GFK-2196

New In Stock!

~~GE Fanuc Manuals~~

http://www.pdfsupply.com/automation/ge-fanuc-manuals/rx3i-plc/GFK-2196

rx3i-plc

1-919-535-3180

DeviceNet Modules for Series 90-30 and PACSystems RX3i
User’s Manual

www.pdfsupply.com

Email: sales@pdfsupply.com

GE Fanuc Automation
Programmable Control Products

DeviceNet Modules

for PACSystems™ RX3i and Series 90™-30

User's Manual, GFK-2196A

January 2006

Warnings, Cautions, and Notes as Used in this Publication

Warning

Warning notices are used in this publication to emphasize that hazardous voltages, currents, temperatures, or other conditions that could cause personal injury exist in this equipment or may be associated with its use.

In situations where inattention could cause either personal injury or damage to equipment, a Warning notice is used.

Caution

Caution notices are used where equipment might be damaged if care is not taken.

Note

Notes merely call attention to information that is especially significant to understanding and operating the equipment.

This document is based on information available at the time of its publication. While efforts have been made to be accurate, the information contained herein does not purport to cover all details or variations in hardware or software, nor to provide for every possible contingency in connection with installation, operation, or maintenance. Features may be described herein which are not present in all hardware and software systems. GE Fanuc Automation assumes no obligation of notice to holders of this document with respect to changes subsequently made.

GE Fanuc Automation makes no representation or warranty, expressed, implied, or statutory with respect to, and assumes no responsibility for the accuracy, completeness, sufficiency, or usefulness of the information contained herein. No warranties of merchantability or fitness for purpose shall apply.

The following are trademarks of GE Fanuc Automation, Inc.

Alarm Master	Genius	ProLoop	Series Six
CIMPLICITY	Helpmate	PROMACRO	Series Three
CIMPLICITY 90–ADS	Logicmaster	PowerMotion	VersaMax
CIMSTAR	Modelmaster	PowerTRAC	VersaPoint
Field Control	Motion Mate	Series 90	VersaPro
GEnet	PACSystems	Series Five	VuMaster
	Proficy	Series One	Workmaster

©Copyright 2002—2006 GE Fanuc Automation North America, Inc.
All Rights Reserved

Contents

Introduction 1-1

Finding Information in this Book 1-1
For Detailed Information about DeviceNet 1-1
DeviceNet Modules 1-2
DeviceNet Master Module 1-3
Features 1-3
DeviceNet Master Module Specifications 1-4
Compatibility 1-4
DeviceNet Slave Module 1-5
Features 1-5
DeviceNet Slave Module Specifications 1-6
Compatibility 1-6
The DeviceNet Network 1-7
DeviceNet Communications for DeviceNet Modules 1-8
I/O Messaging 1-8
Explicit Messaging 1-8

Installation 2-1

Power Supplies 2-2
PLC Power 2-2
DeviceNet Power 2-2
DeviceNet Cable for the Modules 2-3
Cable and Network Specifications 2-3
Bus Connector Pin Assignments 2-3
Bus Length 2-4
Network Termination 2-5
Taps, Daisy-Chaining and Branches 2-5
Grounding 2-6
DeviceNet Cable Grounding 2-6
DeviceNet Power Supply Grounding 2-6
DeviceNet System Grounding 2-6
Installing the DeviceNet Module in the PLC Rack 2-7
Removing the Module from the Rack 2-7
Module LEDs 2-8
RS-232 Serial Port 2-9
Upgrading the DeviceNet Module's Firmware 2-9

PLC Configuration for the DeviceNet Master 3-1

Configuration Steps 3-2
Adding a DeviceNet Master Module to the PLC Configuration 3-3
Configuring the Parameters of a DeviceNet Master Module 3-4
Configuring Network Settings of a DeviceNet Master Module 3-7
Adding Slaves to the Network 3-8

Configuring Network Settings for Slaves Added to the Master 3-9
Assigning the MAC IDs and Baud Rate 3-9
Configuring I/O Messaging Connections for Slaves added to the Master 3-10
Configuring DeviceNet Explicit Messaging 3-15
Configuring Network Settings for a DeviceNet Master Acting as a Slave 3-16
Configuring I/O Messaging Connections for a DeviceNet Master Acting as a Slave 3-16
Configuring DeviceNet Explicit Messaging 3-21

PLC Configuration for the DeviceNet Slave 4-1

Configuration Steps 4-2
Adding a DeviceNet Slave Module to the PLC Configuration 4-3
Configuring the Parameters of a DeviceNet Slave Module 4-4
Configuring the Network Settings of a DeviceNet Slave Module 4-7
Configuring I/O Messaging Connections 4-8
Configuring DeviceNet Explicit Messaging 4-13

Module Operation 5-1

Operation of a DeviceNet Master Module 5-2
DeviceNet Master Module Operation while the PLC is in Stop Mode 5-2
DeviceNet Master Module Operation while the PLC is in Run Mode 5-2
Operation of a Slave (Server) DeviceNet Module 5-4
Fault Table Entries for a DeviceNet Module 5-5
Faults Reported to the I/O Fault Table 5-5
PLC Status References for a DeviceNet Module 5-6
Device Status Bits for the DeviceNet Master Module 5-7

Programmed Communications 6-1

COMMREQs for the DeviceNet Modules 6-2
Using COMMREQs to Program Communications 6-3
Format of the Communication Request Function 6-4
Parameters of the COMMREQ Function 6-4
COMMREQ Programming Requirements and Recommendations 6-5
Checking the Execution of the COMMREQ 6-5
Using COMMREQs for DeviceNet Explicit Messaging 6-6
The COMMREQ Status Block for DeviceNet Modules 6-7
Command Code 9: Reading Identification, Status, and Error Information 6-8
Read Module Header, COMMREQ Example 6-8
Read Module Header, Reply Data Format 6-9
Command Code 4: Getting the Status of a Network Device 6-12
Command Code 5: Getting Status Information of a DeviceNet Slave Module or the Server Function of a Master Module 6-14
Command Code 6: Getting Input Status from a Device 6-16
Command Codes 1 & 7: Sending a DeviceNet Explicit Message on the Network 6-18
Send Device Explicit Extended, Data Block Format 6-22

Send Device Explicit (& Extended), Reply Data Format 6-23
Command Codes 2, 3 & 8: Reading and Responding to Client Explicit Messages 6-24
Reading a DeviceNet Custom-Explicit Message from the DeviceNet Module 6-26
Command Codes 3 & 8: Replying to a DeviceNet Explicit Message 6-28

DeviceNet Objects 7-1

Identity Object 7-2
Class Attributes 7-2
Class Services 7-2
Instance Attributes 7-2
Instance Services 7-2
Message Router Object 7-3
Class Attributes 7-3
Class Services 7-3
Instance Attributes 7-3
Instance Services 7-3
DeviceNet Object 7-4
Class Attributes 7-4
Class Services 7-4
Instance Attributes 7-4
Instance Services 7-4
Assembly Object 7-5
Class Attributes 7-5
Class Services 7-5
Instance Attributes 7-5
Instance Services 7-5
Connection Object 7-6
Class Attributes 7-6
Class Services 7-6
Instance Attributes 7-7
Instance Services 7-7
PLC Data Object 7-8
Specific Services 7-8
Specific Instances 7-8
GET_PLC_DATA Service (0x32) 7-8
SET_PLC_DATA Service (0x33) 7-9

DeviceNet EDS Files A-1

Electronic Datasheet File for the DeviceNet Master Module A-1
Electronic Datasheet File for the DeviceNet Slave Module A-2

Chapter 1 Introduction

This manual describes the features and operation of the DeviceNet Master and Slave modules when they are used in a PACSystems RX3i and in a Series 90-30 system.

Finding Information in this Book

Chapter 1: Introduction, provides basic information about the DeviceNet Master Module (IC693DNM200/IC694DNM200) and the DeviceNet Slave Module (IC693DNS201).

Chapter 2: Installation, discusses power requirements, cable specifications, grounding, module installation, and LED indications.

Chapter 3: PLC Configuration for the DeviceNet Master, explains how to add a DeviceNet Master Module to the configuration of the control system. It also explains how to configure communications connections, the module, and the DeviceNet network.

Chapter 4: PLC Configuration for the DeviceNet Slave, explains how to add a DeviceNet Slave Module to the configuration of the control system.

Chapter 5: Module Operation, describes how the DeviceNet Master Module and the DeviceNet Slave Module function in a PACSystems RX3i or a Series 90-30 PLC system.

Chapter 6: Programmed Communications (COMMREQs), explains how the application program can communicate with the module for Explicit Messaging and for reading status information.

Chapter 7: DeviceNet Objects, describes the information objects that are defined for the DeviceNet modules.

Appendix A: DeviceNet EDS Files, contains the Electronic Datasheet (EDS) Files that are defined for the DeviceNet Master Module and the DeviceNet Slave Module.

For Detailed Information about DeviceNet

For detailed information about DeviceNet, contact the Open DeviceNet Vendor Association.

Open DeviceNet Vendor Association, Inc.
20423 State Road 7
Suite 499
Boca Raton, FL. 33498
phone: (954) 340-5412
FAX: (954) 340-5413
Internet: HTTP://WWW.ODVA.ORG
Email: <ODVA@POWERINTERNET.COM

DeviceNet Modules

Two different DeviceNet modules bring the flexibility of DeviceNet communications to a PACSystems RX3i or a Series 90-30 PLC:

- The DeviceNet Master Module (IC693DNM200/IC694DNM200) operates as the network master. It exchanges I/O messages and custom explicit messages with up to 63 other devices on the network. On DeviceNet networks that include a higher-level host computer, this module's built-in server function can be used for both automatic data transfer and custom explicit messaging with the master.
- The DeviceNet Slave Module (IC693DNS201) operates as a network server (slave) only. It can automatically exchange PLC data with a network master, and respond to custom explicit messages from the master.

In an RX3i system, a DeviceNet Master Module can be installed in the RX3i main rack or in a Series 90-30 expansion rack. In a Series 90-30 PLC the Master Module can only be installed in the main rack.

A DeviceNet Slave Module can be installed in the main rack in a Series 90-30 PLC.

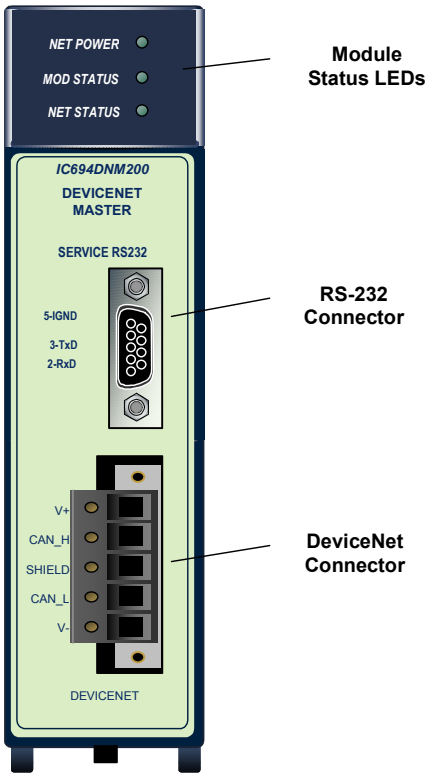
Each module counts as a single node on the DeviceNet network. However, the server function of each module is easily configured for up to two DeviceNet I/O Messaging connections and for Explicit Messaging.

DeviceNet Master Module

The DeviceNet Master Module allows the control system CPU to send and receive data over a DeviceNet network. It can act as master for up to 63 slaves on the DeviceNet network. It can also be configured to simultaneously function as a slave to another master on the bus.

The module's three DeviceNet-compliant LEDs show its operating and communications status. The RS-232 serial port (a 9-pin male D-connector) is used for a computer connection during firmware upgrades. The DeviceNet connector is a removable spring-clamp terminal. It provides bus continuity and can be removed from the module without disrupting bus operation.

The RX3i and Series 90-30 DeviceNet Master Modules are identical except for their plastic housing.



PACSystems RX3i
DeviceNet Master Module

Features

- Bus communications at all standard DeviceNet data rates (125k, 250k, 500k baud)
- Up to 255 bytes input data transfer and 255 bytes output data transfer per slave and up to 3972 bytes of input data transfer and 3972 bytes of output data transfer per master.
- Unconnected Message Manager (UCMM) with 1 proxy connection per slave
- One or two I/O connections plus explicit messaging can be configured for each slave. Each slave I/O connection can be set up for one of the following: Poll, Strobe, Cyclic or Change-of-State (COS) operation. Typically one connection is used for Polled and the other is used for Strobe, Cyclic, or COS.
- Independent configuration of update rates for Poll and COS/Cyclic I/O devices
- Configurable global scan rate
- PLC-application initiated Explicit messaging using COMMREQs
- Status of communication with slaves available in the PLC fault table (configurable). Provides 64 network device status bits
- Configurable fault behavior on loss of communication

DeviceNet Master Module Specifications

Catalog Numbers	IC693DNM200, IC694DNM200
Description	Master Module for DeviceNet networks
Mounting Location Series 90-30 PACSystems RX3i	 Series 90-30 main rack. Any slot except slots 0 and 1. RX3i main rack. Any slot except slot 0. Series 90-30 expansion rack. Any slot except slot 0.
Environment	Storage temperature: -40°C to 85°C Operating temperature: 0°C to 60°C
Backplane Current Consumption	450mA at 5VDC (typical)
Current Consumption from DeviceNet Network	50mA at 24VDC (typical) 80mA at 24VDC (peak)
Data rates	Supports all standard DeviceNet data rates (125K, 250K, and 500K Baud)

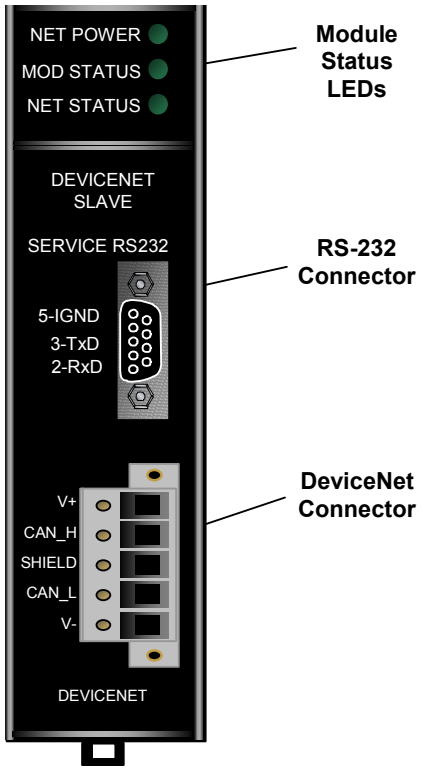
Compatibility

- Compatible with PACSystems™ RX3i CPUs and NIUs. Requires PACSystems firmware release 3.5 or later.
- Compatible with any Series 90-30 CPU except IC693CPU321 and IC693CPU340. Configuration size is limited for CPU311/313/331, as described in chapter 3. Requires Series 90-30 release 8.0 or later CPU firmware. Release 10.6 or later is recommended, if available for the particular CPU.
- For Series 90-30 CPUs, Proficy™ Machine Edition Logic Developer PLC version 3.0 SP1 Special 2 or later is required.
- For PACSystems RX3i, Proficy™ Machine Edition Logic Developer PLC version 5.5 or later is required.
- Not compatible with the VersaPro™, Control, or Logicmaster™ programming software.
- The Series 90-30 Hand-Held Programmer (IC693PRG300) cannot be used to configure this module.

DeviceNet Slave Module

The DeviceNet Slave Module interfaces a Series 90-30 PLC to a DeviceNet bus that is controlled by another master device.

The module's three DeviceNet-compliant LEDs show its operating and communications status. The RS-232 serial port (a 9-pin male D-connector) is used for a computer connection during firmware upgrades. The DeviceNet connector is a removable spring-clamp terminal. It provides bus continuity and can be removed from the module without disrupting bus operation.



Series 90-30 DeviceNet Slave Module

Features

- Bus communications at all standard DeviceNet data rates (125K, 250K, 500K baud)
- Up to 255 bytes input data transfer and 255 bytes output data transfer.
- Configurable for Poll, Strobe, Cyclic and COS I/O Connections, and Explicit Messaging
- Supports Unconnected Message Manager (UCMM) allowing up to 250 simultaneous explicit messaging connections.
- One or two I/O connections plus explicit messaging can be configured. Each I/O connection can be set up for one of the following: Poll, Strobe, Cyclic or Change-of-State (COS) operation. Typically one connection is used for Polled and the other is used for Strobe, Cyclic, or COS.
- Supports the Assembly Object and access to the input and output data for each of the configured slave I/O areas (I/O Area 1 and I/O Area 2) with the SET_ATTRIBUTE_SINGLE and GET_ATTRIBUTE_SINGLE services. Up to 255 bytes of attribute data may be supplied in the SET_ATTRIBUTE_SINGLE operation.
- Configurable fault behavior on loss of communication.
- UCMM-capable Group 2 Server

DeviceNet Slave Module Specifications

Catalog Number	IC693DNS201
Description	Slave Module for DeviceNet networks
Mounting Location	Series 90-30 main rack. Any slot except slots 0 and 1.
Environment	Storage temperature: -40°C to 85°C Operating temperature: 0°C to 60°C
Backplane Current Consumption	450mA at 5VDC (typical)
Current Consumption from DeviceNet Network	50mA at 24VDC (typical) 80mA at 24VDC (peak)

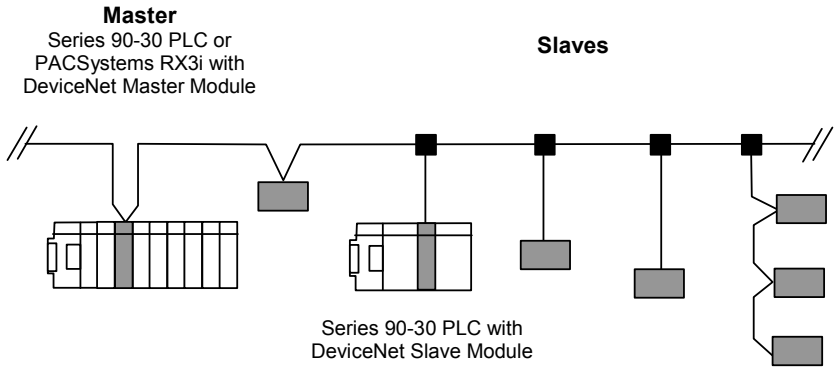
Compatibility

- Compatible with any Series 90-30 CPU except IC693CPU321 and IC693CPU340. Requires Series 90-30 release 8.0 or later CPU firmware. Release 10.6 or later is recommended, if available for the particular CPU.
- Requires Proficy™ Machine Edition Logic Developer PLC version 3.0 SP1 Special 2 or later.
- Not compatible with the VersaPro™, Control, or Logicmaster™ programming software.
- The Series 90-30 Hand-Held Programmer (IC693PRG300) cannot be used to configure this module.

The DeviceNet Network

DeviceNet is a communications network that transmits data between control systems (for example: PLCs, PCs, VMEbus computers, and robot controllers) and distributed industrial devices such as switches, sensors, valve manifolds, motor starters, bar code readers, drives, displays, and operator interfaces. The network can also link intelligent interface modules such as the VersaPoint DeviceNet Network Interface Unit (NIU) and the VersaMax DeviceNet NIU. An NIU is the network interface for an I/O Station of many additional modules.

