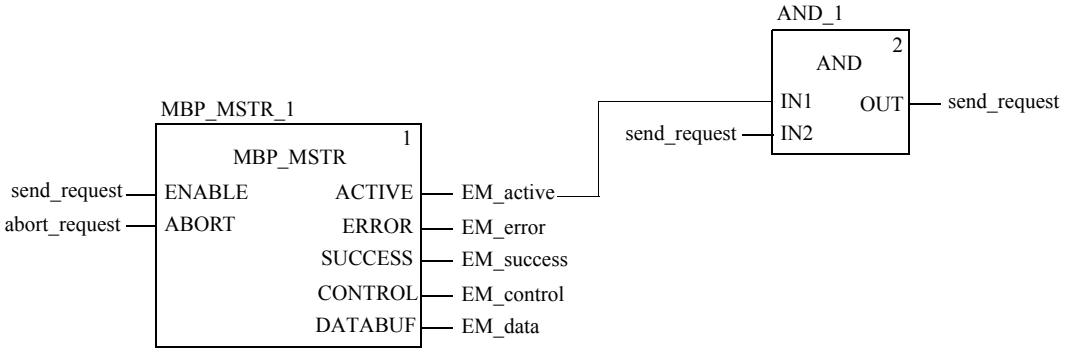
Overview

The following unconnected explicit messaging example shows you how to use the `MBP_MSTR` function block to perform a warm reboot of an STB NIC 2212 EtherNet/IP network interface module, using the Reset service.

You can perform the same explicit messaging service using the *Online Action* window of the Unity Pro EtherNet/IP configuration tool (see *p. 162*).

Implementing the MBP_MSTR Function Block

To implement the `MBP_MSTR` function block, you need to create and assign variables, and connect it to an `AND` block, as follows:



Input Variables

Variables must be created and assigned to input pins as follows:

Input pin	Variable	Data type
`ENABLE`	send_request	`BOOL`
`ABORT`	abort_request	`BOOL`

Output Variables

Variables must be created and assigned to output pins as follows:

Output pin	Variable	Data type
ACTIVE	EM_active	BOOL
ERROR	EM_error	BOOL
SUCCESS	EM_success	BOOL
CONTROL	EM_control	Array of 9 WORDs
DATABUF	EM_data	Array of 5 WORDs

Note: To simplify configuration, you can assign the CONTROL and DATABUF output pins to a byte array consisting of located variables. When configured in this manner, you will not need to be aware of the location of data within a word (for example, high versus low byte, and big or little endian format).

Control Array

The control array parameter (EM_control) consists of 9 contiguous words. You need to configure only some control words; other control words are read-only and are written to by the operation. In this example, the control array defines the operation as an unconnected explicit message, and identifies the target device:

Register	Description	Configure	Setting (hex)
`CONTROL[0]`	Operation: Low byte = 0E (CIP explicit message) High byte = 00 (unconnected)	Yes	16#000E
`CONTROL[1]`	Error status: read-only (written by operation)	No	—
`CONTROL[2]`	Data buffer length = 5 words	Yes	16#0005
`CONTROL[3]`	Response offset: offset—in words—for the beginning of the explicit message response in the databuffer	Yes	16#0005
`CONTROL[4]`	Slot of the 140 NOC 771 00 module: Low byte = 0 (not used) High byte = slot location	Yes	16#0400
`CONTROL[5]`	Device number: from the **Devices** window of the Unity Pro EtherNet/IP configuration tool	Yes	16#0001
`CONTROL[6]`	CIP request length (in bytes)	Yes	16#0008
`CONTROL[7]`	Length of received response (written by operation)	No	—
`CONTROL[8]`	(*Reserved*)	No	—

CIP Request

The CIP request is located at the beginning of the databuffer. In this example, the request calls for a a device reset procedure, and describes the request path through the target device's object structure leading to a target object that performs the requested procedure:

Request word	High byte		Low byte	
	Description	**Value (hex)**	**Description**	**Value (hex)**
1	Request path size (in words)	16#03	EM Service: Reset	16#05
2	Request path: class assembly object	16#01	Request path: logical class segment	16#20
3	Request path: instance	16#01	Request path: logical instance segment	16#24
4	Request path: attribute	16#00	Request path: logical attribute segment	16#30

Combining the high and low bytes, above, the CIP request would look like this:

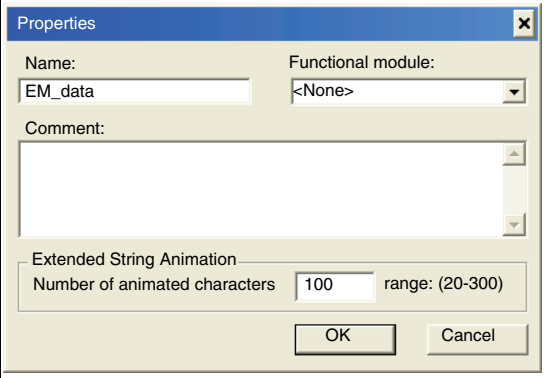
Request word	Value
1	16#0305
2	16#0120
3	16#0124
4	16#0030

Viewing the Response

Use a Unity Pro Animation table to display the EM_data variable array. Because the Reset explicit messaging command returns no data, the EM_data variable includes no CIP response component. The EM_data variable array includes only the:

- CIP request (4 words) located in EM_data(1-4)
- CIP service type (1 word) located in EM_data(5)
- CIP request status (1 word) located in EM_data(6)

To display the contents of the EM_data variable array, follow these steps:

Step	Action	
1	In Unity Pro, select **Tools** → **Project Browser** to open the Project Browser.	
2	In the Project Browser, select the **Animation Tables** folder, then click the right mouse button. A pop-up menu appears.	
3	Select **New Animation Table** in the pop-up menu. A new animation table and its properties dialog both open.	
4	In the Properties dialog, edit the following values:	
	Name	Type in a table name. For this example: **EM_data**.
	Functional module	Accept the default **<None>**.
	Comment	Leave blank.
	Number of animated characters	Type in **100**, representing the size of the data buffer in words.
5	The completed Properties dialog looks like this:  Click **OK** to close the dialog.	
6	In the animation table's *Name* column, type in the name of the variable assigned to the databuffer: **EM_data** and hit **Enter**. The animation table displays the EM_data variable.	

Step	Action
7	Expand the EM_data variable to display its word array, where you can view the CIP response at words EM_data(7-16): EM_data Modification Force Extended Strings Name Value Type Comment EM_data ARRAY(1...100)... EM_data[1] 16#0305 WORD EM_data[2] 16#0120 WORD EM_data[3] 16#0124 WORD EM_data[4] 16#0030 WORD EM_data[5] 16#0085 WORD EM_data[6] 16#0000 WORD **Note:** Each word presents 2 bytes of data in little endian format, where the least significant byte is stored in the smallest memory address. For example, ‘05’ in word[1] is the lower byte, and ‘03’ is the upper byte.

Explicit Messaging Error Codes

If a MBP_MSTR function block fails to execute an explicit message, Unity Pro displays a hexidecimal error code. The error code can describe either:

- an EtherNet/IP error triggered by the failure of the MBP_MSTR block used to execute an explicit message
- legacy TCP/IP and Ethernet errors

For a list of legacy TCP/IP and Ethernet error codes, refer to the list of error codes in Appendix A (see *p. 201*).

EtherNet/IP Error Codes

If an MBP_MSTR function block fails to execute an explicit message, the Unity Pro may display one of the following hexadecimal error codes:

Error Code	Description
16#200F	The space allocated for the CIP response is too small
16#800D	Timeout on the explicit message request
16#8012	Bad device: the device is not configured or the equipment number is too high (>63)
16#8015	Either: ● Nor resources to handle the message, or ● Internal error: no buffer available, no link available, impossible to send to the TCP task
16#8018	Either: ● Another explicit message for this device is in progress, or ● TCP connection or encapsulation session in progress
16#8030	Timeout on the Forward_Open request
Note: The following 16#81xx errors are Forward_Open response errors that originate at the remote target and are received via the CIP connection.	
16#8100	Connection in use or duplicate Forward_Open
16#8103	Transport class and trigger combination not supported
16#8106	Ownership conflict
16#8107	Target connection not found
16#8108	Invalid network connection parameter
16#8109	Invalid connection size
16#8110	Target for connection not configured
16#8111	RPI not supported
16#8113	Out of connections
16#8114	Vendor ID or product code mismatch
16#8115	Product type mismatch
16#8116	Revision mismatch
16#8117	Invalid produced or consumed application path
16#8118	Invalid or inconsistent configuration application path
16#8119	Non-Listen Only connection not opened

Error Code	Description
16#811A	Target object out of connections
16#811B	RPI is smaller than the production inhibit time
16#8123	Connection timed out
16#8124	Unconnected request timed out
16#8125	Parameter error in unconnected request and service
16#8126	Message too large for unconnected_send service
16#8127	Unconnected acknowledge without reply
16#8131	No buffer memory available
16#8132	Network bandwidth not available for data
16#8133	No consumed connection ID filter available
16#8134	Not configured to send scheduled priority data
16#8135	Schedule signature mismatch
16#8136	Schedule signature validation not possible
16#8141	Port not available
16#8142	Link address not valid
16#8145	Invalid segment in connection path
16#8146	Error in Forward_Close service connection path
16#8147	Scheduling not specified
16#8148	Link address to self invalid
16#8149	Secondary resources unavailable
16#814A	Rack connection already established
16#814B	Module connection already established
16#814C	Miscellaneous
16#814D	Redundant connection mismatch
16#814E	No more user-configurable link consumer resources: the configured number of resources for a producing application has reached the limit
16#814F	No more user-configurable link consumer resources: there are no consumers configured for a producing application to use
16#8160	Vendor specific
16#8170	No target application data available
16#8171	No originator application data available
16#8173	Not configured for off-subnet multicast
16#81A0	Error in data assignment
16#81B0	Optional object state error
16#81C0	Optional device state error
Note: All 16#82xx errors are register session response errors.	
16#8200	Target device does not have sufficient resources
16#8208	Target device does not recognize message encapsulation header
16#820F	Reserved or unknown error from target

Explicit Messaging - Online Action: Get_Attributes_Single

Overview

The following example shows you how to use the **Online Action** window in the Unity Pro EtherNet/IP configuration tool to execute an unconnected explicit message that retrieves Advantys STB island diagnostic information from an STB NIC 2212 EtherNet/IP network interface module, using the Get_Attributes_Single service.

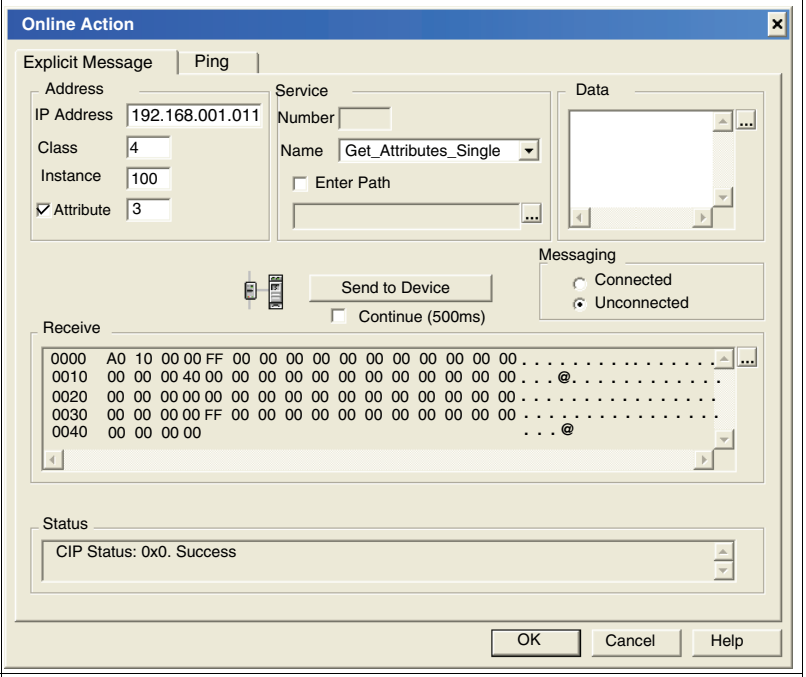
You can perform the same explicit messaging service using the MBP_MSTR function block (see *p. 148*).

Configuring the Explicit Message

To configure, then execute, an unconnected explicit message that will retrieve diagnostic data from the STB NIC 2212 EtherNet/IP module, follow these steps:

Step	Action	
1	Launch the EtherNet/IP configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.	
2	In the EtherNet/IP configuration tool, begin on-line operations by clicking the **Go Online** button .	
3	Click on the **Network Detection** tab to enable online actions	
4	Open the **Online Action** window by selecting **Network** → **Online Action**.	
5	In the **Explicit Messaging** page, complete the following fields:	
	IP Address	Type in the IP address of the STB NIC 2212. In this example, the IP address is: **192.168.001.011**.
	Class	Type in the number that identifies the object class. In this example, the number representing the assembly class object is **4**.
	Instance	Type in the number that identifies the instance of the assembly class object. In this example, the number is **100**.
	Attribute	Place a check mark in the checkbox to enable this field, then type in the number identifying the attribute containing diagnostic data. In this example, the number is **3**.
	Name	Select the name of the explicit messaging service. In this example, select **Get_Attributes_Single**.
	Messaging	Select the type of explicit message. In this example, select **Unconnected**.
	(The explicit messaging configuration is displayed, below.)	
6	To execute the unconnected explicit message, click **Send to Device**.	

Step	Action
7	The **Receive** area displays the message output, and the *Status* area displays the success or failure of the explicit messaging procedure:
8	Click **OK** to close the window.



Explicit Messaging - Online Action: Reset

Overview

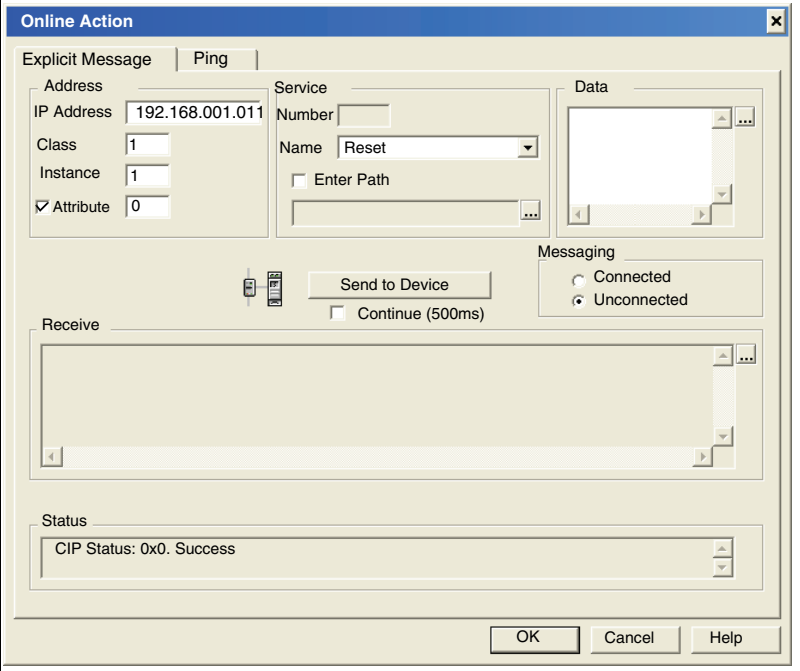
The following example shows you how to use the **Online Action** window in the Unity Pro EtherNet/IP configuration tool to execute an unconnected explicit message that performs a warm reset of the STB NIC 2212 EtherNet/IP network interface module, using the Reset service.