The DeviceNet network has a linear structure. There is a main trunk line with drop lines routed to the networked devices. Power and signals are carried on the same network cable. A Series 90-30 DeviceNet module can be connected directly to the trunk cable or installed as an individual drop or as part of a daisy-chain drop.



DeviceNet Communications for DeviceNet Modules

DeviceNet uses the Producer-Consumer technique of messaging. A device with data produces the data on the network. All devices that need data listen for messages. When a device recognizes the appropriate identifier, it consumes the data. A message is not specific to a particular source or destination, and one message can be consumed by multiple devices. For example, one message could control several motor starter modules.

A DeviceNet message field can range between 0 and 8 bytes. Messages longer than 8 bytes are fragmented into packets. The DeviceNet modules assure data integrity for each network node.

I/O Messaging

I/O messaging is used for the routine and automatic exchange of data between devices. Individual I/O messages can be up to 255 bytes in length. I/O messages provide a dedicated communication path between a producing device and one or more consuming devices. System configuration sets up the parameters for the connections between the producing and consuming devices. With the connections established, communications occur automatically.

There are four basic types of I/O messages: Polled, Strobed, Cyclic, and Change-of-State (COS). The DeviceNet Master Module can be configured for up to two different types of I/O messaging connections to each slave (for example, one Cyclic I/O messaging connection and one Change-of-State I/O messaging connection). See chapter 3, "PLC Configuration for the DeviceMet Master" for details.

Explicit Messaging

Explicit messaging provide a point-to-point communication link between two devices on the network. Explicit messaging is often used for slave configuration and for diagnostics. Specific Explicit messages are defined for the DeviceNet protocol. For the DeviceNet Master module, Communications Request (COMMREQs) are used to send Explicit messages. See chapter 6, "Programmed Communications" for more information. Received Explicit messages are automatically processed by the DeviceNet module except for user defined Objects which require user programmed COMMREQs support.

Chapter 2 Installation

This chapter provides basic installation information for the DeviceNet Modules:

- Power requirements: PLC system power and DeviceNet network power
- The DeviceNet cable: specifications, length, termination, taps
- Grounding: DeviceNet cable, DeviceNet power supply, DeviceNet system
- Installing the DeviceNet module in the PLC
- The Module LEDs: Module Status, Network Status, Network Power
- The RS-232 Serial Port

For Additional Information

Correct installation of cables, power supplies, and other network hardware requires a more detailed knowledge of DeviceNet specifications than can be provided here. Readers are referred to www.ODVA.org for additional information.

Conformance to Standards

Before installing GE Fanuc products in situations where compliance to standards or directives from the Federal Communications Commission, the Canadian Department of Communications, or the European Union is necessary please refer to the following GE Fanuc publications:

Series 90-30:	*Installation Requirements for Conformance to Standards*, GFK-1179
PACSystems RX3i	*PACSystems RX3i System Manual,* GFK-2314

Power Supplies

When using a DeviceNet module, there are two separate power supplies to consider: the PLC power supply and the DeviceNet network power supply.

PLC Power

A DeviceNet module consumes 450mA at 5VDC (typical) from the PLC backplane. In a Series 90-30 system, a high-capacity power supply such as IC693PWR330 or IC693PWR331 is recommended when using these modules, especially for CPU models CPU350 or higher, or if the PLC includes Ethernet adapters and/or multiple DeviceNet modules.

The PLC power supply load is automatically calculated by the Machine Edition configuration software. Additional information about estimating power supply load can be found in the *Series 90-30 Installation and Hardware Manual*, GFK-0356 and the *PACSystems RX3i System Manual,* GFK-2314.

DeviceNet Power

The DeviceNet modules power their network transceivers from the 24VDC DeviceNet network power source. Linear power supplies are recommended for the DeviceNet power source. The DeviceNet power source should *not* also be used for device power. Transients caused by I/O devices can cause communications errors and even create bus-off conditions.

The DeviceNet specification recommends using a power tap to connect a power supply to the network. The power tap should be appropriately fused for the current capacity of the bus cables. The maximum current on the network depends on the cable type.

The DeviceNet modules consume 50mA typical (80mA peak) at 24VDC from the DeviceNet Network.

Note: The DeviceNet Master module fails during a second store of HWC when there is no network power.

Users should make every effort to ensure that DeviceNet network power is available to the DNM200 when:

- storing Hardware Configuration,
- power cycling the PLC with configuration available in retentive memory,
- reading Hardware Configuration from Flash/EEPROM.

If the module fails because of this issue, restore network power and power cycle the rack that contains the DNM200.

Current Limit for Thick Cable

For thick cable, the maximum current on the network is 16 Amps. However, only 8 Amps is permitted on a single network segment. 16 Amps can be drawn from a single power supply by locating the power supply at the center point of two network segments, supplying 8 Amps to each segment.

Current Limit for Thin Cable

For thin cable, the maximum current permitted is 3 Amps.

DeviceNet Cable for the Modules

DeviceNet modules can be used with the either DeviceNet thick cable or thin cable. Thick cable permits greater cable lengths and higher current levels. Generally, thick cable is used for the trunk cable. Thin cable is normally used for shorter distances and is suitable for drop cables and for installations where more cable flexibility is needed.

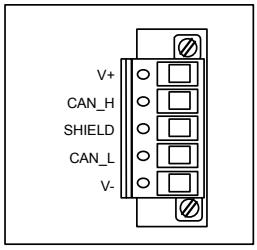
Both thick cable and thin cable are 5-wire, multi-conductor copper cable. Two wires form a transmission line for network communications. A second pair transmits network power. The fifth conductor forms an electromagnetic shield. Most cables have color coded leads which correspond to the color coding on the terminals on the DeviceNet modules.

Cable and Network Specifications

Thick Cable General Specifications	Two shielded pairs - Common axis with drain wire in center
	Overall braid shield - 65% coverage; 36 AWG or 0.12mm tinned Cu braid minimum (individually tinned)
	Drain wire- #18 Copper min.; 19 strands minimum (individually tinned)
	Outside diameter - 0.410 inches (min) to 0.490 inches (max.) roundness - radius delta to be within 15% of 0.5 O.D.
Thin Cable General Specifications	Two shielded pairs - Common axis with drain wire in center
	Overall braid shield - 65% coverage; 36 AWG or 0.12mm tinned Cu braid minimum (individually tinned)
	Drain wire - #22 Copper; 19 strands minimum (individually tinned)
	Outside diameter - 0.240 inches (min.) to 0.280 inches (max.) roundness - radius delta to be within 20% of 0.5 O.D.
Network Topology	Bus with limited branching (trunkline/dropline)
Redundancy	Not Supported
Network Power for Node devices	Nominal 24 VDC ±4%
Allowed Nodes (Bridging excluded)	64 nodes
Data Packet Size	0-8 bytes with allowance for message fragmentation
Duplicate Address Detection	Addresses verified at power-up
Error Detection / Correction	CRC - retransmission of message if validity not acknowledged by recipient

Bus Connector Pin Assignments

The DeviceNet connector on a DeviceNet module has five color-coded screw-clamp terminals. The connector provides bus continuity; it can be removed from the module without disrupting bus operation.



Signal	*Pin*	*Wire Color*
V+	5	Red
CAN_H	4	White
Shield	3	Bare
CAN_L	2	Blue
V-	1	Black

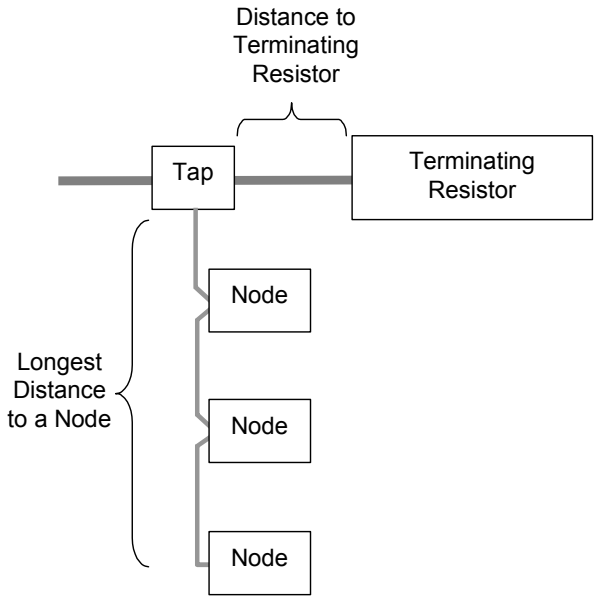
Bus Length

The maximum length of the trunk cable and drops both depend on the cable type and data rate. Individual drops may not exceed 6 meters and are limited to one network node per drop. However, the node may have multiple ports.

Data Rates	***125kbps***	***250kbps***	***500kbps***
thick cable, trunk length	500m (1640ft)	250m (820ft)	100m (328ft)
thin cable, trunk length	100m (328ft)	100m (328ft)	100m (328ft)
maximum drop length	6m (20ft)	6m (20ft)	6m (20ft)
total length of all drops	156m (512ft)	78m (256ft)	39m (128ft)

For each baud rate, the total drop length is the sum of all the drop lines of both cable types in the network.

In addition, if the distance from a tap to the most distant device on its drop is longer than the distance from the tap to the nearest terminating resistor as illustrated below, the drop line length also counts as part of the trunk cable length (as well as the overall drop length).

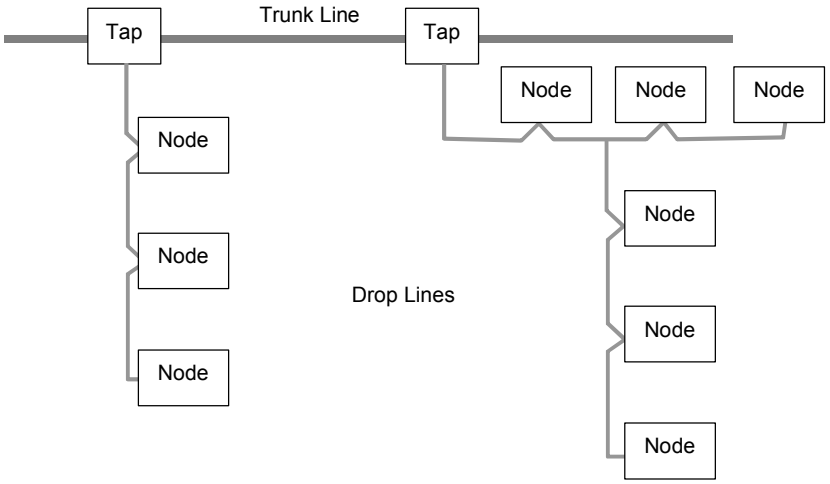


Network Termination

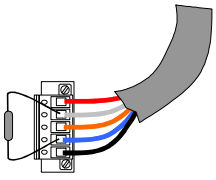
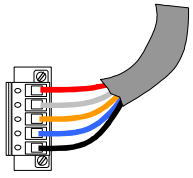
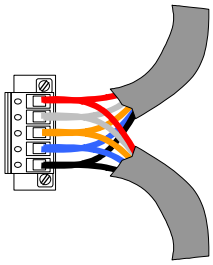
121 Ohm, 1% ¼ watt terminating resistors MUST be installed at both ends of the DeviceNet network. Each terminating resistor is placed across the data communication signals at pin 2 (CAN_L) and pin 4 (CAN_H).

Taps, Daisy-Chaining and Branches

Devices can be connected directly to the trunk cable, or to drop lines that are joined to the trunk cable with taps. Taps can be mounted in junction boxes or panels. Drop lines and daisy-chains are often used inside control panels where multiple devices are grouped together. When using drops with daisy-chains and branches, the maximum length from a tap to its farthest drop is 20 feet.



Wiring to the DeviceNet Master module depends on its location on the network:

If the DeviceNet module is located at either end of the bus trunk, it is wired with one cable connection and a terminating resistor:	If the module is installed at the end of a drop or drop segment, it is wired with one cable connection only.	If the module is installed directly on the trunk cable or as part of a daisy-chained drop cable, it has both an incoming and outgoing cable connected:
		

Grounding

DeviceNet Cable Grounding

All DeviceNet cable shields must be tied to ground at each device connection. This is done by tying the bare wire of the cable to pin 3 (Shield) of the connector.

DeviceNet Power Supply Grounding

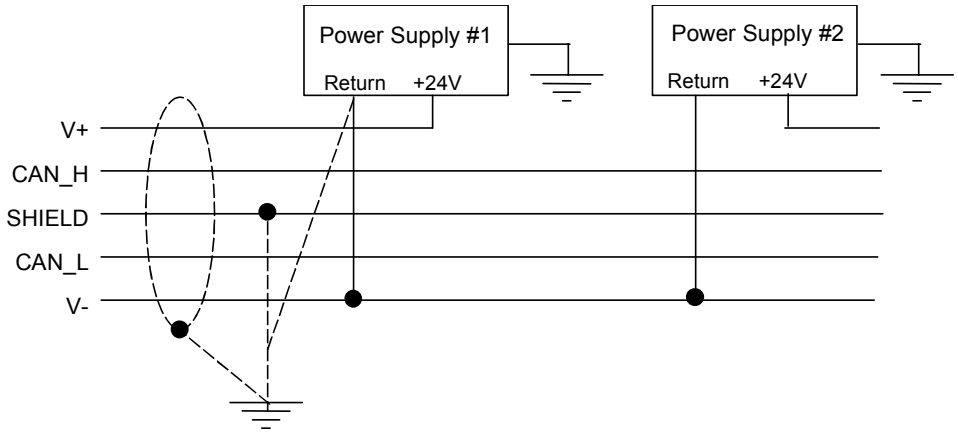
The DeviceNet network power supply must also be grounded, but only at one point. The V- signal must be connected to protective earth ground at the power supply only. If multiple power supplies are used, only one power supply must have V- connected to earth ground.

DeviceNet System Grounding

DeviceNet communications should only be grounded to earth at a single point. Typically this is done in the control cabinet where the DeviceNet power supply is located.

Return for the DeviceNet power (-V), the drain (bare wire) and the cable shields must be directly tied to earth ground. Ideally, this grounding is done at a central location. Connection should be made using a 25mm (1in.) copper braid or a #8 AWG wire not longer than 3meters (10ft.).

The illustration below represents grounding for a network that has two power supplies. The chassis of each power supply is connected to earth ground.



Ground Wire Size

The minimum size ground conductor for the DeviceNet screw-clamp terminals on a DeviceNet module is a 2.5mm^2 (14 AWG) wire. For other network devices, larger wire diameters may be necessary.

Installing the DeviceNet Module in the PLC Rack

In a PACSystems RX3i system, a DeviceNet Master module can be installed in slot 1 or higher of the main rack or any Series 90-30 expansion rack. The DeviceNet Master cannot be configured in a Series 90-30 remote rack.

In a Series 90-30 system, a DeviceNet Master or Slave module can be installed in the main (CPU) rack in slot 2 or higher. The Series 90-30 system does not support DeviceNet modules in expansion or remote racks.

1. Turn off power to the rack.
2. Place the module into its slot by hooking the top of the module on the notch above the slot and slowly lowering the module until it snaps into place.
3. Attach the DeviceNet cable to the module.
4. Terminate the network as required.

Note: For details about installing PACSystems RX3i rack systems and modules, refer to the *PACSystems RX3i System Manual,* GFK-2314.

For details about installing Series 90-30 rack systems and modules, refer to the *Series 90-30 Installation Manual and Hardware Manual*, GFK-0356.

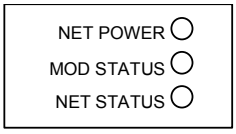
Removing the Module from the Rack

To remove the module from the rack:

1. Turn off power to rack.
2. Remove all cables from the module.
3. Press the release located on the bottom of the module and slowly raise the module from the bottom until it comes out of the slot.

Module LEDs

The module's three LEDs show its operating and communications status:



Network Power LED

LED	Indicates
Red	There is no power detected on the network.
Green	Power detected on the network.

Module Status LED

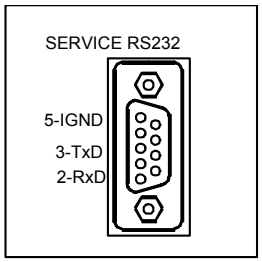
LED	Indicates
Off	There is no backplane power to the module.
Green	The module is operating normally.
Flashing Green	The module is in standby mode. Its configuration is missing, incomplete, or incorrect. The module may be in Standby state.
Flashing Red	Recoverable Fault
Red	The module has an unrecoverable fault; it may need resetting or replacing.
Flashing Red / Green	The module is in Self Test mode.

Network Status LED

LED	Indicates
Off	▪ The module is not online, or ▪ The module has not completed the Duplicate MACID test, or ▪ The module may not be powered. See Module Status LED.
Flashing Green	▪ The module is online but has no connections in the established state, or ▪ The module has passed the Duplicate MACID check, is online, but has no established connections with other nodes.
Green	The module is online and has one or more connections in the established state.
Flashing Red	One or more I/O Connections are in the Timed Out state.
Red	The module is not capable of communicating on the network.
Flashing Red / Green	The module has detected a Network Access error and is in the Communication Faulted State.

RS-232 Serial Port

The RS-232 serial port is a 9-pin male D-connector. This port is used for a computer connection during firmware upgrades.



Upgrading the DeviceNet Module's Firmware

When upgrades to the firmware are made available, the module can be upgraded as described below.

1. Download the firmware upgrade to the computer that will be used to perform the upgrade. Firmware upgrades are usually available from the GE Fanuc WEB site www.gefanuc.com/support/plc/
2. Connect a straight through serial cable from the computer to the serial port on the front of the DeviceNet module. Only the RX and TX lines are used. The cable that is part of the RS232 to RS485 kit (IC693ACC903) is suitable.
3. The computer can use any standard communication software to communicate with the module. The module supports 19200 baud, no parity, 8 data bits, 1 stop bit, and no flow control.
4. Cycle power to the rack that contains the DeviceNet module.
5. At the computer, press ENTER until you see the initial greeting. The greeting indicates that the boot code is waiting for new firmware. Initiate an Xmodem send of the module firmware file using your communication software. Note: You must press the ENTER key immediately after the power is cycled. It is recommended you hold down the ENTER key when turning the power back on.
6. When it finishes successfully storing the new firmware, the module automatically resets and attempts to start the new firmware.
7. If the firmware transfer was not successful, the greeting screen reappears. Retry the transfer.

On success, disconnect the serial cable and cycle power to the rack.

Note: The RX3i CPU will log an Unsupported Hardware fault against the DNM200 when upgrading the DNM200's firmware. This fault can be safely ignored.

Chapter 3 PLC Configuration for the DeviceNet Master

This chapter explains how to add a DeviceNet Master Module to the configuration of the PACSystems RX3i or the Series 90-30 PLC. It also explains how to configure communications connections between a DeviceNet Master Module and the DeviceNet network.

- Configuration Steps
- Adding a DeviceNet Master Module to the PLC Configuration
- Configuring the Parameters of a DeviceNet Master Module
  - Parameters of a DeviceNet Master Module
  - Network Settings of a DeviceNet Master Module
- Telling the DeviceNet Master about Slaves by Adding Slaves to the Network
  - Adding a Device's EDS File
- Configuring Network Settings for Slaves Added to the Master
  - Assigning the MAC IDs and Baud Rate
  - Configuring I/O Messaging Connections
  - Configuring DeviceNet Explicit Messaging
- Configuring Network Settings for a DeviceNet Master Acting as a Slave
  - Assigning the MAC IDs and Baud Rate
  - Configuring I/O Messaging Connections for a DeviceNet Master Acting as a Slave
  - Configuring DeviceNet Explicit Messaging

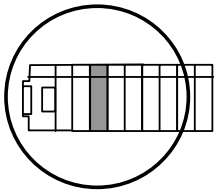
These configuration procedures are written for users who have a basic knowledge of the Machine Edition Logic Developer software and the Series 90-30 PLC or PACSystems RX3i control system. For help with using the software, please see the software's built-in help system.

Note: The DeviceNet Master is only supported in Machine Edition Logic Developer. The Logicmaster™, VersaPro™, and Control software do not support these DeviceNet modules.

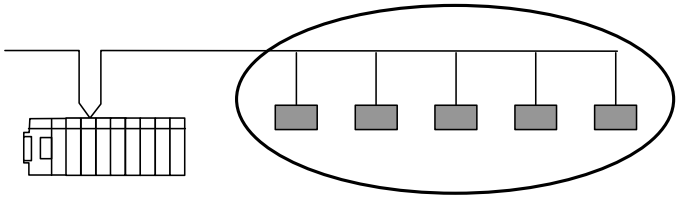
Configuration Steps

There are three basic steps to configuring a DeviceNet Master Module:

- Adding the module to the PLC rack and configuring its operating parameters.



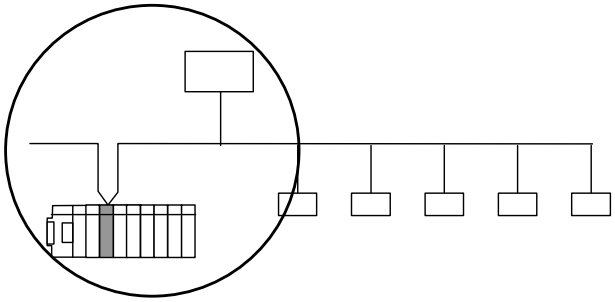
- Telling the DeviceNet Master about slaves by adding the network slaves to the Master and configuring their network settings.



The number and type of slave devices and the amount of data they can exchange with the master may be limited by the CPU memory available. The amount of CPU memory available for the DeviceNet configuration depends on: the CPU model being used, the version of the CPU firmware, the number and type of other modules in the configuration, the number and type of slave devices configured, and the amount and type of communication in progress with an external programmer or HMI devices. With Logic Developer-PLC, the size of the current configuration can be read by selecting "Data View" for the hardware configuration and adding the sizes of the components listed. LD-PLC will not allow configurations to be created that exceed 65,535 bytes. The size of the DeviceNet configuration is also limited by the size of the user configuration space for the models listed below:

CPU 311/313	4,736 bytes available
CPU 331	4,673 bytes available

- (Optional) Configuring the network settings of the DeviceNet Master Module itself. This is ONLY done if the DeviceNet Master Module will also operate as a server to another network master. For example, it might exchange data relating to the operation of its slaves with a higher-level host controller.

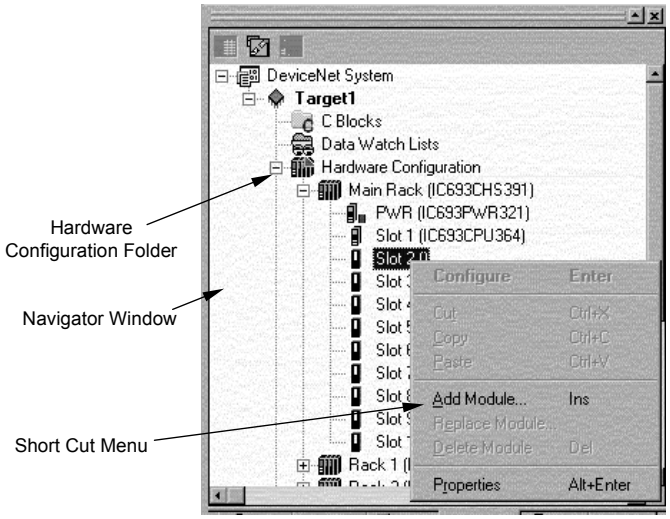


Adding a DeviceNet Master Module to the PLC Configuration

First, add the DeviceNet Master Module to the PLC rack configuration. The Master and Slave Modules are compatible with any Series 90-30 CPU except IC693CPU321 or IC693CPU340. Theyrequire release 8.0 CPU firmware as a minimum. Release 10.6 or later is recommended, if available for your particular CPU.

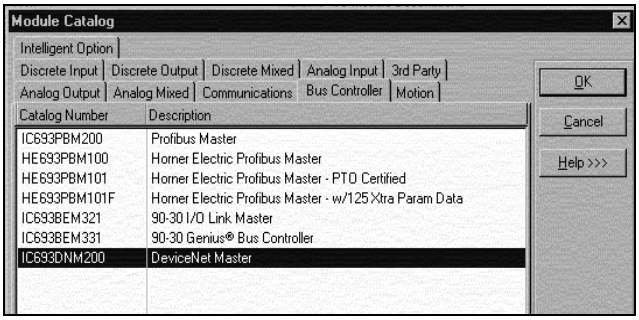
The Master Module is compatible with PACSystems RX3i CPU firmware version 3.50 or later.

1. In that configuration, in the Project tab of the Navigator, expand the Hardware Configuration folder.
2. In the Hardware Configuration folder, right click the intended PLC Slot for the DeviceNet Master Module. In an RX3i system, it can be slot 2 or higher in the main rack or slot 1 or higher in a Series 90-30 expansion rack. In a Series 90-30 system, the module can be in slot 2 or higher in the main rack.
3. Select Add Module from the shortcut menu.



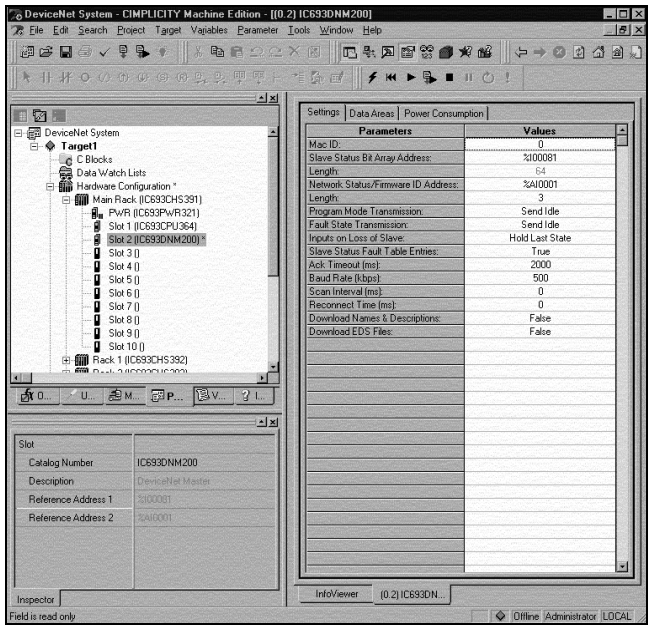
The Module Catalog dialog box appears.

4. To add a DeviceNet Master Module, click on the Bus Controller tab. The Bus Controller module list appears. Select IC693DNM200 DeviceNet Master from the list and click OK.



Configuring the Parameters of a DeviceNet Master Module

The DeviceNet Master Module is added to the PLC configuration in the Navigator window, and the module's Parameter Editor window appears in the InfoViewer window space.



Parameters of a DeviceNet Master Module

Settings tab	
Mac ID	The Mac ID (medium access control identification) of the master on the DeviceNet network. Valid range: 0 - 63. Default: 0.
Slave Status Bit Array Address	The starting address for an array of bits indicating the health of each node on the DeviceNet network. It must be a non-overlapping range in %AI, %I, %Q, %G, %AQ, %R, %T, or %M. It defaults to %I memory and uses the next available %I address. A slave's status address equals Start Address + Station Address of the slave. For example, if the status bits are mapped to %I00001, the status for the slave at Station Address 5 would be found at %I00001 + 5 = %I00006. The master's status is located in the same way as the slaves' (Start Address + Station Address). The master is configured as station 0 by default, but can be set to any valid address (0-64).
Length (of slave status bits)	(Read-only.) Length has a value of 64, corresponding to 64 network devices.
Network Status/Firmware ID Address	The starting address for three words of module/network status information. The default is %AI memory and uses the next available %AI address. The Network Status/Firmware ID Address must be a non-overlapping range in %AI, %I, %Q, %G, %AQ, %R, %T, or %M. During system operation, the module and network status data and module firmware ID will be stored in this memory location.
Length (of network status/firmware ID)	(read only) Length of the Network Status / Firmware ID Address memory location, three words.
Program Mode Transmission	When the PLC is in Program mode (Stop mode), the DeviceNet Master Module can either send idle packets or set data to zero. The default is to send idle packets.

Settings tab	
Fault State Transmission	When the DeviceNet Master Module detects a PLC fault (because the PLC has not requested its regular I/O update from the module), the module can either send idle packets or set data to zero. The default is to send idle packets.
Inputs Default (RX3i) **Inputs on Loss of Slave (Series 90-30)**	If the DeviceNet Master loses communications with a slave, it can hold the module's input reported to the CPU in its last state (default), or clear the input data.
Outputs Default (RX3i) **Outputs on Loss of Master (Series 90-30)**	If a slave loses communications with the DeviceNet master, it can retain the last output value it has successfully received from the master (default), or clear the output values from the master.
Fault Table Entries (RX3i) **Slave Status Fault Table Entries (Series 90-30)**	When slave communications status events (loss and re-establish) occur, the DeviceNet Master Module can either report them in the fault table or not. If this setting is True (the default), the Master makes fault table entries. If this setting is False, slave status events are not reported to the fault table.
Ack Timeout (ms)	Number of milliseconds to wait for a Controller Area Network (CAN) acknowledge of the Duplicate MacID check (performed during startup) before reporting an Ack (acknowledge) failure. Valid range: 0 to 65,535. The default is 2,000ms (2 seconds).
Baud Rate (kbps)	The data transmission rate for the DeviceNet Master Module. The maximum baud rate that can be used depends on the bus length and cable type. See chapter 2 for more information. Choose: 125K, 250K, or 500K.
Scan Interval (ms) for Strobed connections	The time interval between successive scans of Strobed slave connections. This defaults to zero. A time must be specified if any slave connections are set up for strobing. The valid range is 0 to 65,535ms. All strobed connections will be scanned at this same interval.
Reconnect Time (ms)	If a slave fails to respond to three consecutive scan cycles, the slave is flagged as not present and the master tries to reconnect to it. This parameter specifies how long the master should wait before attempting to reconnect. The default time is 100 for RX3i (0 for Series 90-30). The valid range is 0 or 100 to 65535ms.
Download Names & Descriptions **(Series 90-30 only)**	This setting determines whether or not names and descriptions that have been use in the configuration will be downloaded to the PLC when the configuration is downloaded. By default, this parameter is False and names and descriptions are not downloaded to the PLC. This is the recommended choice because downloaded names and descriptions can take up too much memory in the PLC. Names and descriptions are a convenience only. Omitting them from the download does not affect system operation. However, if this parameter is set to False, later uploads of the configuration from the PLC to the programmer will contain only default names and descriptions. If this parameter is set to True, names and descriptions that have been entered for the slaves and the master are downloaded to the PLC and will be present in the configuration if it is uploaded to the programmer later. Names and descriptions are always downloaded to PACSystems RX3i.

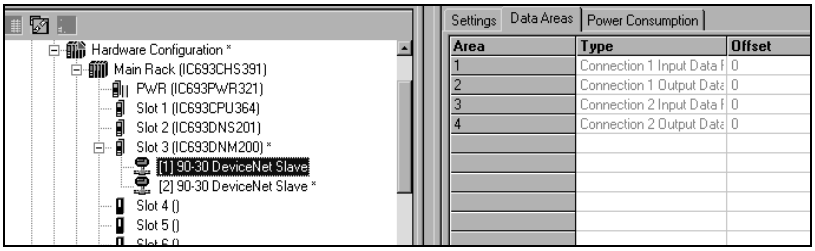
Settings tab	
Download EDS Files (Series 90-30 only)	This setting determines whether or not the EDS files that have been used for the configuration will be downloaded from the programmer to the PLC when the configuration is downloaded. By default, this parameter is False and EDS files are not downloaded to the PLC. This is the recommended choice, because downloaded EDS files can consume too much memory in the PLC. However, when this is set to False, if the configuration is later uploaded from the PLC back to the programmer, it will not contain the EDS files. You would need another source for the EDS files (for example, on disk) to configure more modules of a given type. If the EDS files are no longer available, it is only possible to add more modules of that type as generic modules. If this parameter is set to True, the EDS files are included when the configuration is downloaded to the PLC. If the configuration is later uploaded back to the programmer, the EDS files will be restored to the Toolchest for use by the configuration. EDS files are always downloaded and uploaded to a PACSystems RX3i.
I/O Scan Set (PACSystems RX3i only)	The scan set, as defined in the CPU's Scan Sets tab, to be assigned to this module. The default scan set is 1. The valid range is 1 through 32.

Data Areas Tab

This tab shows the PLC program references assigned to the DeviceNet Master Module's Network Settings *when it is used as a slave*:

This TAB is optional and should only be used when the DeviceNet Master Module is also a slave to another Devicenet master device.

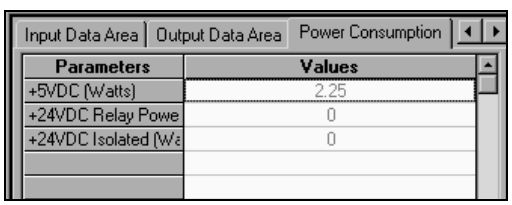
Note: Do not enter values on this tab if the DeviceNet Master Module is not also used as a slave. Entering values on this tab when the DeviceNet Master is not used as a slave causes the DeviceNet Master to fail to communicate with slaves.



Power Consumption Tab

Power consumption

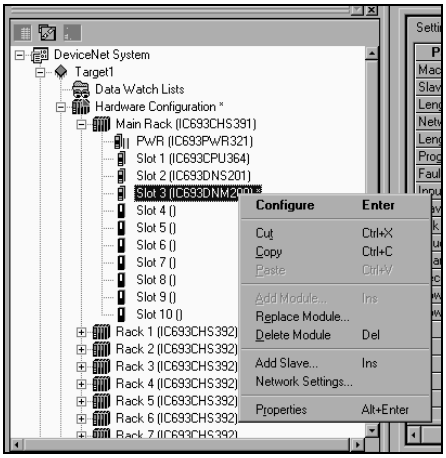
This read-only tab shows the backplane power that will be consumed by the DeviceNet Slave Module. This power will be used for module operation.



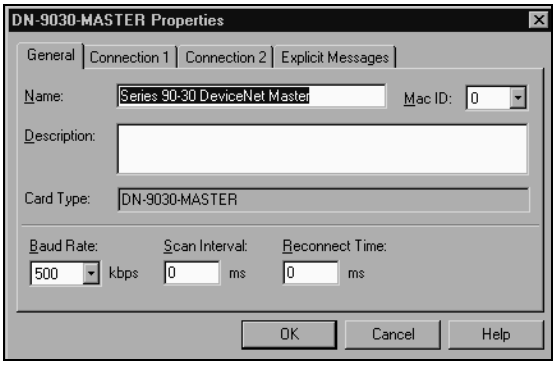
The DeviceNet Slave Module also draws power for its DeviceNet transceiver from the 24VDC power supply on the DeviceNet network.

Configuring Network Settings of a DeviceNet Master Module

To configure the Network Settings for a DeviceNet Master Module, right-click the DeviceNet Master in the PLC configuration, and choose Network Settings:



The Network Settings dialog box appears.



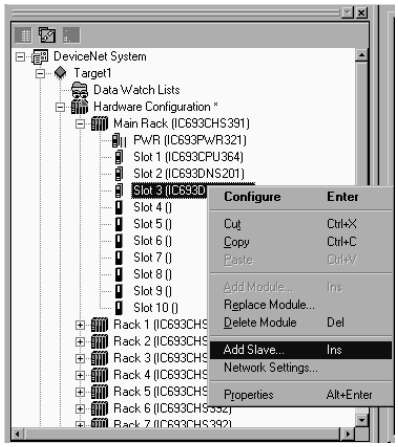
The General tab allows setting a name and description for the module. On this tab, you can also select the MACID, Baud Rate, Scan Interval, and Reconnect Time. These parameters are also found on the Configuration Parameter screen; they can be set in either place.

The rest of the tabs are only used if the DeviceNet Master Module will operate as a server (slave) to another master on the network. If the DeviceNet Master is not a slave to another DeviceNet master these three tabs should be left blank. Please refer to “Configuring Network Settings”on page 3-7 for detailed information about the Network Setting parameters.

Adding Slaves to the Network

After adding the DeviceNet Master Module to the PLC rack, you need to tell the Master about the required communication to the slave devices on the DeviceNet network by adding slaves to the Master. There are two ways to add a slave to the network configuration:

- In the Navigator window, right-click the IC693DNM200 Master Module and choose Add Slave.



- Select the slave type from the list that appears. Click OK to select the slave. For example, to add a DeviceNet Slave Module as a slave on the network, you would select:

- You can also drag and drop a device from the Toolchest to the DeviceNet master. Open the Toolchest by clicking the Toolchest button on the Tools toolbar. Select the DeviceNet Devices drawer. Choose a slave device.

  If you are editing a configuration that was uploaded from the PLC, the presence of device EDS files (and also device names and descriptions) depends on the DeviceNet Master Module configuration, as described earlier in this chapter. If the EDS files and names and descriptions were not downloaded, they will not be part of the uploaded configuration, and the EDS files may no longer be present in the Toolchest.

  Whether you are adding devices in the Navigator or using the drag and drop method , if you don't see the type of device you want to add and also don't have an EDS file for the device, you can only configure the slave as a generic device .

Note: The DeviceNet Master module (DNM200) does not present valid network status/firmware ID inputs unless at least one slave is configured. The DNM200's configuration should always include at least one slave module and the slave module should be present on the network.

Adding a Device's EDS File

If the device you want to configure is not listed and is not in the Toolchest, you can provide the EDS file, which is supplied by the device manufacturer, by clicking Have Disk. In the Open dialog box, browse to the EDS file and click Open. (When you select an EDS file using this method, it is added to the Slave Catalog and the DeviceNet Devices drawer of the Toolchest.)

Configuring Network Settings for Slaves Added to the Master

The Network Settings include MAC IDs, baud rates, and the messaging connections between devices on the network.

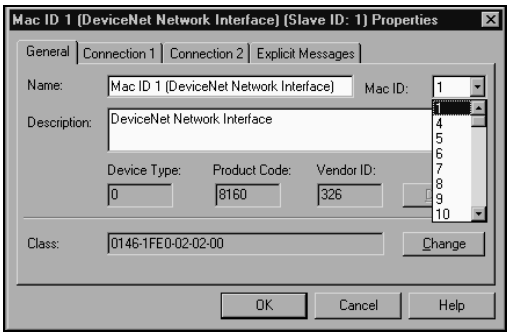
Assigning the MAC IDs and Baud Rate

Be sure the MAC IDs entered for slaves in the Machine Edition configuration match the MAC IDs set up for the devices themselves.