You can perform the same explicit messaging service using the MBP_MSTR function block (see *p. 154*).

Configuring the Explicit Message

To configure, then execute, an unconnected explicit message that will retrieve diagnostic data from the STB NIC 2212 EtherNet/IP module, follow these steps:

Step	Action	
1	Launch the EtherNet/IP configuration tool from the **Configuration** page of the EtherNet/IP communication module's **Properties** window.	
2	In the EtherNet/IP configuration tool, begin on-line operations by clicking the **Go Online** button .	
3	Click on the **Network Detection** tab to enable online actions	
4	Open the **Online Action** window by selecting **Network** → **Online Action**.	
5	In the Explicit Messaging page, complete the following fields:	
	IP Address	Type in the IP address of the STB NIC 2212. In this example, the IP address is: **192.168.001.011**.
	Class	Type in the number that identifies the object class. In this example, the number representing the assembly class object is **1**.
	Instance	Type in the number that identifies the instance of the assembly class object. In this example, the number is **1**.
	Attribute	Place a check mark in the checkbox to enable this field, then type in the number identifying the attribute associated with the reset function. In this example, the number is **0**.
	Name	Select the name of the explicit messaging service. In this example, select **Reset**.
	Messaging	Select the type of explicit message. In this example, select **Unconnected**.
	(The explicit messaging configuration is displayed, below.)	

Step	Action
6	To execute the unconnected explicit message, click **Send to Device**.
7	The **Status** area displays the success or failure of the explicit messaging procedure:  **Note**: Because the service returns no data, the **Receive** area displays no message output.
8	Click **OK** to close the window.

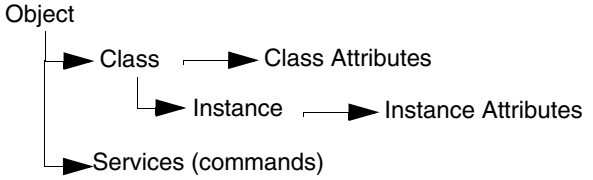
CIP Objects

6

At a Glance

Overview

The EtherNet/IP communication module stores data and offers services in a CIP object hierarchy, consisting of the following nested levels:



When the module's local slave service is activated, remote devices can send explicit messages to the module's object hierarchy and perform services that:

- access module data, or
- execute module commands

The local slave function is activated by selecting **Active Configuration** in the **General Configuration** page of the **Local Slave** window (see *p. 61*).

This chapter describes the CIP objects the EtherNet/IP communication module can expose to remote devices.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Adapter Diagnostic Object	168
Assembly Object	172
Connection Manager Object	173
Ethernet Link Object	175
Identity Object	178
Module Diagnostic Object	180
Scanner Diagnostic Object	182
TCP/IP Interface Object	186

Adapter Diagnostic Object

Overview

The Adapter Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Adapter Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Control Bits	WORD	X	X	0: Deactivate checking time for production and consumption (default) 1: Activate
02	ST_DIAG_CNT	STRUCT	X	X	
	wErrFrameCnt	UINT			Incremented each time a frame isn't sent by missing resources or is impossible to send.
	wErrTimeOutCnt	UINT			Incremented when a connection is timed out.
	wErrRefusedCnt	UINT			Incremented when a connection is refused by the remote station.
	dwErrProdCnt	UDINT			Incremented at each production.
	dwErrConsCnt	UDINT			Incremented at each consumption.
	dwErrProdByteCnt	UDINT			Total bytes produced.
	dwErrConsByteCnt	UDINT			Total bytes consumed.
03	Input Status	WORD	X	—	See Status descriptions, below.
04	Output Status	WORD	X	—	See Status descriptions, below.
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
05	ST_LINK	STRUCT	X	—	
	CIP Status	UINT			See Status descriptions, below.
	Extended Status	UINT			See Status descriptions, below.
	Production Connection ID	DWORD			Connection ID
	Consumed Connection ID	DWORD			Connection ID
	OtoT API	UDINT			API of the Connection
	TtoO API	UDINT			API of the Connection
	OtoT RPI	UDINT			RPI of the Connection
	TtoO RPI	UDINT			RPI of the Connection
06	ST_SOCK_PARAM	STRUCT	X	—	
	IpSockId	DWORD			Internal identifier
	IpForeign	DWORD			IP of the remote station
	wPortForeign	UINT			Port number of the remote station
	IpLocal	DWORD			IP of the local station
	wPortLocal	UINT			Port number of the local station
07	ST_PRODUCTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct production is not valid 1: data of the struct production is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before next production)
	dwProductionTime	UDINT			(Internal Use—number of ticks between production)
	SequenceNumber	UDINT			Number of the dwquence in the production
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 productions
	dwMinTime	UDINT			Minimum time between 2 productions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the production was too long
	wUnderRun	UINT			Number of times the production was too short
	dwCurrentTime	UDINT			(Internal Use)
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
08	ST_CONSUMPTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct consumption is not valid 1: data of the struct consumption is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before the timeout)
	dwConsumptionTime	UDINT			(Internal Use—number of ticks in the timeout)
	SequenceNumber	UDINT			Number of the sequence in the consumption
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 consumptions
	dwMinTime	UDINT			Minimum time between 2 consumptions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the consumption was too long
	wUnderRun	UINT			Number of times the consumption was too short
	dwCurrentTime	UDINT			(Internal Use)
09	Connection Entry List	STRUCT	X	—	Status of the CCO object. See Status descriptions, below.
	byGeneralStatus	BYTE			
	byReserved	BYTE			
	Extended Status	WORD			
X = supported **— = not supported**					

Adapter Status Adapter status values include the followng:

Status	Description	CIP Status	Extended	Explanation
0	OK	0	0	The I/O data are correctly exchanged.
33	No connection	0	0	No connection.
		0xFB	0xFB01	Connection in timeout.
		0xFB	0xFB07	Optimization error / unknown MAC Address.
		0xFB	0xFB0B	Timeout on consumption.
		0xFB	0xFB0C	Connection closed by a Fw_Close.
		0xFB	0xFB0E	Module in STOP.
		0xFD		Error from encapsulation layer.
		0xFE		Error on TCP connection.
		0x02	0	No more resources to handle the connection.
		0x20	0	Connections refused by bad format or parameters.
53	IDLE	0	0	An IDLE notification is received.
54	Connection in progress	0	0	The connection is established, but I/O data is not yet consumed.

Services The CIP Adapter Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes		
01	Get_Attributes_All	X	X			
61	Get_Output	—	X	Returns the status and values of the output:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Output Data
62	Get_Intput	—	X	Returns the status and values of the input:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Input Data
63	Set_DiagCounters	—	X	Sets the values of the structure: ● ST_DIAG_CNT to 0, and ● ST_CHECK_TIME (production and consumption) to 0 (but not fields dwLastTime and dwCurrentTime) the structure ST_DIAG_CNT to 0.		
X = supported **— = not supported**						

Assembly Object

Overview

The Assembly CIP object consists of the attributes and services described below.

Attributes

The Assembly CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET
03	Data	Array of BYTE	X	X
X = supported **— = not supported**				

Services

The CIP Assembly object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
05	Set_Attributes_Single	—	X	Returns these values: 0E=attribute not settable: assembly is not o->T type 0F=permission denied: assembly is being used by an active connection 13=config too small: the Set_Attributes_Single command contains partial data 15=too big data: the Set_Attributes_Single command contains too much data
X = supported **— = not supported**				

Connection Manager Object

Overview

The Connection Manager CIP object consists of the attributes and services described below.