Many devices have their MAC IDs set by DIP switches. Some have their MAC IDs set by configuration commands from the master. All software-configured devices originally have the same default MAC ID: 63. To configure a software configured device, add a generic device with MACID 63 and enable Explicit Messaging to allow the configuration message to be sent to the device. Therefore, assigning the MAC ID 63 to be used by a device on the network should be done carefully, to prevent duplicate MAC ID conflicts when adding a new slave. Because all software-configurable slaves originally have the same default MAC ID, such slaves should be connected to the network one at a time. As each new slave is connected, its MAC ID should be changed using a Send Device Explicit COMMREQ, and its operation should be checked before connecting the next slave. Remember that the device will take on the new MAC ID after the configuration message is sent to it.

Configuring the MAC ID of a Slave Device

On the Network Settings: General tab, configure the device's MAC ID. The MAC ID will default to the next available address but will not fill in skipped addesses.The dropdown list displays MAC IDs that have not been used in the Machine Edition configuration. In this example, IDs 2 and 3 have already been assigned, so those numbers do not appear in the list.



Name

A field is provided to give the slave an identifying name

Description

A field is provide to add a description of the slave

Configuring I/O Messaging Connections for Slaves added to the Master

I/O Messaging is the term used for the routine and automatic exchange of data between the master and slaves in a DeviceNet system. Each configured I/O Message defines a dedicated communication path between a producing device and one or more consuming devices. Once these connections have been established, I/O Messaging communications occur automatically during system operation.

Each Series 90-30 device in a DeviceNet system can be set up for up to two different I/O Messaging connections. Each connection can be disabled (the default), or set up for Polled, Strobed, Change-of-State, or Cyclic operation. Connections should be configured to meet the needs of the application. For example, the master might Strobe all the input-only slave devices and Poll the remaining slave devices.

The selection made for one I/O Messaging connection to a slave determines which connection types remain available for the same slave's other I/O Messaging connection, as shown by the table below. For example, you can only select one polling connection for a device.

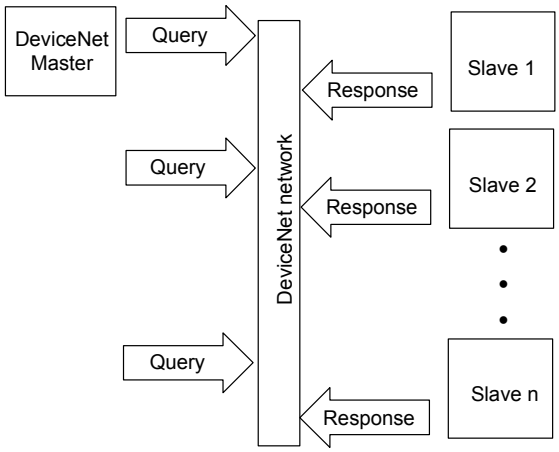
Selected for One Connection	*Available for the Other Connection*
Disabled	Disabled, polled, strobe, cos. cyclic
Polled	Disabled, strobed, cos, cyclic
Strobed	Disabled, polled, cos, cyclic
Cos	Disabled, polled, strobed
Cyclic	Disabled, polled, strobed

Configuration of each of these connection types is described on the following pages.

Configuring a Polled I/O Messaging Connection

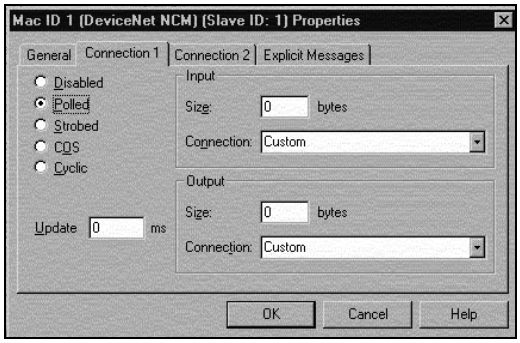
In Polled I/O mode (the most common method of doing I/O), the DeviceNet master automatically sends a message containing outputs to each slave with a connection configured for polling. The slave sends back a response containing input data. Polling therefore requires 2 messages to update the I/O data for each polled device.

The master completes the polling sequence as quickly as possible. Polling is the most accurate but least efficient method of updating I/O data. It is most suitable for high-availability control data that is used to drive application logic.



To configure polling for slave connection 1 or connection 2, select Polled on the Slave Properties menu.

Polled I/O Messaging Screen for a DeviceNet Slave in the DeviceNet Master configuration



For input resources, specify the number of data bytes the slave will send to the master.

For output resources, enter the number of bytes the slave will consume from the master.

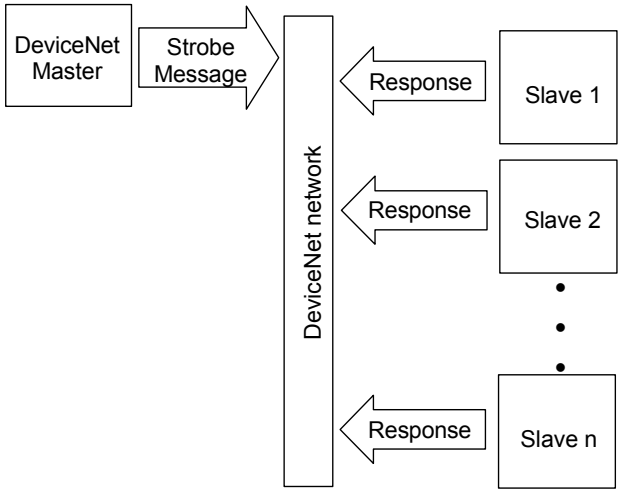
For Update, specify the interval in milliseconds at which the DeviceNet master will update data for this device. To select the fastest update rate possible, enter zero.

Note: The number of bytes for input and output must match between this setup and the setup of the slave device, or the DeviceNet Master Module will be unable to communicate with the slave.

Configuring a Strobed I/O Messaging Connection

In Strobed I/O mode, the master produces a single Strobe request message that is consumed by all devices with a connection configured for strobing, requesting their current status. This occurs at the rate selected using the Scan Interval parameter of the DeviceNet Master Module.

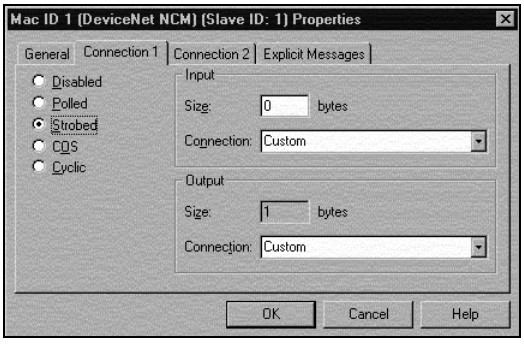
Each strobed device then responds with its input data. Devices respond in the order of their MAC IDs, beginning with the lowest MAC ID first. MAC IDs can be specifically assigned to prioritize I/O reporting by the slaves.



Strobed I/O Messaging can be more efficient than Polled I/O messaging because the master does not send an individual Poll request to each device. Strobed I/O Messaging is particularly useful for slave devices that have input data only, such as sensors.

To configure Strobed I/O Messaging for a slave connection, select Strobed on the Slave Properties menu.

Strobed I/O Messaging Screen for a Slave on the Network

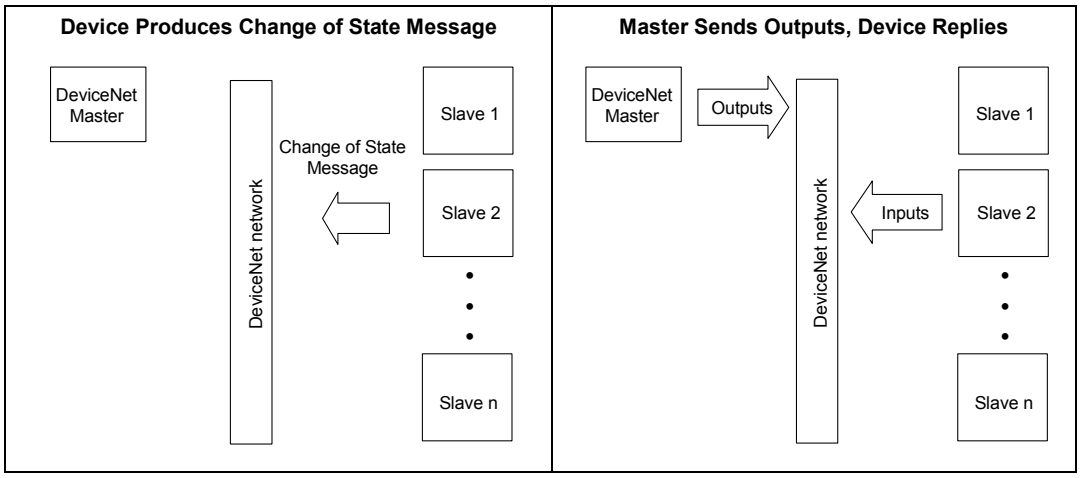


For input resources, specify the number of data bytes the device will send to the master.

The length for output resources is automatically set to 1 byte. The message from the Master to the Slave telling them to send back their inputs is a 1 byte message. It reflects the state of the I/O bit in the strobe request message for the device: set (1) or clear (0).

Configuring a Change-of-State (COS) I/O Messaging Connection

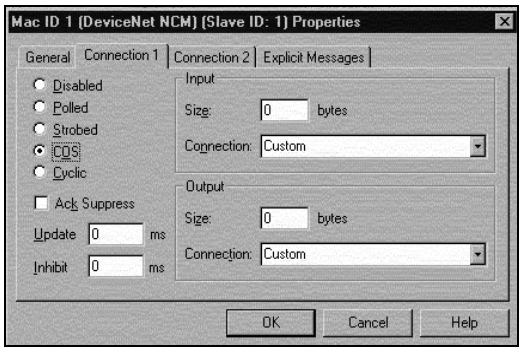
A connection configured for Change-of-State (COS) I/O Messaging is activated only when the device sends a message to the master, reporting a change of status. The master then sends an output message to the device and the device responds with its input data.



Change-of-State I/O Messaging is the most efficient type of messaging on the network, but it can be less precise than the other methods.

To configure Change-of-State I/O Messaging for a connection, select COS on the slave properties menu.

COS I/O Messaging Screen for a Slave on the Network



For input resources, specify the number of data bytes the device will send to the master.

For output resources, enter the number of bytes the device will consume from the master.

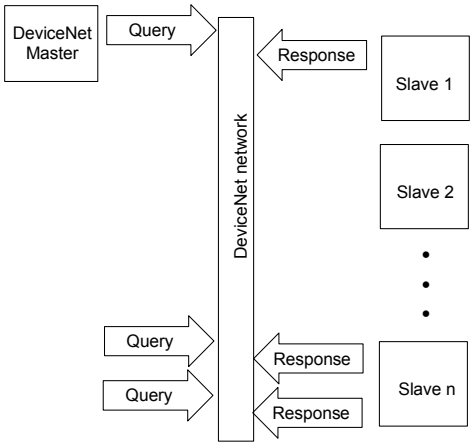
If Acknowledge Suppress is selected, the DeviceNet master will not wait for an acknowledge message from the device.

For Inhibit, specify the minimum delay in milliseconds between two data productions. For example, a slave is running a COS connection that has a change in data every 5ms, but the control application needs new data every 25ms. Setting the inhibit value to 25 causes the slave to transmit data at minimum intervals of 25ms and avoids needless use of network bandwidth.

For Update, specify the interval in milliseconds at which the DeviceNet master will update data for this device. To select the fastest update rate possible, enter zero.

Configuring a Cyclic I/O Messaging Connection

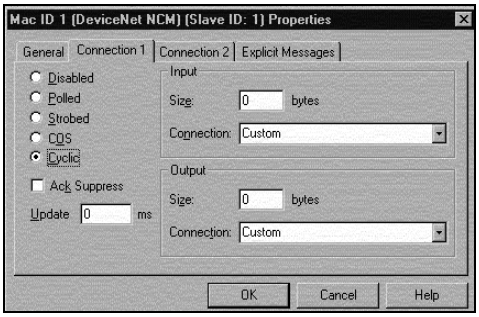
In Cyclic I/O Messaging as in Polled I/O Messaging, the DeviceNet master automatically sends a message containing outputs to a device with a connection configured for Cyclic update. The device sends back a response containing input data. Like Polling, Cyclic I/O Messaging requires 2 messages to update the I/O data for a device. Unlike Polled messaging, Cyclic messaging can use a different interval as configured for each slave.



Cyclic messaging can be appropriate for devices such as analog input sensors. For example, a temperature sensor might use Cyclic messages to report its measurements every 500ms. Cyclic messaging can cut down on network traffic while accurately capturing certain types of input measurements. This can also be more efficient for the application program in the PLC CPU. A Cyclic I/O connection can also be used as a 'heartbeat' to provide assurance of a device's continued operation, with a Change-of-State I/O connection to the same device used to update its I/O state.

To configure Cyclic I/O Messaging for a connection, select Cyclic on the Slave Properties menu.

Cyclic I/O Messaging Screen for a Slave on the Network



For input resources, specify the number of data bytes the device will send to the master.

For output resources, enter the number of bytes the device will consume from the master.

If Acknowledge Suppress flag is selected, the DeviceNet master does not wait for an acknowledge message from the device.

For Update, specify the interval in milliseconds at which the DeviceNet master will update data for a network device. To select the fastest update rate possible, enter zero.

Configuring DeviceNet Explicit Messaging

Explicit Messaging is the highest priority messaging. Explicit messaging provides access to objects other than the default I/O connection set, and optionally creates a buffer for explicit connection. Not all slaves support explicit services, and many, including most discrete I/O devices, do not use explicit services.

Some devices rely on Explicit Messaging for configuration of selected parameters. For example, some VersaPoint modules can be configured using Explicit Messaging. To be configured by Explicit Messaging, an Explicit Messaging connection to the device must first be set up as shown below.

If Explicit Messaging should be enabled, click on Enable Explicit Connection on the Explicit Messages tab.

In addition, for a DeviceNet Slave Module, or DeviceNet Master module operating as a slave, specify the message request and message response size. Make sure the size specified is large enough. The DeviceNet modules implement Explicit Messaging through the use of COMMREQ instructions in the application program. These COMMREQ messages are described in chapter 6, "Programmed Communications." The message sizes of each explicit message type are described in that chapter.

Explicit Messaging Screen for a Slave on the Network

Configuring Network Settings for a DeviceNet Master Acting as a Slave

The Network Settings include MAC IDs, baud rates, and the messaging connections between devices on the network. This was described previously in the section "Configuring Network Settings for Slaves Added to the Master".

Configuring I/O Messaging Connections for a DeviceNet Master Acting as a Slave

I/O Messaging is the term used for the routine and automatic exchange of data between the master and slaves in a DeviceNet system. Each configured I/O Message defines a dedicated communication path between a producing device and one or more consuming devices. Once these connections have been established, I/O Messaging communications occur automatically during system operation.

Each Series 90-30 device in a DeviceNet system can be set up for up to two different I/O Messaging connections. Each connection can be disabled (the default), or set up for Polled, Strobed, Change-of-State, or Cyclic operation. Connections should be configured to meet the needs of the application. For example, the master might Strobe all the input-only slave devices and Poll the remaining slave devices.

The selection made for one I/O Messaging connection to a slave determines which connection types remain available for the same slave's other I/O Messaging connection, as shown by the table below. For example, you can only select one polling connection for a device.

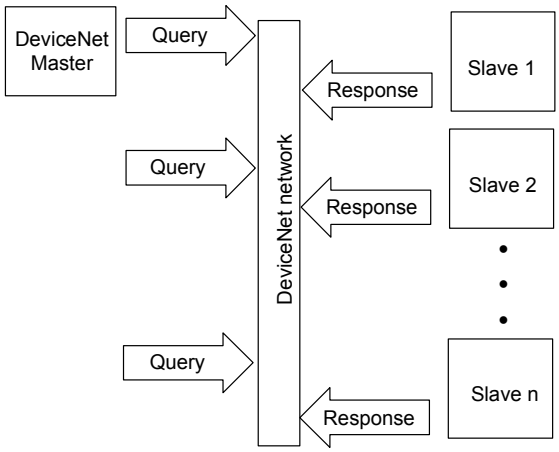
Selected for One Connection	*Available for the Other Connection*
Disabled	Disabled, polled, strobe, cos. cyclic
Polled	Disabled, strobed, cos, cyclic
Strobed	Disabled, polled, cos, cyclic
Cos	Disabled, polled, strobed
Cyclic	Disabled, polled, strobed

Configuration of each of these connection types is described on the following pages.

Configuring a Polled I/O Messaging Connection

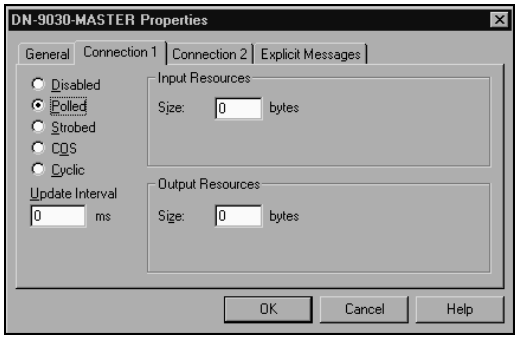
In Polled I/O mode (the most common method of doing I/O), the DeviceNet master automatically sends a message containing outputs to each slave with a connection configured for polling. The slave sends back a response containing input data. Polling therefore requires 2 messages to update the I/O data for each polled device.

The master completes the polling sequence as quickly as possible. Polling is the most accurate but least efficient method of updating I/O data. It is most suitable for high-availability control data that is used to drive application logic.



To configure polling for slave connection 1 or connection 2, select Polled on the Slave Properties menu.

Polled I/O Messaging Screen for a DeviceNet Master Module



For input resources, specify the number of data bytes the slave will send to the master.

For output resources, enter the number of bytes the slave will consume from the master.

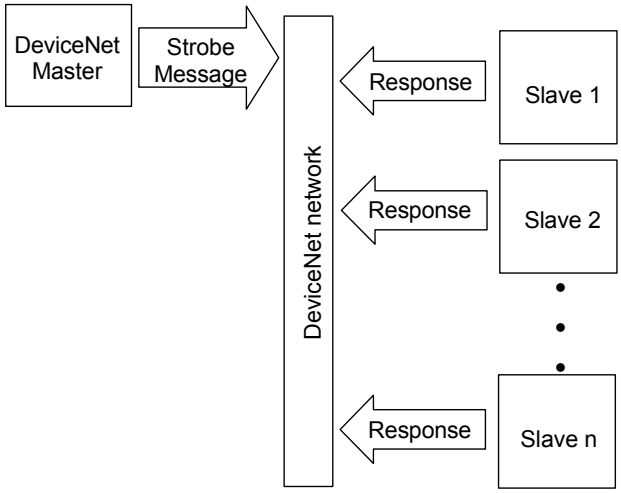
For Update, specify the interval in milliseconds at which the DeviceNet master will update data for this device. To select the fastest update rate possible, enter zero.

Note: The number of bytes for input and output must match between this setup and the setup of the Slave Device or the Master will be unable to communicate with the slave.

Configuring a Strobed I/O Messaging Connection

In Strobed I/O mode, the master produces a single Strobe request message that is consumed by all devices with a connection configured for strobing, requesting their current status. This occurs at the rate selected using the Scan Interval parameter of the DeviceNet Master Module.

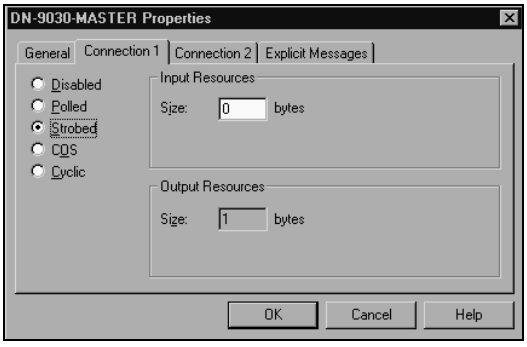
Each strobed device then responds with its input data. Devices respond in the order of their MAC IDs, beginning with the lowest MAC ID first. MAC IDs can be specifically assigned to prioritize I/O reporting by the slaves.



Strobed I/O Messaging can be more efficient than Polled I/O messaging because the master does not send an individual Poll request to each device. Strobed I/O Messaging is particularly useful for slave devices that have input data only, such as sensors.

To configure Strobed I/O Messaging for a slave connection, select Strobed on the Slave Properties menu.

Strobed I/O Messaging Screen for a DeviceNet Master Module

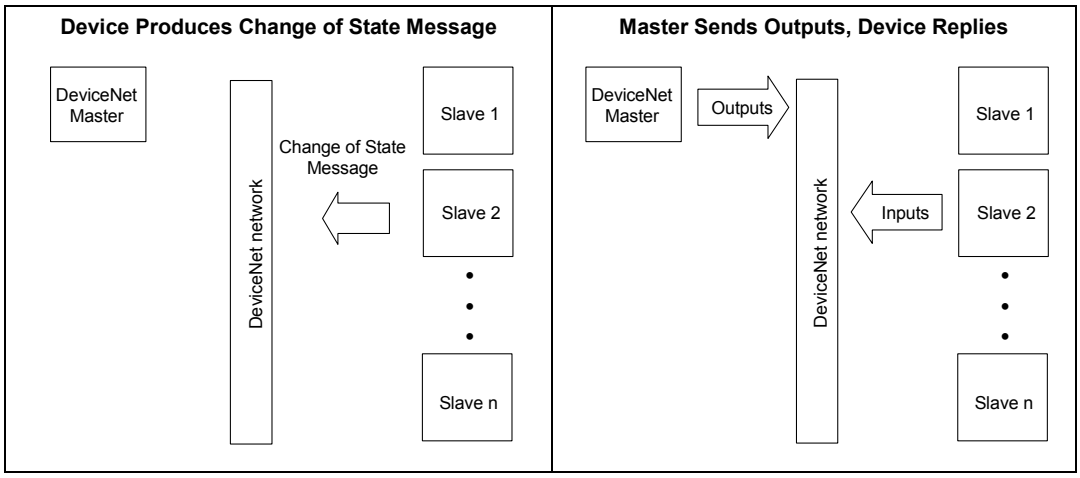


For input resources, specify the number of data bytes the device will send to the master.

The length for output resources is automatically set to 1 byte. The message from the Master to the Slave telling them to send back their inputs is a 1 byte message. It reflects the state of the I/O bit in the strobe request message for the device: set (1) or clear (0).

Configuring a Change-of-State (COS) I/O Messaging Connection

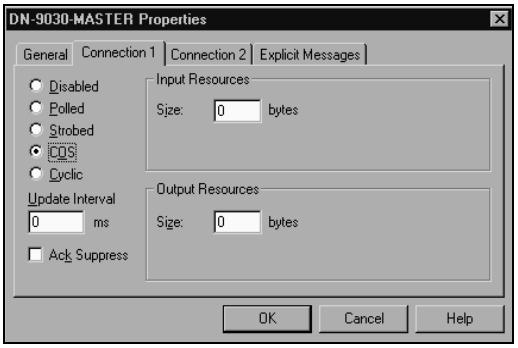
A connection configured for Change-of-State (COS) I/O Messaging is activated only when the device sends a message to the master, reporting a change of status. The master then sends an output message to the device and the device responds with its input data.



Change-of-State I/O Messaging is the most efficient type of messaging on the network, but it can be less precise than the other methods.

To configure Change-of-State I/O Messaging for a connection, select COS on the slave properties menu.

COS I/O Messaging Screen for a Series 90-30 DeviceNet Master Module



For input resources, specify the number of data bytes the device will send to the master.

For output resources, enter the number of bytes the device will consume from the master.

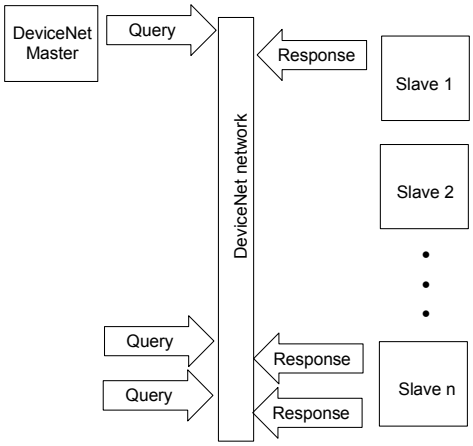
If Acknowledge Suppress is selected, the DeviceNet master will not wait for an acknowledge message from the device.

For Inhibit, specify the minimum delay in milliseconds between two data productions. For example, a slave is running a COS connection that has a change in data every 5ms, but the control application needs new data every 25ms. Setting the inhibit value to 25 causes the slave to transmit data at minimum intervals of 25ms and avoids needless use of network bandwidth.

For Update, specify the interval in milliseconds at which the DeviceNet master will update data for this device. To select the fastest update rate possible, enter zero.

Configuring a Cyclic I/O Messaging Connection

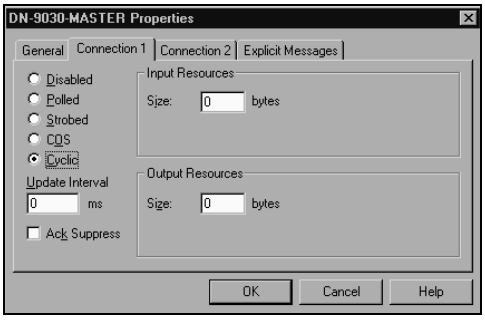
In Cyclic I/O Messaging as in Polled I/O Messaging, the DeviceNet master automatically sends a message containing outputs to a device with a connection configured for Cyclic update. The device sends back a response containing input data. Like Polling, Cyclic I/O Messaging requires 2 messages to update the I/O data for a device. Unlike Polled messaging, Cyclic messaging can use a different interval as configured for each slave.



Cyclic messaging can be appropriate for devices such as analog input sensors. For example, a temperature sensor might use Cyclic messages to report its measurements every 500ms. Cyclic messaging can cut down on network traffic while accurately capturing certain types of input measurements. This can also be more efficient for the application program in the PLC CPU. A Cyclic I/O connection can also be used as a 'heartbeat' to provide assurance of a device's continued operation, with a Change-of-State I/O connection to the same device used to update its I/O state.

To configure Cyclic I/O Messaging for a connection, select Cyclic on the Slave Properties menu.

Cyclic I/O Messaging Screen for a DeviceNet Master Module



For input resources, specify the number of data bytes the device will send to the master.

For output resources, enter the number of bytes the device will consume from the master.

If Acknowledge Suppress flag is selected, the DeviceNet master does not wait for an acknowledge message from the device.

For Update, specify the interval in milliseconds at which the DeviceNet master will update data for a network device. To select the fastest update rate possible, enter zero.

Configuring DeviceNet Explicit Messaging

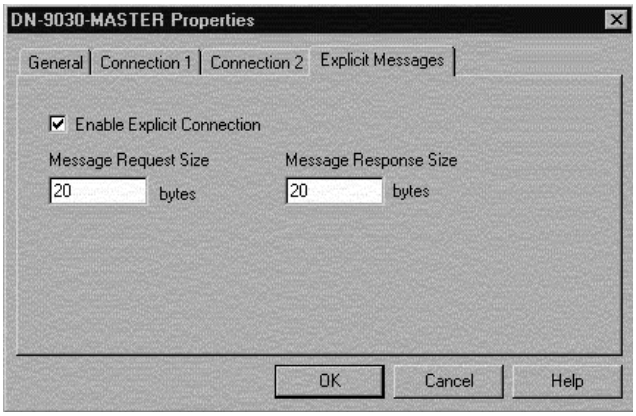
Explicit Messaging is the highest priority messaging. Explicit messaging provides access to objects other than the default I/O connection set, and optionally creates a buffer for explicit connection. Not all slaves support explicit services, and many, including most discrete I/O devices, do not use explicit services.

Some devices rely on Explicit Messaging for configuration of selected parameters. For example, some VersaPoint modules can be configured using Explicit Messaging. To be configured by Explicit Messaging, an Explicit Messaging connection to the device must first be set up as shown below.

If Explicit Messaging should be enabled, click on Enable Explicit Connection on the Explicit Messages tab.

In addition, for a DeviceNet Slave Module, or DeviceNet Master module operating as a slave, specify the message request and message response size. Make sure the size specified is large enough. The DeviceNet modules implement Explicit Messaging through the use of COMMREQ instructions in the application program. These COMMREQ messages are described in chapter 6, "Programmed Communications". The messages sizes of each explicit message type are described in that chapter.

Explicit Messaging Screen for a DeviceNet Master Module



Chapter 4

PLC Configuration for the DeviceNet Slave

This chapter explains how to add a DeviceNet Slave Module to the configuration of the Series 90-30 PLC. It also explains how to configure communications connections between a DeviceNet Slave Module and the DeviceNet network.

- Configuration Steps
- Adding a DeviceNet Slave Module to the PLC Configuration
- Configuring the Parameters of a DeviceNet Slave Module
- Configuring the Network Settings of a DeviceNet Slave Module
  - Configuring the MAC ID
  - Configuring I/O Messaging Connections
  - Configuring DeviceNet Explicit Messaging

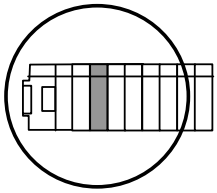
These configuration procedures are written for users who have a basic knowledge of the Machine Edition Logic Developer software and the Series 90-30 PLC. For help with using the software, please see the software's built-in help system.

Note: The DeviceNet Slave Module is only supported in Machine Edition Logic Developer. The Logicmaster™, VersaPro™, and Control software do not support this module.

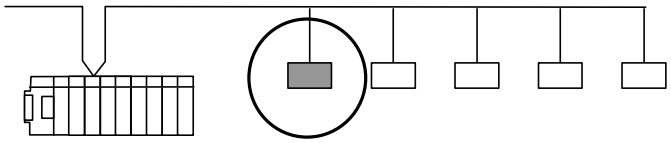
Configuration Steps

There are three basic steps to configuring a DeviceNet Slave Module:

- Configuring its module parameters as part of the PLC CPU system in which it is installed. This is described on the next page.



- Configuring its network settings. For information, see the section "Configuring Network Settings " later in this chapter.

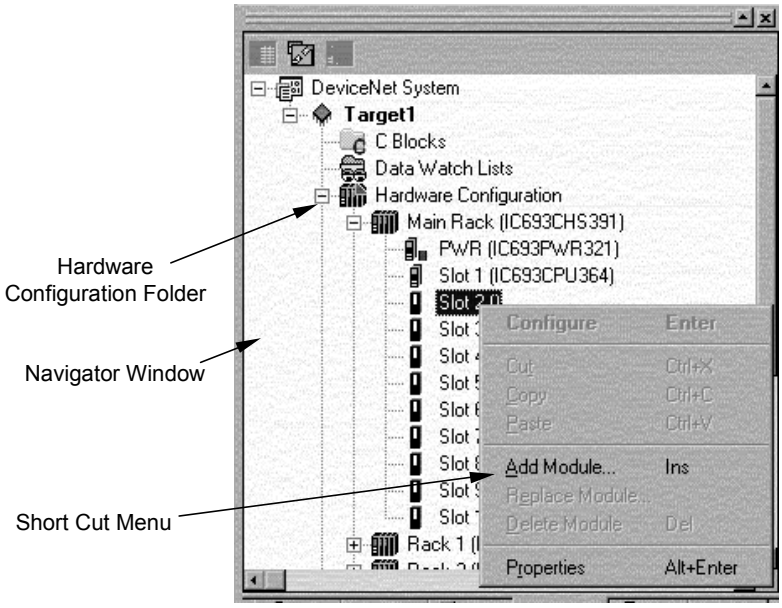


- Configuring the module as a slave on the network of a master module. Instructions for configuring the network of a DeviceNet Master Module are given in chapter 3. Instructions for configuring other master types are not described in this manual.

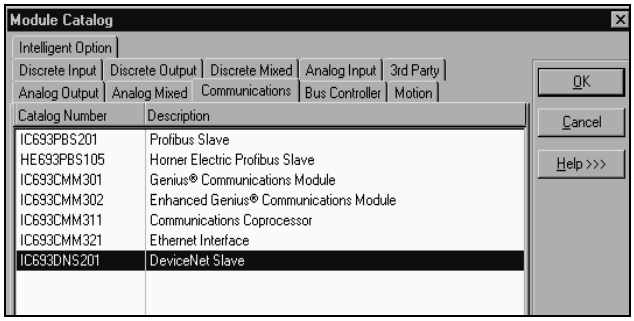
Adding a DeviceNet Slave Module to the PLC Configuration

First, add the DeviceNet Slave Module to the PLC rack configuration. The DeviceNet Slave Module is compatible with any Series 90-30 CPU except IC693CPU321 and IC693CPU340. It requires release 8.0 CPU firmware as a minimum. Release 10.6 or later is recommended, if available for your particular CPU.

1. In that configuration, in the Project tab of the Navigator, expand the Hardware Configuration folder.
2. In the Hardware Configuration folder, right click the PLC Slot for the DeviceNet Slave Module. It can be in slot 2 or higher in the main rack.
3. Select Add Module from the shortcut menu.

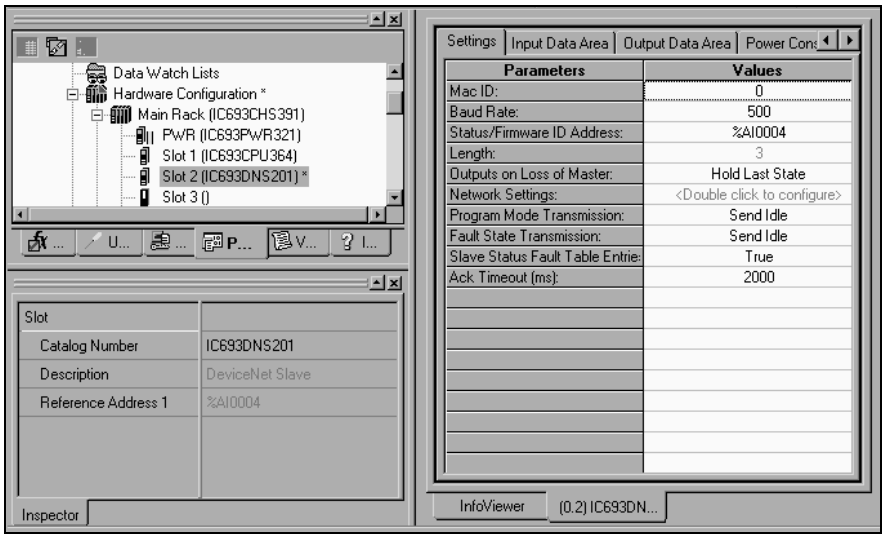


4. The Module Catalog dialog box appears. To add a DeviceNet Slave Module, click on the Communications tab. Select the IC693DNS201 DeviceNet Slave from the list and click OK.



Configuring the Parameters of a DeviceNet Slave Module

After selecting the module, its Parameter Editor window appears in the InfoViewer window space.

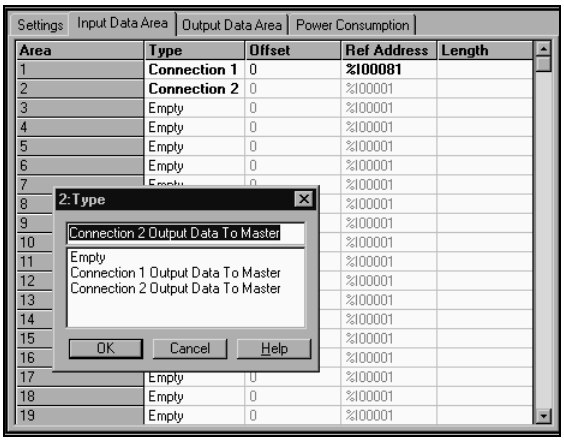


Parameters of a DeviceNet Slave Module

Settings tab	
Mac ID	The Mac ID (medium access control identification) of the slave on the DeviceNet network. Valid range: 0 - 63. Default: 0. To set this parameter in the DeviceNet Slave Properties dialog box, go to the Navigator, right-click the DeviceNet slave and choose Network Settings. The DeviceNet Slave Properties dialog box appears.
Baud Rate (kbps)	The data transmission rate for the DeviceNet Slave Module. The maximum baud rate that can be used depends on the bus length and cable type. See chapter 2 for more information. Choose: 125K, 250K, or 500K. Default is 500.
Status/ Firmware ID Address	The starting address for three words of module status information. The default is %AI memory. The Network Status/Firmware ID Address must be a non-overlapping range in %AI, %I, %Q, %G, %AQ, %R, %T, or %M. The default offset is the next available reference in the memory type selected. The Firmware ID word contains the current firmware version running on the DeviceNet Master Module. The Major Revision number resides in the upper byte and the Minor Revision number resides in the lower byte of this word.
Length (of module status/firmware ID)	This is the length of the Network Status / Firmware ID Address memory location described above. The length is 3 words or 48 bits (read only).
Outputs on Loss of Master	This parameter determines how a slave will handle inputs / outputs if the slave loses communications with the master. The default is Hold Last State. It can be changed to clear.
Network Settings	Double-click to edit the network communications settings of the slave. See "Configuring Network Settings" in this chapter for more information.
Program Mode Transmission	When the PLC is in Program mode (Stop mode), the module can either send idle packets or data to zero. The default is to send idle packets.
Fault State Transmission	When the module detects a PLC fault, the module can either send idle packets or set data to zero. The default is to send idle packets.
Slave Status Fault Table Entries	When slave communications status events (loss and re-establish) occur, the DeviceNet Master Module can either report them in the fault table or not. If this setting is True (the default), the Master makes fault table entries. If this setting False, slave status events are not reported to the fault table.
Ack Timeout (ms)	Number of milliseconds to wait for a CAN Acknowledge of the Duplicate MacID check (performed during startup) before reporting an Ack (acknowledge) failure. Valid range: 0 to 65,535. The default is 2,000ms (2 seconds).

Data Areas Tabs

These tabs show the PLC program references assigned to the selected module: These reference assignments can be edited, or left at their defaults. Note that the input data area tab shows *master* inputs, which are module outputs. The output data area tab shows *master* outputs, which are module inputs. For both inputs and outputs, data can be configured for both connection 1 and connection 2.