Attributes

The Connection Manager CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Open Requests	UINT	X	X	Number of Forward Open service requests received
02	Open Format Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to bad format
03	Open Resource Rejects	UINT	X	X	Number of Forward Open service requests that were rejected due to lack of resources
04	Open Other Rejects	UINT	X	X	Number of Forward Open service requests that were rejected for reasons other than bad format or lack of resources
05	Close Requests	UINT	X	X	Number of Forward Close service requests received
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
06	Close Format Requests	UINT	X	X	Number of Forward Close service requests that were rejected due to bad format
07	Close Other Requests	UINT	X	X	Number of Forward Close service requests that were rejected for reasons other than bad format
08	Connection Timeouts	UINT	X	X	Total number of connection timeouts that occurred in connections controlled by this connections manager
09	Connection Entry List	STRUCT	X	—	List of connections—always 0
11	CPU_Utilization	UINT	X	—	CPU Utilization in tenths of a percent—always 0
12	MaxBuffSize	UDINT	X	—	Amount of buffer space originally available—always 0
13	BufSize Remaining	UDINT	X	—	Amount of buffer space now available—always 0
X = supported **— = not supported**					

Services

The CIP Connection Manager object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
4E	Forward Close	—	X	Managed internally by the EtherNet/IP stack—no link to CPU exists
52	Unconnected Send	—	X	
54	Forward Open	—	X	
X = supported **— = not supported**				

Ethernet Link Object

Overview

The Ethernet Link CIP object consists of the attributes and services described below.

Attributes

The Ethernet Link CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Interface Speed	UDINT	X	—	Valid values include: 0, 10000000, 100000000
02	Interface Flags	DWORD	X	—	Bit 0: Link Status 0 = Inactive 1 = Active
					Bit 1: Duplex Mode 0 = half duplex 1 = full duplex
					Bits 2-4: Negotiation Status 3 = successfully negotiated speed and duplex 4 = forced speed and link
					Bit 5: Manual Setting Requires Reset 0 = automatic 1 = device need reset
					Bit 6: Local Hardware Fault 0 = no fault 1 = fault detected
03	Physical Address	ARRAY of 6 USINT	X	—	Module MAC Address
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
04	Interface Counters	STRUCT	X	—	
	In octets	UDINT			Octets received on the interface
	In Ucast Packets	UDINT			Unicast packets received on the interface
	In NUcast Packets	UDINT			Non-unicast packets received on the interface
	In Discards	UDINT			Inbound packets received on the interface, but discarded
	In Errors	UDINT			Inbound packets that contain errors (does not include In Discards)
	In Unknown Protos	UDINT			Inbound packets with unknown protocol
	Out Octets	UDINT			Octets sent on the interface
	Out Ucast Packets	UDINT			Unicast packets sent on the interface
	Out NUcast Packets	UDINT			Non-unicast packets sent on the interface
	Out Discards	UDINT			Outbound packets discarded
	Out Errors	UDINT			Outbound packets that contain errors
05	Media Counters	STRUCT	X	—	
	Alignment Errors	UDINT			Frames that are not an integral number of octets in length
	FCS Errors	UDINT			Frames received that do not pass the FCS check
	Single Collisions	UDINT			Successfully transmitted frames that experienced exactly one collision
	Multiple Collisions	UDINT			Successfully transmitted frames that experienced more than one collision
	SQE Test Errors	UDINT			Number of times the SQE test error is generated
	Deferred Transmissions	UDINT			Frames for which first transmission attempt is delayed because the medium is busy
	Late Collisions	UDINT			Number of times a collision is detected later than 512 bittimes into the transmission of a packet
	Excessive Collisions	UDINT			Frames for which transmission fails due to excessive collisions
	MAC Transmit Errors	UDINT			Frames for which transmission fails due to internal MAC sublayer transmit error
	Carrier Sense Errors	UDINT			Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame
	Frame Too Long	UDINT			Frames received that exceed the maximum permitted frame size
	MAC Receive Errors	UDINT			Frames for which reception on an interface fails due to an internal MAC sublayer receive error
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
06	Interface Control	STRUCT	X	X	API of the connection
	Control Bits	WORD			Bit 0: Auto-negotiation 0 = Disabled 1 = Enabled **Note:** When auto-negotiation is enabled, the error 0x0C (Object State Conflict) is returned when attempting to set either: ● Forced Interface Speed, or ● Forced Duplex Mode
					Bit 1: Forced Duplex Mode (if auto-negotiation bit = 0) 0 = half duplex 1 = full duplex
	Forced Interface Speed	UINT			Valid values include: 10000000, 100000000 **Note:** Attempting to set any other value returns the error 0x09 (Invalid Attribute Value)
X = supported **— = not supported**					

Services

The CIP Ethernet Link object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance
01	Get_Attributes_All	X	X
05	Set_Attribute_Single	—	X
0E	Get_Attribute_Single	X	X
X = supported **— = not supported**			

Identity Object

Overview

The Identity CIP object consists of the attributes and services described below.

Attributes

The Identity CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET
01	Vendor ID	UINT	X	—
02	Device Type	UINT	X	—
03	Product Code	UINT	X	—
04	Revision	STRUCT	X	—
	Major	USINT		
	Minor	USINT		
05	Status bit 2: 0x01=the module is configured bits 4-7: 0x03=no I/O connections established 0x06=at least 1 I/O connection in run mode 0x07=at least 1 I/O connection established, all in IDLE mode	Word	X	—
06	Serial Number	UDINT	X	—
07	Product Name	STRING	X	—
X = supported **— = not supported**				

Services

The CIP Identity object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	Applies to all class and all instance attributes
0E	Get_Attributes_Single	X	X	Applies to all class and all instance attributes
05	Reset	—	X	Two types: 00=power cycle 01=return to factory defaults and power cycle
X = supported **— = not supported**				

Module Diagnostic Object

Overview

The Module Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Module Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Module Status	WORD	X	—	01=started 02=stopped 03=running
02	CNF Version	WORD	X	—	0x0100
03	CRC	UDINT	X	—	—
04	I/O Connection Status	STRUCT	X	—	—
	Size Table	WORD			size (16 bytes)
	Table	WORD[]			table of I/O status (8 WORDS) 1=INPUT and OUTPUT status of I/O connection are OK 0=at least 1 INPUT or OUTPUT status of I/O connection is not OK
05	Cco Mode	WORD	X	X	01=activate status to CCO in the module 02=block access to CCO
X = supported **— = not supported**					

Services

The CIP Module Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Set_Attributes_Single	—	X	—
X = supported **— = not supported**				

Scanner Diagnostic Object

Overview

The Scanner Diagnostic CIP object consists of the attributes and services described below.

Attributes

The Scanner Diagnostic CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Control Bits	WORD	X	X	0: Deactivate checking time for production and consumption (default) 1: Activate
02	ST_DIAG_CNT	STRUCT	X	X	
	wErrFrameCnt	UINT			Incremented each time a frame isn't sent by missing resources or is impossible to send.
	wErrTimeOutCnt	UINT			Incremented when a connection is timed out.
	wErrRefusedCnt	UINT			Incremented when a connection is refused by the remote station.
	dwErrProdCnt	UDINT			Incremented at each production.
	dwErrConsCnt	UDINT			Incremented at each consumption.
	dwErrProdByteCnt	UDINT			Total bytes produced.
	dwErrConsByteCnt	UDINT			Total bytes consumed.
03	Input Status	WORD	X	—	See Status descriptions, below.
04	Output Status	WORD	X	—	See Status descriptions, below.
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
05	ST_LINK	STRUCT	X	—	
	CIP Status	UINT			See Status descriptions, below.
	Extended Status	UINT			See Status descriptions, below.
	Production Connection ID	DWORD			Connection ID
	Consumed Connection ID	DWORD			Connection ID
	OtoT API	UDINT			API of the Connection
	TtoO API	UDINT			API of the Connection
	OtoT RPI	UDINT			RPI of the Connection
	TtoO RPI	UDINT			RPI of the Connection
06	ST_SOCK_PARAM	STRUCT	X	—	
	IpSockId	DWORD			Internal identifier
	IpForeign	DWORD			IP of the remote station
	wPortForeign	UINT			Port number of the remote station
	IpLocal	DWORD			IP of the local station
	wPortLocal	UINT			Port number of the local station
07	ST_PRODUCTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct production is not valid 1: data of the struct production is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before next production)
	dwProductionTime	UDINT			(Internal Use—number of ticks between production)
	SequenceNumber	UDINT			Number of the dwquence in the production
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 productions
	dwMinTime	UDINT			Minimum time between 2 productions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the production was too long
	wUnderRun	UINT			Number of times the production was too short
	dwCurrentTime	UDINT			(Internal Use)
X = supported **— = not supported**					