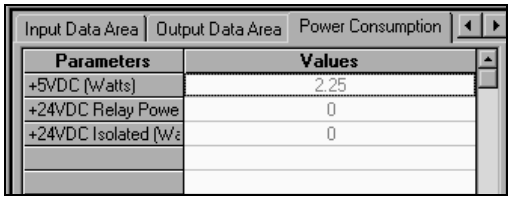
For both inputs and outputs, the offset is the starting address in the *master* PLC memory area. It must be less than the size of the data area specified in the Network Settings dialog box. To specify data area Size, in the Navigator, right-click the DeviceNet slave and choose Network Settings. The DeviceNet Slave Properties dialog box appears. On the Connection 1 or Connection 2 tab, set the Size under Resources.

The Reference Address is the location in *master* PLC memory where the data is mapped. This field is read-only if Size is set to 0. This must be a range in %AI, %I, %Q, %G, %AQ, %R, %T, or %M. If the number of bytes (Size) is odd, only the discrete addresses (%I, %Q, %G, %M, %T) can be used.

Power Consumption Tab

Power consumption

This read-only tab shows the backplane power that will be consumed by the DeviceNet Slave Module. This power will be used for module operation.



The DeviceNet Slave Module also draws power for its DeviceNet transceiver from the 24VDC power supply on the DeviceNet network.

Configuring the Network Settings of a DeviceNet Slave Module

To configure the Network Settings for a DeviceNet Slave Module, right-click the DeviceNet Slave in the PLC configuration, and choose Network Settings.

The Network Settings dialog box appears.

The General tab allows setting a name and description for the module. On this tab, you can also select the MACID and Baud Rate. These parameters are also found on the Configuration Parameter screen; they can be set in either place.

The rest of the tabs set up the messaging connections that will be used by the module.

Configuring the MAC ID

All software-configured devices originally have the same default MAC ID: 63. Therefore, assigning the MAC ID 63 to be used by a device on the network should be avoided if possible, to prevent duplicate MAC ID conflicts when adding a new slave.

Configuring I/O Messaging Connections

I/O Messaging is the term used for the routine and automatic exchange of data between the master and slaves in a DeviceNet system. Each configured I/O Message defines a dedicated communication path between a producing device and one or more consuming devices. Once these connections have been established, I/O Messaging communications occur automatically during system operation.

The DeviceNet Slave Module can be set up for up to two different I/O Messaging connections. Each connection can be disabled (the default), or set up for Polled, Strobed, Change-of-State, or Cyclic operation. Connections should be configured to meet the needs of the application.

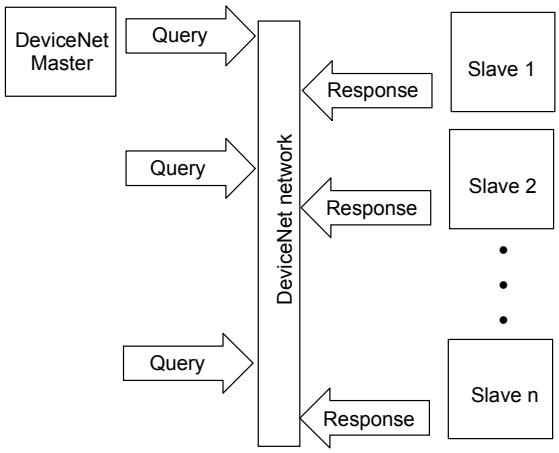
The selection made for one I/O Messaging connection determines which connection types remain available for the other I/O Messaging connection, as shown by the table below. For example, you can only select one polling connection for the module.

Selected for One Connection	*Available for the Other Connection*
Disabled	Disabled, polled, strobe, cos. cyclic
Polled	Disabled, strobed, cos, cyclic
Strobed	Disabled, polled, cos, cyclic
Cos	Disabled, polled, strobed
Cyclic	Disabled, polled, strobed

Configuration of each of these connection types is described on the following pages.

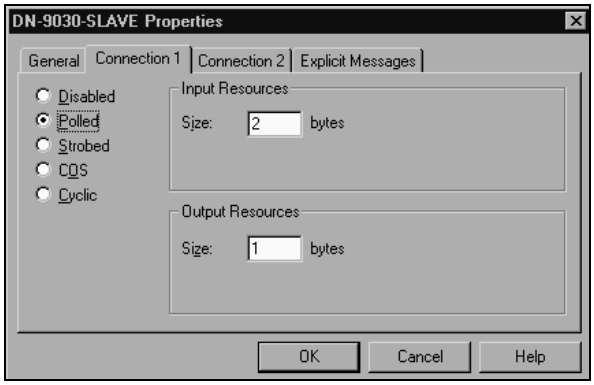
Configuring a Polled I/O Messaging Connection

In Polled I/O mode, the DeviceNet master automatically sends a message containing outputs to each slave with a connection configured for polling. The slave sends back a response containing input data. Polling therefore requires 2 messages to update the I/O data for each polled device.



To configure polling for slave connection 1 or connection 2, select Polled on the Slave Properties menu.

Polled I/O Messaging Screen for a DeviceNet Slave Module



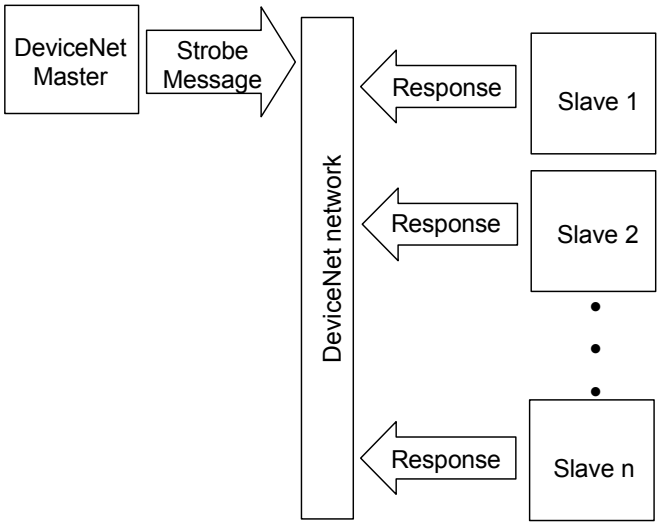
For input resources, specify the number of data bytes the DeviceNet Slave Module will send to the master.

For output resources, enter the number of bytes the DeviceNet Slave Module will consume from the master.

Configuring a Strobed I/O Messaging Connection

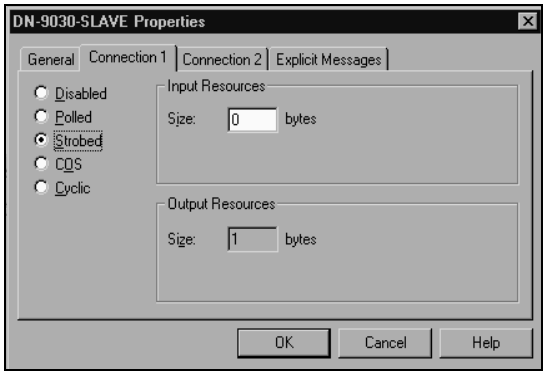
In Strobed I/O mode, the master produces a single Strobe request message that is consumed by all devices with a connection configured for strobing, requesting their current status.

Each strobed device then responds with its input data. Devices respond in the order of their MAC IDs, beginning with the lowest MAC ID first. MAC IDs can be specifically assigned to prioritize I/O reporting by the slaves.



To configure Strobed I/O Messaging for a connection, select Strobed on the Slave Properties menu.

Strobed I/O Messaging Screen for a DeviceNet Slave Module

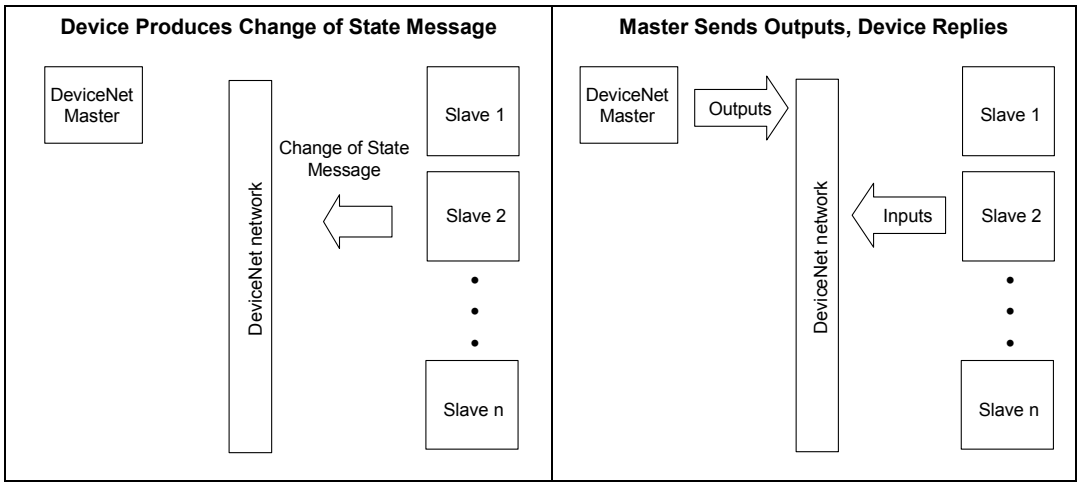


For input resources, specify the number of data bytes the module will send to the master.

The length for output resources is automatically set to 1 byte. The message from the Master to the Slaves telling them to send back their inputs is a 1 byte message. It reflects the state of the I/O bit in the strobe request message for the device: set (1) or clear (0).

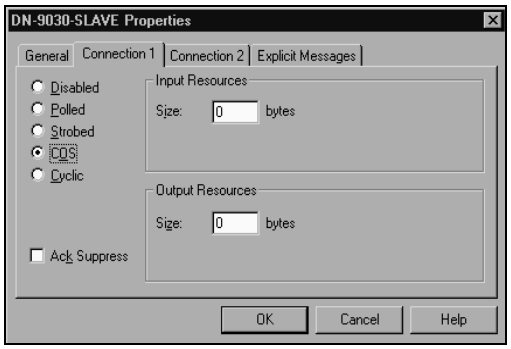
Configuring a Change-of-State (COS) I/O Messaging Connection

A connection configured for Change-of-State (COS) I/O Messaging is activated only when the module sends a message to the master, reporting a change of status. The master then sends an output message to the module and the module responds with its input data.



To configure Change-of-State I/O Messaging for a connection, select COS on the Connection tab menu.

COS I/O Messaging Screen for a DeviceNet Slave Module



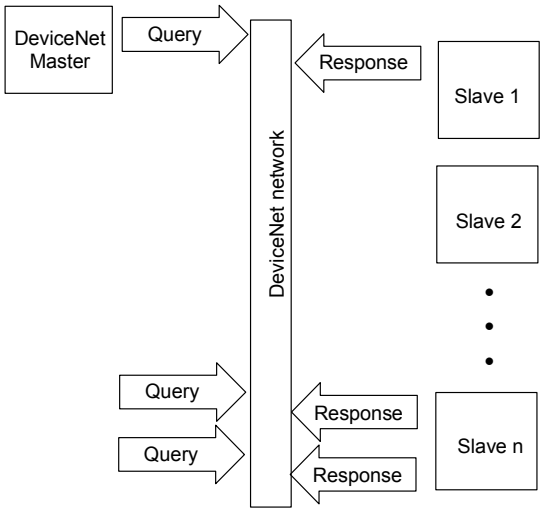
For input resources, specify the number of data bytes the module will send to the master.

For output resources, enter the number of bytes the module will consume from the master.

If Acknowledge Suppress is selected, the master will not wait for an acknowledge message from the module.

Configuring a Cyclic I/O Messaging Connection

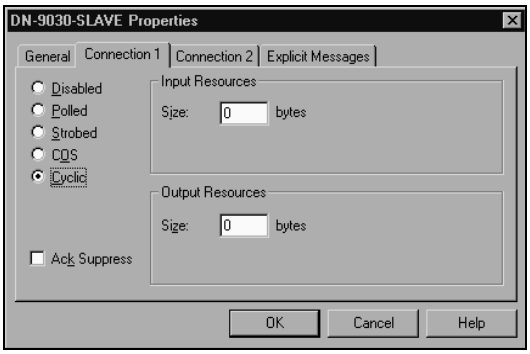
In Cyclic I/O Messaging as in Polled I/O Messaging, the DeviceNet master automatically sends a message containing outputs to a device connection configured for Cyclic update. The module sends back a response containing input data. Like Polling, Cyclic I/O Messaging requires 2 messages to update the I/O data for a device. Unlike Polled messaging, Cyclic messaging can use a different interval as configured for each slave.



A Cyclic I/O connection can be used as a 'heartbeat' to provide assurance of a device's continued operation, with a Change-of-State I/O connection to the same device used to update its I/O state.

To configure Cyclic I/O Messaging for a connection, select Cyclic on the Slave Properties menu.

Cyclic I/O Messaging Screen for a DeviceNet Slave Module



For input resources, specify the number of data bytes the module will send to the master.

For output resources, enter the number of bytes the module will consume from the master.

If Acknowledge Suppress flag is selected, the DeviceNet master does not wait for an acknowledge message from the module.

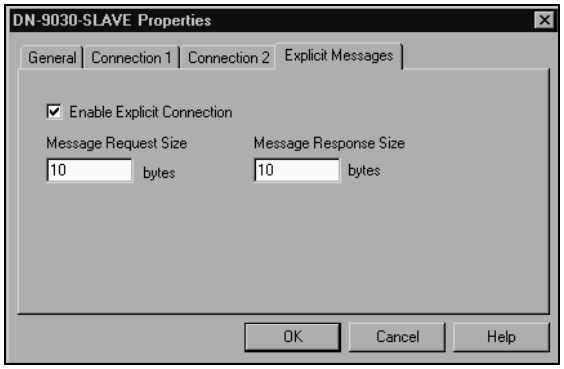
Configuring DeviceNet Explicit Messaging

Explicit Messaging is the highest priority messaging. Explicit messaging provides access to objects other than the default I/O connection set, and optionally creates a buffer for explicit connection.

If Explicit Messaging should be enabled for the DeviceNet Slave Module, click on Enable Explicit Connection on the Explicit Messages tab.

Also specify the message request and message response size. Make sure the size specified is large enough. The DeviceNet module implements Explicit Messaging through the use of COMMREQ instructions in the application program. These COMMREQ messages are described in chapter 6, "Programmed Communications." The messages sizes of each explicit message type are described in that chapter.

Explicit Messaging Screen for a DeviceNet Slave Module



Chapter 5 Module Operation

This chapter describes how the DeviceNet Master Module and the DeviceNet Slave Module function in a control system.

- Operation of the DeviceNet Master Module
  - Operation while the PLC is in Stop Mode
  - Operation while the PLC is in Run Mode
- Operation of a Slave(Server) DeviceNet Module
- Fault Table Entries for a DeviceNet Module
  - Faults Reported to the I/O Fault Table
  - Faults Reported to the PLC Fault Table
- PLC Status References for a DeviceNet Module
- Device Status Bits for the DeviceNet Master Module

Operation of a DeviceNet Master Module

A DeviceNet Master Module automatically exchanges data with devices on the network once connections are established. However, the content of the data it sends to network devices depends on the operating mode of the PLC.

DeviceNet Master Module Operation while the PLC is in Stop Mode

While the PLC is in STOP Mode, the DeviceNet Master Module updates its internal memory with input data from devices on the network, and with any output data it may receive from the PLC CPU. However, the DeviceNet Master Module does not transmit PLC output data on the network until the PLC returns to Run mode. Depending on its configuration, it transmits either of the following during Stop mode:

- ***Idle Messages (zero length)*** - Transmit Idle is the default state. In the Transmit Idle state, the DeviceNet Master Module sends zero-length I/O messages to all its configured DeviceNet devices. Each device detects this as a "Receive Idle" condition, as defined in the DeviceNet Specification. The device then implements its configured safe state behavior.
- ***Zeroed Data*** – In the Send Zeroed Data state, the DeviceNet Master Module sets the output data for all devices to zero.

If the DeviceNet Master Module receives a new configuration while the PLC is in Stop Mode, it closes down connections with all DeviceNet devices and processes the new configuration.

Operation when the PLC Transitions from Stop Mode to Run Mode

When the PLC transitions from Stop mode to Run mode, the DeviceNet Master Module transmits the most recent output data it received from the PLC CPU to nodes on the DeviceNet network. If no previous output updates were received, the network output data is all zeros.

DeviceNet Master Module Operation while the PLC is in Run Mode

While the PLC is in Run mode, the DeviceNet Master Module communicates with the devices on the network according to their configured connections for polled, strobe, change-of-state, or cyclic communications.

- The DeviceNet Master Module sends the strobed multi-cast message first. Devices that have been configured for Strobed I/O messaging begin responding, in order of their MAC IDs.
- The module then interrogates each device connection that is configured for Polling. Polling can begin while strobe responses are being received. The module performs this polling sequence as quickly as possible.
- The module interrogates devices that have been configured for Cyclic communications according to their configured update rates. The devices then respond with their present inputs.
- Network devices configured for Change-of-State communications send the master a message only when their status changes. After receiving a Change-of-State message

from a device, the master sends a request to the device. The device responds with its new data.

The DeviceNet Master Module stores data it receives from the network in its own memory. During normal operation, the CPU reads input data from the DeviceNet Master Module during its normal input scan. The PLC CPU updates the data in each device's configured memory location.

After solving the application program, the PLC CPU sends the appropriate output data to the DeviceNet Master Module. The module then provides output data to network devices according to their configured I/O Messaging communications type.

Operation if the CPU Stops Reading Inputs from the Module in PLC Run Mode

If the PLC CPU stops reading inputs from the DeviceNet Master Module for an interval of 1 second, the module assumes the CPU is no longer communicating and declares a PLC fault. The module continues to monitor inputs from the network but does not send the input data to the PLC CPU. It also stops sending output data on the network. Instead, it either transmits idle messages or zeroed data as described for Stop Mode. Network devices then enter their own configured safe states.

The module does not recover until the rack containing the module is power-cycled. Because of this, if you have a DNM200 (or DNS201) module in your PLC, do not execute a SUS_IO (Suspend I/O) function block or SVC_REQ #45 (Skip Next I/O Scan).

RX3i users should use caution when configuring the DNM200 for a scan set that is not scanned every sweep (the # of sweeps times the sweep time cannot exceed 1000ms). Finally, RX3i users should not let the CPU's sweep time approach a length of 1000ms.

Detecting Bus-Off Errors

If a DeviceNet bus-off failure occurs, the module is not able to communicate on the network. The CAN Status portion of the module's status data (shown in this chapter) indicates that a bus-off condition has occurred. For a DeviceNet Master Module, after the internal connection timeouts expire, the status of each node indicates that the device is no longer active. The application can monitor both the CAN Status word and the Slave Status bits to check for bus-off conditions and loss of slaves.

Fatal Errors

If a DeviceNet Module has a fatal error, it takes the following actions if possible:

- It sets its Module Status LED to solid RED and its Network Status LED to OFF.
- It informs the PLC CPU.
- It stops any current network activity.