ID (hex)	Description	Type	GET	SET	Value
08	ST_CONSUMPTION	STRUCT	X	—	
	bValid	WORD			0: data of the struct consumption is not valid 1: data of the struct consumption is valid
	dwCurrentTime	UDINT			(Internal Use—number of ticks before timeout)
	dwConsumptionTime	UDINT			(Internal Use—number of ticks of the timeout)
	SequenceNumber	UDINT			Number of the sequence in the consumption
	stCheckTime	STRUCT			
	dwLastTime	UDINT			(Internal Use)
	dwMaxTime	UDINT			Maximum time between 2 consumptions
	dwMinTime	UDINT			Minimum time between 2 consumptions
	dwRPI	UDINT			API of the connection
	wOverRun	UINT			Number of times the consumption was too long
	wUnderRun	UINT			Number of times the consumption was too short
	dwCurrentTime	UDINT			(Internal Use)
09	Connection Entry List	STRUCT	X	—	Status of the CCO object. See Status descriptions, below.
	byGeneralStatus	BYTE			
	byReserved	BYTE			
	Extended Status	WORD			
X = supported **— = not supported**					

Scanner Status Scanner status values include the followng:

Status	Description	CIP Status	Extended	Explanation
0	OK	0	0	The I/O data are correctly exchanged.
33	Timeout	0xFB	0xFB0B	Timeout detected on consumption.
53	IDLE	0	0	An IDLE notification is received.
54	Connection established	0	0	The connection is established, but I/O is not yet consumed.
		0xFB	0xFB08	Impossible to start the production.
		0xFB	0xFB09	Impossible to start the consumption.
		0xFB	0xFB0A	Not enough resources to manage the connection.
58	Not connected (TCP)	0xFE		Error on TCP connection.
65	Not connected (CIP)	0xFB	0xFB01	Timeout for Fw_Open response.
		0xFB	0xFB02	Bad format of the Fw_Open response (so addr).
		0xFB	0xFB03	Bad parameters in the Fw_Open response (OT Net Par).
		0xFB	0xFB04	Bad parameters in the Fw_Open response (TO Net Par).
		0xFB	0xFB05	Fw_Open respons asks for port number other than 2222.
		0xFB	0xFB06	Error joining the UDP multicast group.
		0xFB	0xFB07	Optimization error / unknown MAC Address.
68	Connection establishing	0xD0	0x0001	Connection is closed.
		0xD0	0x0002	Connection is pending.
70	Not connected (EPIC)	0xFD		Error code in register session response.
		0xFD		Error code in the frame.
		0xFD		Encapsulation session un-registered.
77	Scanner stopped	0	0	Connection is stopped.

Services The CIP Scanner Diagnostic object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes		
01	Get_Attributes_All	X	X			
61	Get_Output	—	X	Returns the status and values of the output:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Output Data
62	Get_Input	—	X	Returns the status and values of the input:		
				Offset	Type	Description
				0	UINT	Status
				2	USINT[0...409]	Input Data
63	Set_DiagCounters	—	X	Sets the value of the structure ST_DIAG_CNT to 0.		
X = supported **— = not supported**						

TCP/IP Interface Object

Overview

The Identity CIP object consists of the attributes and services described below.

Attributes

The TCP/IP Interface CIP object consists of the following attributes:

1. Class attributes:

ID (hex)	Description	GET	SET
01	Revision	X	—
02	Max Instance	X	—
X = supported **— = not supported**			

2. Instance attributes:

ID (hex)	Description	Type	GET	SET	Value
01	Status	DWORD	X	—	always = 0x01
02	Configuration Capability	DWORD	X	—	0x01 = from BootP 0x11 = from flash 0x00 = other
03	Configuration Control	DWORD	X	X	0x01 = out-of-box default
04	Physical Link Object	STRUCT	X	—	
	Path Size	UINT			
	Path	Padded EPATH			
05	Interface Configuration	STRUCT	X	X	0x00 = out-of-box default
	IP Address	UDINT			
	Network Mask	UDINT			
	Gateway Address	UDINT			
	Name Server	UDINT			
	Name Server 2	UDINT			
	Domain Name	STRING			
06	Host Name	STRING	X	—	
X = supported **— = not supported**					

Services

The CIP TCP/IP Interface object performs the following services upon the listed object types:

ID (hex)	Description	Class	Instance	Notes
01	Get_Attributes_All	X	X	—
0E	Get_Attributes_Single	X	X	—
05	Get_Attributes_Single	—	X	—
X = supported **— = not supported**				

Diagnostics

7

At a Glance

Overview

This chapter describes the diagnostic features of the EtherNet/IP communication module and the Unity Pro EtherNet/IP configuration software.

What's in this Chapter?

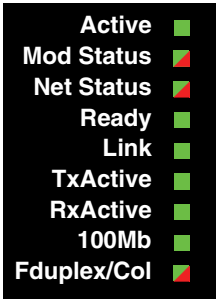
This chapter contains the following topics:

Topic	Page
LED Indicators for the 140 NOC 771 00 EtherNet/IP Communication Module	190
Diagnostic Testing Using the Unity Pro EtherNet/IP Software	192
Ping a Network Device	194
Viewing Output Messages in the Unity Pro EtherNet/IP Configuration Tool	195

LED Indicators for the 140 NOC 771 00 EtherNet/IP Communication Module

LED Indicators

The 140 NOC 771 00 displays the following LED indicators:



LED Descriptions

Use the LED display to diagnose the state of the module, as follows:

LED	Color	Description
Active	Green	• **Off:** Indicates that the module is not communicating with the CPU over the backplane. • **Steady Green:** Indicates that the module is communicating with the CPU over the backplane.
Mod Status (Module Status)	Green/ Red	• **Off:** Power is not being supplied to the module. • **Steady Green:** The module is operating normally. • **Flashing Green:** The module has not been configured. • **Steady Red:** The module has detected a major fault. • **Flashing Red:** The module has detected a recoverable fault.
Net Status (Network Status)	Green/ Red	• **Off:** Power is not being supplied to the module or the module does not have an IP address assigned. • **Steady Green:** The module has established at least one CIP connection. • **Flashing Green:** The module has obtained an IP address but has not established any CIP connections. • **Steady Red:** The module has detected that its IP address is a duplicate IP address. • **Flashing Red:** One or more CIP connections has timed out and the connection(s) need to be re-established or the module has been reset.

LED	Color	Description
Ready	Green	**Steady Green:** Indicates that the module is being configured and is ready to start operating. The Ready LED also provides diagnostic information by using the following sequence of flashes. ● **Two flashes:** The module has an invalid MAC address. ● **Three flashes:** The Ethernet link is not connected ● **Four flashes:** The module has detected a duplicate IP address. ● **Five flashes:** The module is waiting for a served IP configuration. ● **Six flashes:** The module is using its default IP configuration. ● **Seven flashes:** The module has detected a configuration error.
Link	Green	● **Off:** An Ethernet link has not been established. ● **Steady Green:** The module has an Ethernet link.
TxActive (Transmission Activity)	Green	● **Off:** There is no transmission activity. ● **Flashes Green:** Indicates transmission activity.
RxActive (Reception Activity)	Green	● **Off:** There is no reception activity. ● **Flashes Green:** Indicates reception activity.
100Mb	Green	**Steady Green:** The module connected with a 100Mb Ethernet link.
FDuplex/Col (Full Duplex/ Collision)	Green/ Red	● **Off:** The module is not connected to a full duplex Ethernet link. ● **Steady Green:** Indicates that the module is connected with a full duplex link (it can transmit and receive at the same time.) ● **Flashing Red:** A collision has been detected on the Ethernet link.