The module tries to maintain communications with the PLC CPU, and to respond if it receives COMMREQ #9, "Read Module Header". In addition, the module starts continuously sending the Module Header information to its RS-232 serial port, using the following communications parameters: 19200 baud, 8N1 (8 data bits, no parity, 1 stop bit):

- The entire Module Header in ASCII hex codes.
- The message field of the Module Header in readable form.

Operation of a Slave (Server) DeviceNet Module

A DeviceNet Slave Module operates as a server (slave) to a master device on a DeviceNet network. The DeviceNet Master Module can be also be configured for server operation, and can perform the same functions as the DeviceNet Slave Module. For the DeviceNet Master module, use of the server features is optional.

Server features are configured using the module's Network Settings. The module can be configured for two types of I/O Messaging connections. It can also be configured to enable (or not enable) DeviceNet Explicit Messaging.

I/O Messaging for a DeviceNet Module

PLC references are assigned to each I/O Messaging connection configured for the server. A master on the DeviceNet network can use these I/O Messaging connections to read or write data in up to two memory areas in the PLC CPU. One Memory Area for Polled and one memory area for cyclic, strobed, or COS I/O.

Explicit Messaging for a DeviceNet Module

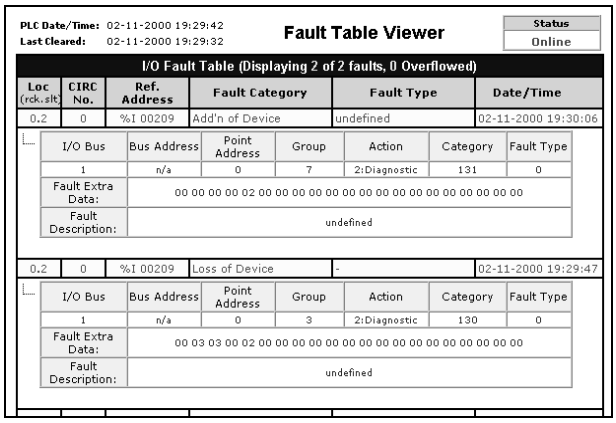
If Explicit Messaging is enabled in its Network Settings, a module can receive DeviceNet Explicit Messages from a network master. The module automatically responds to Explicit messages, except for Explicit Messages for User defined objects which are queued for transfer to the PLC CPU where the user application program is responsible for handling them.. The application program in the PLC CPU reads the queued Explicit Messages individually and can respond appropriately. The process of doing this is explained in chapter 5, "Programmed Communications".

Error Handling

If an error occurs, the DeviceNet Slave Module handles it as described on the previous page for a DeviceNet Master Module.

Fault Table Entries for a DeviceNet Module

A DeviceNet module automatically reports fault information to the PLC Fault Table screens.



PLC Date/Time: 02-11-2000 19:29:42
Last Cleared: 02-11-2000 19:29:32

Fault Table Viewer

Status: Online

I/O Fault Table (Displaying 2 of 2 faults, 0 Overflowed)

Loc (rck.slt)	CIRC No.	Ref. Address	Fault Category	Fault Type	Date/Time
0.2	0	%I 00209	Add'n of Device	undefined	02-11-2000 19:30:06

I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type
1	n/a	0	7	2:Diagnostic	131	0

Fault Extra Data:	00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Fault Description:	undefined

0.2	0	%I 00209	Loss of Device	-	02-11-2000 19:29:47

I/O Bus	Bus Address	Point Address	Group	Action	Category	Fault Type
1	n/a	0	3	2:Diagnostic	130	0

Fault Extra Data:	00 03 03 00 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Fault Description:	undefined

This automatic fault reporting can be disabled by setting the **Fault Table Entries** parameter in the module's software configuration to False. Regardless of whether the information appears in the fault tables, status information is available in the module's status references as explained on the next page.

For details on the effects of different types of faults on Series 90-30 PLC behavior, refer to the *Series 90-30 CPU Reference Manual*, GFK-0467. For PACSystems RX3i fault information, refer to the *PACSystems CPU Reference Manual,* GFK-2222.

Faults Reported to the I/O Fault Table

The I/O Fault Table will list the following DeviceNet module faults:

Loss of Device - This indicates a DeviceNet fault on a configured slave. For example, if the master detects a slave timeout and there is not a Loss of Network, the master logs a Loss of Device fault when the device bit in the Slave Status Table transitions from 1 to 0.

Addition of Device - This indicates that a device bit in the Slave Status Table has gone from 0 to 1, indicating that a slave device is active in the DeviceNet scan list.

Headend Fault - A headend fault affects the operation of the DeviceNet module or the network. Configuration errors, initialization errors, and bus faults are some possible headend faults. Because of the nature of these faults, some are not reported to the fault table. However, the PLC application program can detect many types of errors by monitoring the module's three status words. If a fault is detected, the program may be able to get more information about the fault by sending a Read Module Header COMMREQ to the module. See chapter 5, "Programmed Communications" for more information.

PLC Status References for a DeviceNet Module

A DeviceNet module automatically reports the three words of status information shown below to its configured PLC references (Network Status/Firmware ID). The application logic can monitor these references and take appropriate actions in response to specific changes. This can be done in conjunction with, or instead of, monitoring faults in the fault tables.

The format of the three status words is the same for both modules. For the DeviceNet Master Module, word 1 (Server Status) is meaningful only if the module is set up to operate as a slave (server) on a network with another master device.

The same information mapped to the status words can also be read directly from the module using COMMREQ #6, Get Status Information, as described in chapter 6, "Programmed Communications".

Word 1 — Server Status

	bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
byte 0	res.	AKS	CYC	COS	res.	ST	P	EX
byte 1	reserved				SERA	IDLE2	IDLE1	G3

Group 2 only I/O connections	AKS	Acknowledge suppress enabled
	CYC	Cyclic I/O connection allocated
	COS	Change-of-state I/O connection allocated
	ST	Bit Strobed I/O connection allocated
	P	Polled I/O connection allocated
Group 2 Explicit Connections	EX	Explicit connection allocated
Group 3 Connection	G3	At least one Group 3 (UCMM) connection allocated
Status Bits	IDLE1	Output area 1 receive idle status bit.
	IDLE2	Output area 2 receive idle status bit
	SERA	Server Explicit Request Available. Use Receive server explicit command to retrieve the request

Word 2 — CAN Network Status

	bit 7	*bit 6*	*bit 5*	*bit 4*	*bit 3*	*bit 2*	*bit 1*	*bit 0*
byte 0	ML	RO	TO	TA	A	BO	BW	OL
byte 1	SA	O5	O2	O1	RE	reserved	BP	ER

Application Specific Flags

SA	Scanner Active (at least one connection established)
O5	Online at 500 Kbaud
O2	Online at 250 Kbaud
O1	Online at 125 Kbaud
RE	Firmware is resetting so DeviceNet I/O data is not valid

Common Flags

BP	Bus power present (zero if power sense not supported)
ER	CAN communication error
ML	Message lost (CAN controller / receive ISR)
RO	Receive buffer overrun (host app. too slow emptying receive queue)
TO	Transmit failed due to timeout (flooded network)
TA	Transmit failed due to ack error (no other nodes connected)
A	Network activity detected (messages received or transmitted)
BO	Bus off (this node has been disconnected due to excessive errors)
BW	Bus warning (this node is experiencing a large number of errors)
OL	Online, CAN interface has been initialized

Word 3

Firmware ID, Minor revision:	In BCD four hex digits. For example, revision 1.10 = 01 10 hex.
Firmware ID, Major revision:	See above.

Device Status Bits for the DeviceNet Master Module

In addition to maintaining the three status words described previously, a DeviceNet Master Module maintains 64 device status bits in PLC memory. There is one bit for each potential device on the network. The memory type and starting offset of this status area in memory are configurable.

An individual device's status bit location equals the configured Start Address plus the Station Address (MAC ID) of the device. For example, if the status bits begin at %I00001, the status for the device at Station Address 5 is at %I00001 + 5, which is %I00006. The status bit for the DeviceNet Master Module also corresponds to its MAC ID. The value of DeviceNet Master Module's own status bit is always 0.

For other devices, if a status bit is 0, the device at the corresponding MAC ID is inactive (not configured or faulted). If a status bit is 1, the corresponding device is active. Transitions in these bits, from 0 to 1 and from 1 to 0, are reported as Addition of Module and Loss of Module faults in the I/O Fault Table.

The example status bits shown below represent a network with 25 devices assigned MAC IDs 0 to 24. The DeviceNet Master Module is assigned MAC ID #0 and the device with MAC ID #8 is faulted, so there are 0s in status bits %I00001, %I00009, and %I00026 through %I00064:

Example Device Status Bits																
1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	0	%I00001
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	%I00017
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	%I00033
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	%I00049

The same information about device status can be read directly from the module using COMMREQ #6, Get Status Information, as described in chapter 6, "Programmed Communications".

Chapter 6

Programmed Communications

This chapter explains how the application program can use Communications Requests to exchange information with a DeviceNet Master Module or DeviceNet Slave Module. COMMREQs are used for reading information from the DeviceNet module itself and for sending or replying to DeviceNet explicit messages over the network. See the following sections:

- ***COMMREQs for DeviceNet Modules***
- ***Using COMMREQs to Program Communications*** (an overview)
- ***COMMREQ Programming Requirements and Recommendations***
- ***Reading Identification, Status, and Error Information*** (from DeviceNet Master or Slave Module)
  - Module Type, Module ID, Module revision.
  - CAN Kernel identification and revision.
  - DeviceNet serial number.
  - Error codes.
  - CAN Network status.
- ***Getting the Status of a Network Slave*** (for DeviceNet Master Module only)
  - whether it is included in the master's list of configured devices
  - whether it is being scanned
  - its configuration error status
  - its connection 1 and connection 2 input states
- ***Getting Slave Status Information*** (of a DeviceNet Master or Slave Module)
  - whether the module is set up for slave operation
  - the module's output connection states
  - whether the module has sent a DeviceNet explicit message.
  - how the module's I/O messaging settings are configured.
- ***Getting Input Status from a Device*** (from DeviceNet Master Module)
  - the network activity status of each MAC ID on the network
  - the DeviceNet Master Module's own configured Network Settings.
  - the DeviceNet Master Module's current network status.
  - the DeviceNet Master Module's firmware ID.
- ***Sending a DeviceNet Explicit Message on the Network*** (DeviceNet Master)
- ***Reading and Responding to DeviceNet Explicit Messages from the Network*** (DeviceNet Slave Module or Master Module configured for slave operation)

For more general information about programming COMMREQs, please use the online help provided with the programming software.

COMMREQs for the DeviceNet Modules

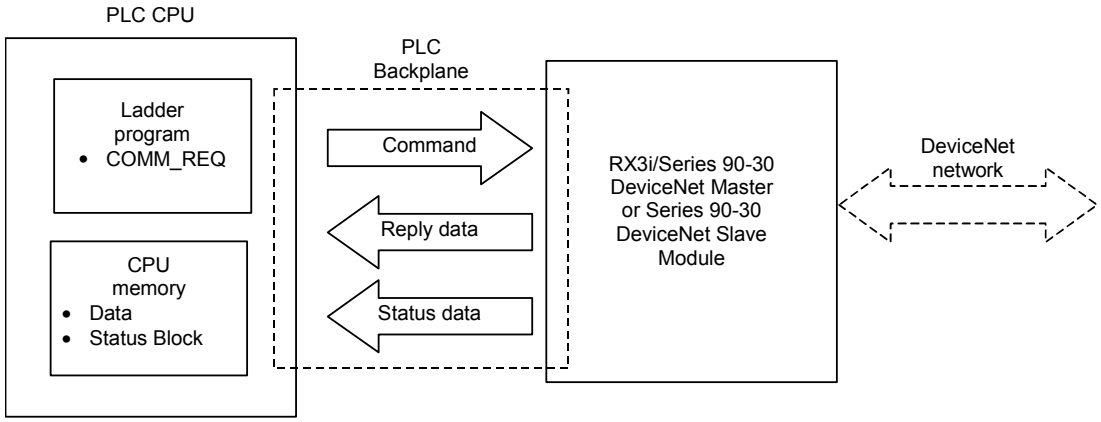
The DeviceNet Master Module and Slave Module can use the COMMREQs listed below. COMMREQs are described in detail on the following pages.

Command Code	Feature Support	Command Name	Description
1	Master	Send Device Explicit	Used to send and receive explicit messages to a server device over the DeviceNet network. Maximum 238 service data bytes.
2	Slave	Receive Server Explicit	Used to retrieve an explicit request sent to the server within the module.
3	Slave	Send Server Explicit	Used to service an explicit request sent to the server within the module e.g. reply to the client explicit request. Maximum 238 service data bytes.
4	Master	Get Detailed Device Status	Used to retrieve detailed status information for a specified node on the network.
5	Slave	Get Detailed Server Status	Used to retrieve detailed status about the server function of the module.
6	Master & Slave	Get Status Info	Used to retrieve the module status bytes that may also be configured to PLC I/O. An alternate method to access this data. Includes Device status, Server Status, CAN status and firmware ID.
7	Master	Send Device Explicit Extended	Extended version of Send Device Explicit (command code 1). Does not have the 238 byte limit and allows separate data area in PLC memory.
8	Slave	Send Server Explicit Extended	Extended version of Send Server Explicit (command code 3). Does not have the 238 byte limit and allows separate data area in PLC memory.
9	Master	Read Module Header	Used to retrieve the detailed status of the client within the module.

Using COMMREQs to Program Communications

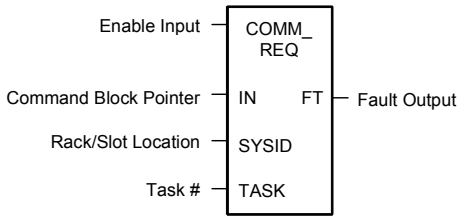
A Communications Request begins when a COMMREQ ladder instruction is activated in the PLC application program. The CPU sends the COMMREQ to an intelligent module in the PLC system, in this case, a DeviceNet Master or Slave module. The module receives the command and performs the requested function. Some COMMREQs for the DeviceNet Master or Slave are used to read or write module data. Other COMMREQs cause the DeviceNet module to execute DeviceNet messages on the network. In that case, the data in the COMMREQ itself includes the DeviceNet message in a format that will be recognized by the DeviceNet recipient(s).

At the conclusion of every COMMREQ, the PLC CPU reports the status of the operation into designated a location in CPU memory.



Format of the Communication Request Function

The Communication Request (COMMREQ) function has the following parameters:



Parameters of the COMMREQ Function

Input / Output	*Choices*	*Description*
Enable	Flow	When the function is energized, the communications request is performed. Must be Logic 1 to enable the COMMREQ instruction. The enabling logic should be a contact from a transition ("one-shot") coil.
IN	R, AI, AQ	Pointer to the first word of the Command Block. For example, %R00100 at IN would mean that the starting address of the Command Block is %R00100. The length of the Command Block depends on which command is being executed. Command Blocks for all the DeviceNet module commands are shown in this chapter.
SYSID	I, Q, M, T, G, R, AI, AQ, constant	Hex value that specifies the rack number (most significant byte) and slot number (least significant byte) of the target device. For example: rack 0 slot 4 = 00 04 rack 2 slot 9 = 02 09
TASK	1	Must be set to 1 for the DeviceNet Master or DeviceNet Slave module
FT	flow, none	The FT (fault) output can provide an output to optional logic that can verify successful completion of the Communications Request. The FT output is set High if: ▪ The specified target address is not present (for example, specifying Rack 1 when the system only uses Rack 0). ▪ The specified task number is not valid for the device. ▪ Data length is set to 0. The FT output can either be connected to another device, such as a set coil, or can be left open. Note: If the COMMREQ instruction is executed via a "one-shot" permissive as recommended then the FT output state will only be valid for one PLC sweep. The FT output can have these states:

Enable Input Status	*Does an Error Exist?*	*FT Output*
Active	No	Low
Active	Yes	High
Not active	No execution	Low

COMMREQ Programming Requirements and Recommendations

- Data must be placed in the Command Block before executing the COMMREQ instruction. For some commands, additional data areas must also be set up. Since the normal PLC sweep order is from top to bottom, this should be done ahead of the rung that contains the COMMREQ.
- If you use MOVE instructions to load values into Command Block registers, use a Word-type MOVE to load a hexadecimal number, and an Integer-type MOVE to load a decimal number.
- COMMREQ instructions should be enabled by a contact from a transition coil.
- The FT output is held False if the Enable Input is not active. If the COMMREQ is enabled by a transitional (one-shot) contact and a fault occurs, the FT output is High for only one PLC scan. To capture the fact that a fault occurred, you can program the fault output as a Set coil, which would not be automatically reset at the end of a scan. Additional logic would then be needed to reset the fault output coil after the fault is acknowledged and before the next execution of the COMMREQ.
- Programming a device, such as a Set Coil, on the FT output of the COMMREQ is optional. The FT output can be left open.
- COMMREQs must be sent sequentially. ***A new COMMREQ may be sent only after the previous COMMREQ indicates completion*** (either success or error) via the Status Block. When using more than one COMMREQ in a ladder program, verify that a previous COMMREQ executed successfully before executing another one. This can be done by checking the Status Word and the FT (Fault) output.

Checking the Execution of the COMMREQ

The FT output of the COMMREQ function goes high for certain faults and can be used for fault detection. In addition to using the FT output, the program should monitor the first COMMREQ Status Word, and use error message logic to generate text on an Operator Interface device. In this case, specific Status Word error codes might correspond to appropriate operator messages on a display screen.

The CPU places a value of 1 in the “State” COMMREQ Status word [status word 1] if the COMMREQ is processed normally. If any error condition is detected, a value greater than 1 is returned. For error detection in a ladder program, you can use a Greater Than (GT) compare instruction to determine if the value in the Status Word is negative (less than zero). If an error occurs, the GT instruction’s output (Q) will go high. A coil driven by the output can be used to enable fault handling or error reporting logic.

To dynamically check the Status Word, write a non-significant positive number, outside the range of 0x01-0x0D or 0x0E-0xFF, (0 is typically used) into all four of the Status Words each time before its associated COMMREQ is executed. If the instruction executes successfully, the CPU will write the number 1 in the “State” status word. If the number 1 is present, it means that the last COMMREQ executed successfully; the 1 is not from a previous execution.

When multiple COMMREQs are used in the application program scan, it is important to verify the successful completion of each one before enabling the next. Monitoring the Status Word is one way to accomplish that.

Corrective Actions for COMMREQ Errors

The type of corrective action to take in response to a COMMREQ error depends upon the application. If an error occurs during the startup or debugging stage of ladder development, you should check the COMMREQ parameters. The COMMREQ parameters should also be checked if an error occurs right after a program is modified.

When an error occurs in a proven application that has been running successfully, the problem is more likely to be hardware-related. Check the PLC fault tables for possible additional information when troubleshooting COMMREQ errors.