Diagnostic Testing Using the Unity Pro EtherNet/IP Software

Overview

Use the Unity Pro EtherNet/IP configuration tool to perform a diagnostic test of the EtherNet/IP module and all other devices in your configuration.

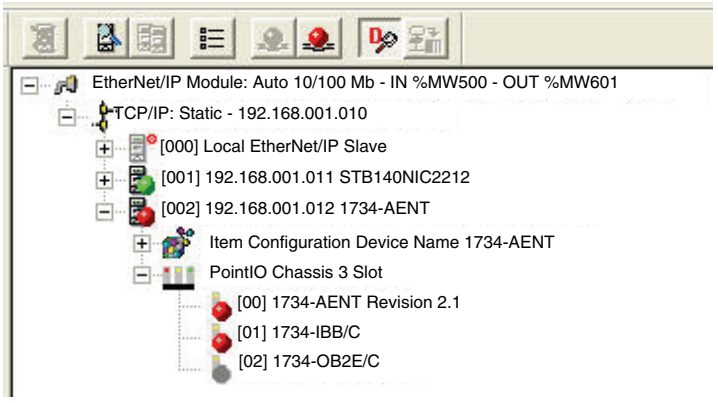
Note: Diagnostic testing is performed directly between the Unity Pro EtherNet/IP configuration tool software running on your PC and the target EtherNet/IP device.

The software displays the results of the diagnostic test, as follows:

- the task bar's **Module State** indicator reads **Diagnostic**
- the **Devices** window depicts the state of connections for each device using a:
  - green icon, indicating all connections are functioning
  - red icon, indicating at least one connection has failed
  - gray icon, indicating a rack optimized module connection
- a diagnostic tab is added to the properties window for each EtherNet/IP device and I/O module displaying:
  - each connection's status, information, and performance data, and
  - the value of each input and output

Performing a Diagnostic Test

To perform a diagnostic test in the Unity Pro EtherNet/IP software:

Step	Action
1	Do one of the following: ● click the Diagnostics toolbar button , or ● select **Devices** → **Diagnostic...** The configuration tool enters its diagnostic state.
2	The EtherNet/IP module enters a diagnostic state and displays the status of each connected device and module. An example of a diagnostic status display appears, below:  In the above example: ● a green icon indicates that all connections are functioning for the device at address [001] ● a red icon indicates at least one connection has failed for the device at address [002] ● red icons indicates that at least one connection has failed for the modules at slots [00] and [01] ● a gray icon indicates a rack optimized connection exists for the module at slot [02]
3	To exit diagnostic mode, repeat the command in step 2.

Ping a Network Device

Overview

Use the Unity Pro EtherNet/IP configuration tool's Ping function to send an ICMP echo request to a target EtherNet/IP device to determine:

- if the target device is present, and if so
- the elapsed time to receive an echo response from the target device

The target device is identified by its IP address setting. The Unity Pro EtherNet/IP configuration tool will verify that the target address is not a:

- loopback address (127.000.000.000 to 127.255.255.255)
- multicast address (224.000.000.000 to 239.255.255.255)
- reserved address (240.000.000.000 to 255.255.255.255)
- broadcast address

The ping function can be performed from either the:

- **General** page of a device's properties window
- **Ping** page of the **Online Action** window

Pinging a Network Device

To ping a network device:

Step	Action
1	Be sure the Unity Pro EtherNet/IP configuration tool is operating online.
2	Do one of the following: ● Select **Network** → **Online Action**, then click on the **Ping** page, or ● Select a device in the **Devices** window, then select **Devices** → **Properties**
3	If you are working in the Ping page of the Online Action window, type in the **IP Address** of the target device. **Notes**: ● The default is the IP address of the device currently selected in the **Network Detection** list. ● If you are working in the **General** page of a device's **Properties** window, the Unity Pro EtherNet/IP configuration tool uses the IP address of the device selected in the **Devices** window.
4	To send... ● a single ping, de-select the **Loop** checkbox ● a series of pings—1 every 100 ms—select **Loop**
5	(Optional) Select **Stop on Error** to stop pinging if an error occurs.
6	Click **Ping** once to begin pinging.
7	Click **Ping** a second time to stop looped pinging, where no error has been detected.

Viewing Output Messages in the Unity Pro EtherNet/IP Configuration Tool

Overview

Use the Unity Pro EtherNet/IP configuration tool's **Output Message** window to diagnose the health of your EtherNet/IP network. This window maintains a log of network events. You can:

- show or hide the window
- display for each item in the window its:
  - date and time
  - level of significance
- copy the contents of the **Output Message** window to your PC's Windows Clipboard
- clear the contents of the window

Show/Hide the Output Message Window

The **Output Message** window is displayed in the Unity Pro EtherNet/IP configuration tool by default. To hide the window, select:
File → **Preferences** → **Output Window**.

To reopen the **Output Message** window, repeat the above command.

Add Date/Time and Level to Output Message Window Items

To show or hide the date and time, or level of significance for **Output Message** window entries:

Step	Action
1	Select **File** → **Message View** → **Configuration**. The Output Message View Configuration dialog opens.
2	Select—or de-select—either or both: ● **Add Date to Messages** ● **Add Level to Messages**
3	Click **OK**.

Copy/Clear

To copy the contents of the **Output Message** window to your PC's Windows Clipboard, select: **File** → **Message View** → **Copy**.

To clear the contents of the **Output Message** window select:
File → **Message View** → **Clear**.

Replacing the EtherNet/IP Communication Module

8

Replacing the EtherNet/IP Communication Module

Overview

You can replace the EtherNet/IP communication module at any time using another module with compatible firmware. A module can be replaced when power to the module is either:

- off (cold swap), or
- on (hot swap)

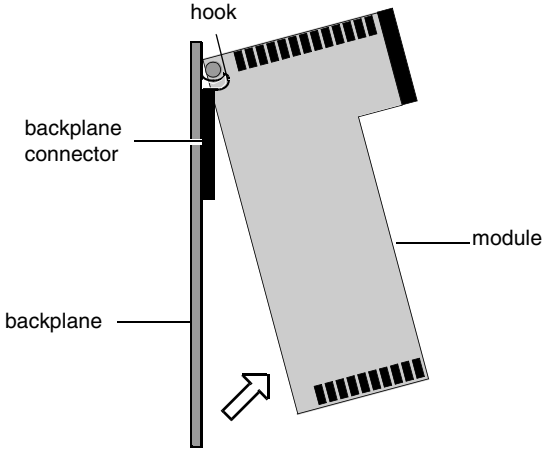
The replacement module obtains its operating parameters over the backplane connection from the CPU. The transfer occurs either immediately (hot swap) or when power is next cycled to the device (cold swap).

Note: The operating parameters that the CPU sends to a replacement module do not include any parameter values that were edited in the original module using explicit messaging "SET" commands. Explicit messaging can be performed in the Online Action window of the Unity Pro EtherNet/IP configuration tool.

Replacing the Communication Module

Replacing the module involves removing the old module and mounting a new one in its place.

To remove a module:

Step	Action
1	Use a Phillips-head screw driver to detach the safety screw, located at the lower front side of the module, from the backplane.
2	Swing the bottom of the module away and up from the backplane, pivoting it on the hooks holding the module at the top of the backplane, until the module detaches from backplane connector: 
3	Lift the module up and off of the hooks located at the top of the backplane.

To install the replacement module, follow the instructions in the module mounting procedure (see *p. 14*).

Appendices

At a Glance

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Error Codes	201

Error Codes

A

TCP/IP Ethernet Error Codes

TCP/IP Ethernet Error Codes

An error in an `MSTR` routine via TCP/IP Ethernet may produce one of the following errors in the `MSTR` control block:

The error code appears as **Mmss**, where:

- **M** is the high code
- **m** is the low code
- **ss** is a subcode

Hexadecimal Error Codes TCP/IP Ethernet

Hexadecimal error codes TCP/IP Ethernet:

Hex. Error Code	Meaning
1001	Abort by user
2001	An operation type that is not supported has been specified in the control block
2002	One or more control block parameters were modified while the `MSTR` element was active (this only applies to operations which require several cycles for completion). Control block parameters my only be modified in inactive `MSTR` components.
2003	Invalid value in the length field of the control block
2004	Invalid value in the offset field of the control block
2005	Invalid value in the length and offset fields of the control block
2006	Unauthorized data field on slave
2008	Unauthorized network routing path on slave
200E	The control block is not assigned, or parts of the control block are located outside of the %MW (4x) range.
3000	Generic Modbus failure code
30ss	Exceptional response by Modbus slave (see *p. 202*)
4001	Inconsistent response by Modbus slave

ss Hexadecimal Value in 30ss Error Code

ss hexadecimal value in 30ss error code:

ss hex. Value	Meaning
01	Slave does not support requested operation
02	Non-existing slave registers were requested
03	An unauthorized data value was requested
05	Slave has accepted a lengthy program command
06	Function cannot currently be carried out: lengthy command running
07	Slave has rejected lengthy program command

Hexadecimal Error Codes TCP/IP Ethernet Network

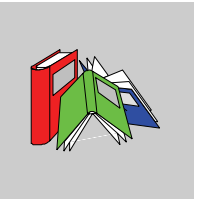
An error on the TCP/IP Ethernet network itself may produce one of the following errors in the `CONTROL[1]` register of the control block.

Hexadecimal error codes TCP/IP Ethernet network:

Hex. Error Code	Meaning
5004	Interrupted system invocation
5005	I/O error
5006	No such address
5009	The socket descriptor is not valid
500C	Not enough storage space
500D	Authorization denied
5011	Entry exists
5016	An argument is not valid
5017	An internal table has no more space
5020	There is interference on the connection
5023	This operation was blocked and the socket is non-blocking
5024	The socket is non-blocking and the connection cannot be closed down
5025	The socket is non-blocking and a previous connection attempt has not been concluded
5026	Socket operation on a non-socket
5027	The destination address is not valid
5028	Message too long
5029	Wrong type of protocol for the socket
502A	Protocol not available
502B	Protocol not supported
502C	Socket type not supported
502D	Operation not supported at socket
502E	Protocol family not supported
F502	Address family not supported
5030	Address is already in use
5031	Address not available
5032	Network is out of order
5033	Network cannot be reached
5034	Network shut down the connection during reset
5035	The connection was terminated by the peer
5036	The connection was reset by the peer

Hex. Error Code	Meaning
5037	An internal buffer is required, but cannot be assigned
5038	The socket is already connected
5039	The socket is not connected
503A	Cannot transmit after the socket has been shut off
503B	Too many references; cannot splice
503C	Connection timed out
503D	The connection attempt was denied
5040	Host is out of order
5041	The destination host could not be reached from this node
5042	Directory not empty
5046	NI_INIT returned -1
5047	The MTU is not valid
5048	The hardware length is not valid
5049	The route specified cannot be found
504A	Collision when invoking Select; these conditions have already been selected by another job
504B	The job ID is not valid
5050	No Network Resource
5051	Length Error
5052	Addressing Error
5053	Application Error
5054	Client cannot process request
5055	No Network Resource
5056	Non-Operational TCP connection
5057	Incoherent configuration
6003	FIN or RST not expected
F001	In reset mode
F002	Component not fully initialized

Glossary



A

Adapter

(*I/O adapter*) An I/O adapter is the target of real-time I/O data connection requests from I/O scanners. It cannot send or receive real-time I/O data unless it is requested to do so by an I/O scanner, and it does not store or originate the data communications parameters necessary to establish the connection. An I/O adapter receives explicit message requests (connected and unconnected) from other devices. It can also exchange (peer) data using explicit messages with other devices, but cannot originate such relationships.

Advance mode

In the Unity Pro EtherNet/IP configuration tool, Advance mode is a selection that displays additional configuration properties, which appear in the:

- **EtherNet/IP** page of the **Channel Properties** window
- **Module Information** page of the **Channel Properties** window
- **Module Information** page of the EtherNet/IP device properties window

Advance mode is enabled by selecting **File** → **Preferences** → **Advance**.

B

BOOTP

(*bootstrap protocol*) A UDP network protocol that can be used by a network client to automatically obtain an IP address (and possibly other data) from a server. The client identifies itself to the server using the client's MAC address. The server—which maintains a pre-configured table of client device MAC addresses and associated IP addresses—sends the client its pre-configured IP address.

BOOTP originally was used as a method to enable diskless hosts to be remotely booted over a network. The BOOTP process assigns an infinite lease of an IP address. The BOOTP service utilizes UDP ports 67 and 68.

C

CIP

(*common industrial protocol*) A comprehensive suite of messages and services for the collection of manufacturing automation applications—control, safety, synchronization, motion, configuration and information. CIP allows users to integrate these manufacturing applications with enterprise-level Ethernet networks and the Internet.

class 1 connection

A CIP transport class 1 connection used for I/O data transmission via implicit messaging.

class 3 connection

A CIP transport class 3 connection used for explicit messaging.

connected messaging

Connected messaging uses a CIP connection for communication over UDP. A connected message is a relationship between two or more application objects on different nodes. The connection establishes a virtual circuit in advance for a particular purpose, such as frequent explicit messages or real-time I/O data transfers.

connection

A virtual circuit between two or more network devices, created prior to the transmission of data. After a connection is established, a series of data is transmitted over the same communication path, without the need to include routing information—including source and destination address—with each piece of data.

connection originator

The network node that initiates a connection establishment request for I/O data transfer or explicit messaging.

connectionless

Describes communication between two network devices, whereby data is sent without prior arrangement between the two devices. Each piece of transmitted data also includes routing information—including source and destination address.

consumer

See producer/consumer, below.

CSMA/CD

(*carrier sense multiple access with collision detection*) An Ethernet and IEEE 802.3 media access method. All network devices contend equally for access to transmit. When a device (device ‘A’) detects a signal sent by another device (device ‘B’) while A is transmitting, A aborts its transmission and retries after a random period of time.

D

DHCP (*dynamic host configuration protocol*) An extension of the BOOTP communications protocol that provides for the automatic assignment of reusable IP addressing settings—including IP address, subnet mask, gateway IP address, and DNS server names. DHCP does not require the maintenance of a table identifying each network device. All IP addresses assignments are made by temporary lease. The client identifies itself to the DHCP server using either its MAC address, or a uniquely assigned device identifier. The DHCP service utilizes UDP ports 67 and 68.

DNS (*domain name server/service*) A service that translates an alpha-numeric domain name into an IP address, the unique identifier of a device on the network.

domain name An alpha-numeric string that identifies one or more devices on the internet, and which appears as the primary component of a web site's Uniform Resource Locator (URL). For example, the domain name "schneider-electric.com" is the primary component of the URL "www.schneider-electric.com".
Each domain name is assigned as part of the Domain Name System, and is associated with an IP address.
Also called a host name.

E

Ethernet A 10 or 100 Mb/s, CSMA/CD, frame-based LAN that typically runs over twisted pair or fiber optic cable. The IEEE standard 802.3 defines the rules for configuring an Ethernet network.

EtherNet/IP *Ethernet Industrial Protocol:* A network communication protocol for industrial automation applications that combines the standard internet transmission protocols of TCP/IP and UDP with the application layer Common Industrial Protocol (CIP) to support both high speed data exchange and industrial control. EtherNet/IP employs electronic data sheets (EDS) to classify each network device and its functionality. Because EtherNet/IP is based on standard Ethernet protocols, it can be implemented using commercially available Ethernet components and cabling.

explicit messaging EtherNet/IP TCP/IP-based class-3 type messaging. Explicit messaging can be either connection-based or connectionless. It is used for point-to-point, client/server messages that include both data—typically unscheduled information between a client and a server—and routing information.

F

full duplex The ability of a two networked devices to independently and simultaneously communicate with each other in both directions.

G

gateway A device that interconnects two different networks—sometimes with different network protocols. When used to connect networks based on different protocols, a gateway converts one protocol stack into the other. When used to connect two IP-based networks, a gateway has two separate IP addresses - one on each network.

H

host name A domain name.

hub A multiport OSI-layer 1 (physical layer) device used to isolate network faults and span longer network distances by connecting several Ethernet devices with shielded/unshielded twisted pair or fiber optic cables. Messages received by a hub are regenerated and repeated on all ports. All connected devices are part of the same segment, share bandwidth and operate via half-duplex communication. Because a hub lacks the ability to filter network messages based on their source and destination address, the likelihood of collisions is increased. Collisions are handled by each connected device using CSMA/CD.

I

I/O messaging Used interchangeably with the term Implicit Messaging.

implicit messaging EtherNet/IP UDP/IP-based class 1 connected messaging. Implicit messaging maintains an open connection for the scheduled transfer of control data between a producer and consumer. Because an open connection is maintained, each message contains primarily data—typically I/O messaging data—plus a connection identifier.

IP address The 32-bit identifier—consisting of both a network address and a host address—assigned to a device connected to a TCP/IP Internet network using the Internet Protocol (IP).

L

local slave Some EtherNet/IP communication modules can simultaneously operate both as an I/O scanner and as an I/O adapter. In this dual-capacity, the role of the EtherNet/IP communication module as an I/O adapter is called a local slave.

M

message client (*explicit message client*) An explicit message client initiates request/response oriented communication with other devices. Message rates and latency requirements are typically not too demanding. Examples of explicit messaging clients are HMI devices, programming tools, or applications that gather data from control devices.

message server (*explicit message server*) An explicit message server responds to request/response oriented communications initiated by explicit message clients.

multicast A message sent out to multiple devices on the network by a host. A special form of broadcast where copies of the packet are delivered to only a subset of all possible destinations.

P

producer/ consumer A network communication model that provides increased control over device transmissions, resulting in reduced network traffic and more timely communication. Individual network devices (producers) can be configured to simultaneously send data:

- to recipient (consumer) devices, as follows:
  - only one specified network device (peer-to-peer)
  - a defined group of network devices (multicast), or
  - all devices on the network (broadcast)
- upon the occurrence of:
  - a specific event, or
  - the expiration of a time period uniquely configured for the producer device

R

rack optimized connection

Data from multiple IO modules are consolidated in a single data packet.

RPI

requested packet interval The time period between cyclic data transmissions made by the producer device in a producer/consumer communications model.

S

scanner

(I/O scanner) An I/O scanner originates I/O data connection requests to I/O adapters as well as to other I/O scanners (i.e. peer-to-peer explicit or I/O data). An I/O scanner may also be the originator or target of explicit connection requests to and from other devices, and they can also send or receive explicit messages to or from other devices.

subnet mask

The 32-bit value used to hide (or mask) the network portion of the IP address and thereby reveal the host address of a device on a network using the IP protocol.

switch

A multiport OSI-layer 2 (data link layer) device used to segment the network and limit the likelihood of collisions. Packets are filtered or forwarded based upon their source and destination addresses. Switches are capable of full-duplex operation and provide full network bandwidth to each port. A switch can have different input/output speeds (for example, 10, 100 or 1000Mbps).

T

target

Destination for I/O connection or message requests. Can only respond to a request, cannot initiate an I/O connection or message.

TCP

(*transmission control protocol*) TCP is the transport layer protocol that supports connection-oriented communications, by establishing the connection necessary to transmit an ordered sequence of data over the same communication path.

TCP/IP

(*transmission control protocol/internet protocol*) A set of protocols developed by the U.S. Defense Department's Advanced Research Projects Agency (ARPA) during the early 1970s. Its intent was to develop ways to connect different kinds of networks and computers. TCP/IP does not have the functionality that OSI provides. TCP/IP is a transport and Internet working protocol—i.e., the de facto networking standard. It is commonly used over X.25 and Ethernet wiring and is viewed as one of the few protocols available that is able to offer a true migration path towards OSI.
TCP/IP is able to operate in most environments. TCP/IP operates at Layers 3 and 4 of the OSI model (Network and Transport respectively).
TCP and IP are the standard network protocols in UNIX environments. They are almost always implemented and used together.
TCP/IP is a connection-oriented protocol.

trap

A trap is an SNMP agent detected event that indicates either:
- a change has occurred in the status of the agent, or
- an unauthorized SNMP manager device has attempted to get data from, or change data on, the SNMP agent

U

UDP

(*user datagram protocol*) UDP is a transport layer protocol that supports connectionless communications. Applications running on networked nodes can use UDP to send datagrams to one another. Unlike TCP, UDP does not guarantee reliable delivery or ordering of datagrams. However, by avoiding the overhead required by guaranteed delivery and checking of datagrams, UDP is faster than TCP. UDP may be the preferred protocol for time-sensitive applications, where dropped datagrams are preferable to delayed datagrams.

unconnected messaging	Unconnected messaging uses TCP (without a CIP connection) to send explicit messages. Unconnected messaging is also used in the process of establishing a CIP connection for connected messaging. More overhead is contained within each unconnected message than for a connected message. The unconnected message is not guaranteed destination node resources. Unconnected Messaging is used for non-periodic requests (i.e. network "Who" function).

Index

Numerics

140 NOC 771 00
    illustration, 12
    LED descriptions, 190
    LED indicators, 190
1734-AENT
    configuring, 117
    viewing I/O addresses, 121

A

adapter diagnostic object, 168
Advantys STB island
    connecting to, 92
assembly object, 172

B

BOOTP, 50

C

channel properties
    Ethernet, 39
    EtherNet/IP, 40
    general, 37
    module information, 42
CIP objects, 167
configuration
    EtherNet/IP configuration tool, 49
connection manager object, 173
connections
    CIP, 134
    TCP, 133

D

detect network devices, 87, 116
device library, 72
devices window, 33
DHCP client, 55
DHCP server, 54
diagnostic test, 192
diagnostics
    ping, 194

E

EDS file
    add, 74, 114
ethernet link object, 175
explicit message, 131
explicit messaging, 144
    error codes, 160
    Get_Attributes_Single, 148, 162
    Reset, 154, 164
    services, 142

F

full duplex, 127

I

identity object, 178
IGMP snooping, 128
implicit message, 132
IP address, 50

L

load
    example, 138
    limits, 134
local slave
    I/O, 59
    identifying, 58

M

MBP_MSTR, 148, 154
    error codes, 160
    explicit messaging, 144
module addresses
    EtherNet/IP configuration tool, 49
module diagnostic object, 180
multicast messaging, 128

N

network example, 85
    extended, 113

O

output messages, 195

P

ping, 194
port mirroring, 128
project file
    save, 82

R

remote device
    configuring, 78
replacement, 197

S

scanner diagnostic object, 182
SNMP agent, 52, 129
STB NIC 2212
    configuring adapter, 88
    configuring I/O items, 97
switch
    managed, 127
    recommended features, 127

T

TCP/IP interface object, 186
TCP/IP properties, 50

U

Unity Pro
    explicit messaging, 144

V

VLAN, 129