Using COMMREQs for DeviceNet Explicit Messaging

When using the COMMREQ commands to send or receive explicit messages it is important to follow the proper sequence of events. The DeviceNet explicit message interface is a half duplex interface, which means that only a single command can be in progress per configured device. The module always provides a response [in the COMMREQ status] to an explicit command. If the device does not respond, the module generates an error response on behalf of the unresponsive device so that the command / reply sequence at the PLC level does not get hung up. The appropriate sequence is always:

a.) load a complete COMMREQ command into the configured explicit message transmit buffer memory in the PLC specified by the COMMREQ command.

b.) Clear the COMMREQ status memory (4 words) location indicated by the command to a known, cleared state.

c.) Execute the COMMREQ command in ladder with a transitional permissive (one-shot).

d.) The PLC application logic must examine the COMMREQ status for success (State = 1), or failure (State > 1). If the command does not succeed, it always returns a failure status after an appropriate time delay. On failure state, determine the appropriate action: retry the command, issue alarm to operator, increment a fault counter, stop the PLC, or skip to next message.

e.) On a success state, optionally manipulate the reply data depending on command executed. You control where the reply data is located in PLC via the COMMREQ command. However it may be prudent to buffer the reply and copy it elsewhere in the PLC only after status is validated as success. A buffer is simply a location in PLC memory used to contain the reply data until it is validated to be a result of a sucessful command. When the COMMREQ status = 1, the data can be copied to its final location. In the event of a failure status, the reply data will not be the data you expect and will contain various values describing the failure mode. Any time the reply data will be tested as part of the application logic or used as a control value within an application, the buffer method is recommended.

f.) Start over at the beginning (a.) of the command sequence for the next explicit message.

Caution

It is very important that the PLC logic does not set its own reply timeouts on explicit messaging commands. If this is done it is possible that the command / reply sequence will get out of synch and prevent further proper generation of explicit messages.

The COMMREQ Status Block for DeviceNet Modules

The COMMREQ Status Block is four words long. Each status word is data type UINT. The status block can be located in any of the supported PLC memory areas.

The PLC application ***must*** set the Status Block contents to 0 before each COMMREQ.

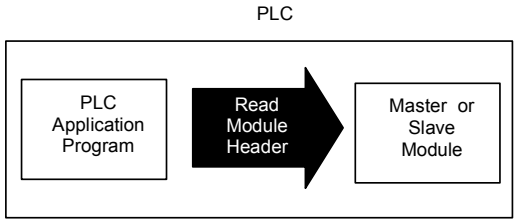
Word	Name	Description	
1	**State:** The state of the current COMMREQ request		
	0x00	Module has not yet processed the COMMREQ	
	0x01	Command Complete Note: This status does not necessarily mean success. Some commands have reply data that must also be checked.	
	0x02	Busy – Command is being processed and has not completed Note: It is not guaranteed that the status will transition to busy before complete or terminated.	
	0x03	Command Terminated – invalid command	
	0x04	Command Terminated – invalid command data	
	0x05	Command Terminated – not enough data	
	0x06	Reserved	
	0x07	Command Terminated – not enough memory in reply area The command did not specify sufficient PLC memory for the reply. Command will be ignored.	
	0x08	Command Terminated – command-specific error. See Error Code and Additional Code in the Status Block for more information.	
	0x09	Command Terminated – invalid COMMREQ	
	0x0A	Command Terminated – specific segment selector for COMMREQ reply is not supported	
	0x0B	Command Terminated – reply failed to write PLC memory	
	0x0C	Command Terminated - specific segment selector for COMMREQ data is not supported	
	0x0D	Command Terminated – failed to read PLC memory	
	0x0E to 0xFF	Reserved	
2	**Lost Command**	Command code of the last command lost. Set to 0 if no command was lost.	
3	**Error code:** Meaning Depends on the Command number		
	Command	Error Code	
		0	Reserved
	1,3,7,8	1	Explicit data too large for shared memory buffer. Additional Code word holds the real size of the shared memory buffer.
	1, 4, 7	2	Invalid MacID specified.
	1,2,3,7,8	3	Explicit connection not configured.
	2	4	Explicit request not available.
4	**Additional code:** for error reporting.		

Command Code 9: Reading Identification, Status, and Error Information

To read the following information from a DeviceNet Master or Slave module, use ***COMMREQ #9, Read Module Header***:

- Module Type, Module ID, Module revision.
- CAN Kernel identification and revision.
- DeviceNet serial number.
- Error codes for any existing faults.
- CAN Network status.

This command reads data from the module internal memory; no DeviceNet message is sent on the network.



Upon detecting an error, the PLC application program can send a ReadModuleHeader COMMREQ to the module. Unless the error prevents normal backplane operation, the module returns information about the fault in the reply data. Error codes are listed in this section.

Read Module Header, COMMREQ Example

This example COMMREQ does the following:

- Gets the Module Header Data
- Returns the COMMREQ Status Words to %R10-%R13
- Returns the Device Status to %R251-%R283.

Word #	*Dec*	*(Hex)*	*Description*
1	00004	(0004)	**Length of command Block:** Always 8 bytes (4 words) for command 9.
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status segment select:** Memory type for COMMREQ status words (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status memory offset:** status words starting address minus 1. (%R10 for this example)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00009	(0009)	**Command Code:** Read Module Header; command 9
8	00008	(0008)	**Reply segment select:** Memory type for the reply data (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply memory offset:** Offset within the memory type for the response minus 1. (%R251 for this example).
10	00065	(0041)	**Reply memory size:** Maximum size for the reply (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Maximum 2048 bytes. Note: For command 9 must be 130 bytes (65 words) or more, or an error will be returned in the COMMREQ status and the command will be ignored.

Read Module Header, Reply Data Format

<table>
<tr><th>Word</th><th>Description</th></tr>
<tr><td>1</td><td>Command Code. Echo of Command Code that this data block is replying to (0x0009)</td></tr>
<tr><td>2</td><td>Module Type. Contains "DN" (0x4E44) or "ER" (0x4552) if a fatal error is detected</td></tr>
<tr><td>3</td><td>Window size: Indicates host interface window size.
0 = 16K, 1 = 32K, 2 = 64K, 3=128K</td></tr>
<tr><td>4</td><td>Reserved</td></tr>
<tr><td>5</td><td>Kernel identification. 0x0001 = CAN 2.0A kernel</td></tr>
<tr><td>6</td><td>Kernel revision</td></tr>
<tr><td>7</td><td>Module ID, 0x0017 (DeviceNet module)</td></tr>
<tr><td>8</td><td>Module revision in binary coded decimal (BCD), 4 hex digits XX.XX
(i.e. rev 1.0 = 0x0100, rev 1.10 = 0x0110)</td></tr>
<tr><td>9,10</td><td>DeviceNet serial number</td></tr>
<tr><td>11 - 18</td><td>Card type, IC693DNM200 or IC693DNS201</td></tr>
<tr><td>19 - 22</td><td>Module serial number (i.e. "9409001")</td></tr>
<tr><td>23, 24</td><td>Reserved</td></tr>
<tr><td>25</td><td>Main Application Error Code. See the error code listings on the following pages.</td></tr>
<tr><td>26</td><td>CAN Network Status word.
<table>
<tr><th></th><th>bit 7</th><th>bit 6</th><th>bit 5</th><th>bit 4</th><th>bit 3</th><th>bit 2</th><th>bit 1</th><th>bit 0</th></tr>
<tr><td>byte 0</td><td>ML</td><td>RO</td><td>TO</td><td>TA</td><td>A</td><td>BO</td><td>BW</td><td>OL</td></tr>
<tr><td>byte 1</td><td>SA</td><td>O5</td><td>O2</td><td>O1</td><td>RE</td><td></td><td>BP</td><td>ER</td></tr>
</table>
Application –Specific Flags
SA Scanner Active (at least one connection established)
O5 Online at 500 Kbaud
O2 Online at 250 Kbaud
O1 Online at 125 Kbaud
RE Firmware is performing DeviceNet reset, I/O data is not valid
Common Flags
BP Bus power present (zero if power sense not supported)
ER CAN communication error
ML Message lost (CAN controller / receive ISR)
RO Receive buffer overrun (host app. too slow emptying receive queue)
TO Transmit failed due to timeout (flooded network)
TA Transmit failed due to ack error (no other nodes connected)
A Network activity detected (messages received or transmitted)
BO Bus off (this node has been disconnected due to excessive errors)
BW Bus warning (this node is experiencing a large number of errors)
OL Online, CAN interface has been initialized</td></tr>
<tr><td>27</td><td>CAN transmit counter. Incremented when messages are submitted to the CAN controller.</td></tr>
<tr><td>28</td><td>CAN acknowledgment error counter. Increments if a transmit message is terminated due to lack of acknowledgment from other stations. When this counter is incremented, the CAN transmit counter (word 27) is decremented to compensate for a message not actually transmitted.</td></tr>
<tr><td>29</td><td>CAN receive counter. Increments when messages are received. Messages that fail the receive filter still increment this counter.</td></tr>
<tr><td>30</td><td>CAN communication error counter. Increments if a CAN frame error is detected.</td></tr>
<tr><td>31</td><td>CAN lost messages counter. Increments if a CAN message is received before the previous message is placed into the receive queue.</td></tr>
<tr><td>32</td><td>CAN receive queue overrun counter. Increments if a CAN message is lost due to a full receive queue.</td></tr>
<tr><td>33</td><td>Additional Application Error Code. See the error code listings on the following pages.</td></tr>
<tr><td>34 - 63</td><td>When Module Type in word 2 is "DN", contains the module identification string. For example:
“DeviceNet Module 1.00.00\n(C) 2002 GE Fanuc Automation.”
The format is: major rev.minor rev.build
When Module Type is "ER”, contains the kernel error string.</td></tr>
<tr><td>64</td><td>Major Tick Interval (equivalent of system time base)</td></tr>
<tr><td>65</td><td>Number of minor ticks per major tick interval</td></tr>
</table>

Headend Fault Codes in the Module Header

After a headend fault [word 2 of the Read module header, reply data = "DN" (0x444E)], the Main Error Code [word 25] and Additional Code [word 33] fields of the reply data describe the fault. A zero value in the main error code word indicates no error.

Category Name	*Main Code*	*Error Name*	*Additional Code*	*Description*
No Error	0x0000	-	0x0000	Zero in Main Code indicates no error
Config File Error	0x0001	Unknown Version	0x0001	Incorrect / unsupported version
Init File Error	0x0002	Unknown Version	0x0001	Incorrect / unsupported version
		Unknown Header Id	0x0002	Init file's header ID not recognized or invalid
		Invalid Block Definition Count	0x0003	Block's definition count is invalid
		Unknown Block Type	0x0004	Block type not recognized
		Invalid Block Checksum	0x0005	Block's checksum is invalid
		Invalid Shared Memory Offset	0x0006	Shared memory offset not in the 0x1000 to 0x3FFF range.
		Unknown9030IoType	0x0007	I/O Type code not recognized
		Invalid Mac Id	0x0008	Mac Id in the Data Pointer list not in the range of 0 to 64, inclusive, or 255.
		Unknown Memory Area Type	0x0009	Memory area type code not recognized
		Invalid Block Size	0x000A	Size of block in INIT file did not match what was expected, or larger than the maximum size supported by the firmware.
		Invalid Block Offset	0x000B	Offset in Block Definition Record points beyond maximum INIT file size.
		Duplicate Block	0x000C	Block of with same type code already exists in INIT file.
		Data Pointer Out Of Range	0x000D	Data pointer refers to a location outside of shared memory.
		Missing Block	0x000E	One or more of required blocks 1, 3, and 4 are missing from the INIT file.
Add Device Error	0x0003	Duplicate Device	0x0005	Device already in scan list.
		Invalid Shared Memory Offset	0x000D	Shared memory offset not in the 0x1000 to 0x3FFF range.
		Invalid Connection Flags	0x000F	The combination of bits in the Flags field is invalid.
		Invalid Explicit Buffer Size	0x0010	Explicit buffer size is invalid.
		Invalid Strobe Buffer Size	0x0011	Strobe buffer size is invalid. Note that the output size must be 1.
		Invalid Path Buffer	0x0012	Path buffer is not initialized.
Online Error	0x0004	Invalid Mac Id	0x0002	MacId in the Server Config block is not in the range of 0 to 63 inclusive.
		Invalid Baud Rate	0x0003	Baud rate not set to 0, 1, or 2 (i.e. 125K, 250K, or 500K)
		Duplicate Mac Id Failure	0x0004	DUP MacId check failed while attempting to go online.
		Bus Not Offline	0x0009	Bus is already online. Internal firmware error; report to manufacturer.
		Bus Off	0x000E	Bus fault detected.
		Invalid Connection Flags	0x000F	Combination of bits in the Flags field of the Server Config block is invalid.
		Invalid Explicit Buffer Size	0x0010	Explicit buffer size is invalid.
		Invalid Strobe Buffer Size	0x0011	Strobe buffer size is invalid. Note that the output size must be 1.
		Invalid Path Buffer	0x0012	Path buffer is not initialized.
		Ack Fault	0x0013	No CAN acknowledge received during Duplicate MacId Sequence.
Start Scan	0x0005	Bus Offline	0x0007	Bus is not online yet
		Scanner Running	0x000A	Scanner is already started
		Scanner Stopping	0x000C	Scanner is stopping

Fatal Error Codes in the Module Header

If the module is capable of executing and reporting an error after a fatal error, the Module Type field of the Module Header data contains the value “ER” (0x4552).

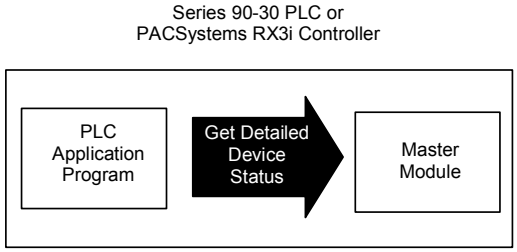
- All DeviceNet network activity, if active, stops. Proper DeviceNet shutdown messages are not guaranteed.
- The Module Status LED turns RED and the Network status LED goes off.
- The entire Module Header (in ACSII hex code) and the message field of the Module Header (in string form) are continuously sent over the RS232 port at 19200 baud, N81.
- Status flags and information update to cause a PLC fault table entry.
- Backplane communication continues (possibly limited) in order to service the PLC COMMREQ (command 9) to read header data.

For specific errors listed below that do not have any source code information available, the value 0xFFFF is placed in the Main Code and one of the listed error numbers is placed in the Additional Code field. Fatal errors that can report source code location store the source file number in Main Code and the source line number in Additional Code.

Add'l Error Code	*Error*	*Description*
1	RAM data test failed	An error occurred during testing of the RAM data bus. Return the module for repair.
2	RAM address test failed	An error occurred during testing of the RAM address bus. Return the module for repair.
3	RAM A16 address test failed	An error occurred during testing of the RAM A16 signal. Return the module for repair.
4	RAM A17 address test failed	An error occurred during testing of the RAM A17 signal. Return the module for repair.
5	Module checksum is invalid	The most likely cause of this error is an undetected memory failure. If this error occurs with more than one application module, the module should be returned for repair.
6	CAN reset flag failed to clear	An error occurred testing the CAN controller. Return the module for repair.
7	CAN data test failed	An error occurred testing the CAN controller data bus. Return the module for repair.
8	CAN address test failed	An error occurred testing the CAN controller address bus. Return the module for repair.
9	Invalid NVRAM data	The module's non-volatile memory contains invalid information. Return the module for repair.
10	Execution permission denied	This module has not been configured to execute the application module. Contact the vendor of the application module for assistance.
11	Application initialization error	An error occurred initializing the application module. Report this condition to the vendor of the application module.
12	Unknown application initialization code	An error occurred initializing the application module. Report this condition to the vendor of the application module.
13	Application terminated	The application module terminated (abnormal condition). Report this condition to the vendor of the application module.
14	Application fatal error	A fatal headend fault occurred. Report this condition to the vendor of the application module.
15 - 21	XXX interrupt	An unexpected interrupt was detected. This error should be reported to the vendor of the application module. Make note of the circumstances that caused this error.
22	Event queue overflow	This error should be reported to the vendor of the application module. Make note of the circumstances that caused this error.
23	Nested user timer interrupt	
24	Invalid CAN interrupt	
25	Nested system timer interrupt	
26	Imperfect interrupt	This error should be reported to the vendor of the application module. Make note of the circumstances that caused this error. This error is caused by an incorrectly generated interrupt from the host bus adapter to the module.
27	Stack Overflow	This error should be reported to the vendor of the application module. Make note of the circumstances that caused this error.
99	Unexpected condition encountered	A fatal headend fault occurred. Report this condition to the vendor of the application module.

Command Code 4: Getting the Status of a Network Device

To retrieve from the DeviceNet Master Module a set of detailed status information about a selected network device, use ***COMMREQ #4, Get Detailed Device Status***.



This command obtains the following information for the specified node:

- whether it is included in the master's list of configured devices
- whether it is being scanned
- configuration error status (invalid vendor id, device type, product code, I/O connections, etc)
- its connection 1 and connection 2 input states

This function is internal to the PLC; it does not generate a DeviceNet network message.

Get Detailed Device Status, Example COMMREQ

This example COMMREQ does the following:

- Gets the Device Status of the slave with MAC ID #4 from the DeviceNet Master Module.
- Returns the COMMREQ Status to %R10-%R13
- Returns the Device Status to %R251-%R260.

Word #	*Dec*	*(Hex)*	*Description*
1	00005	(0005)	**Length of command Data Block:** For the Get Detailed Device Status COMMREQ, always 5
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status segment select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status memory offset:** COMMREQ status words address minus 1 (%R10)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00004	(0004)	**Command code:** Get Detailed Device Status command number 4
8	00008	(0008)	**Reply segment select:** Memory type for the reply data (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply memory offset:** Offset within the memory type for the response minus 1. For this example %R251. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22)
10	00009	(0009)	**Reply memory size:** Maximum size for the reply (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Maximum 2048 bytes. Note: For command 9 must be 18 bytes (9 words) or more, or an error will be returned in the COMMREQ status and the command will be ignored.
11	00004	(0004)	**MAC ID:** of the network device. For this example, 4.

Get Detailed Device Status, Reply Data Format

Upon receiving COMMREQ 4 from the PLC CPU, the DeviceNet Master Module generates a reply containing the status data it currently has stored for the specified MAC ID.

<table>
<tr><th>Word #</th><th colspan="4">Description</th></tr>
<tr><td>1</td><td colspan="4">Command number that this data block is replying to. (4)</td></tr>
<tr><td rowspan="14">2
low byte</td><td colspan="4">Status Code: Number indicating the status of the client connection to the device.</td></tr>
<tr><td>Status</td><td>Meaning</td><td>Status</td><td>Meaning</td></tr>
<tr><td>0x 00</td><td>Device not in device list</td><td>0x 0D</td><td>Invalid I/O connection 1 input size</td></tr>
<tr><td>0x 01</td><td>Device idle (not being scanned)</td><td>0x 0E</td><td>Error reading I/O connection 1 input size</td></tr>
<tr><td>0x 02</td><td>Device being scanned</td><td>0x 0F</td><td>Invalid I/O connection 1 output size</td></tr>
<tr><td>0x 03</td><td>Device timed-out</td><td>0x 10</td><td>Error reading I/O connection 1 output size</td></tr>
<tr><td>0x 04</td><td>UCMM connection error</td><td>0x 11</td><td>Invalid I/O connection 2 input size</td></tr>
<tr><td>0x 05</td><td>Master/Slave connection set is busy</td><td>0x 12</td><td>Error reading I/O connection 2 input size</td></tr>
<tr><td>0x 06</td><td>Error allocating Master/Slave connection set</td><td>0x 13</td><td>Invalid I/O connection 2 output size</td></tr>
<tr><td>0x 07</td><td>Invalid vendor id</td><td>0x 14</td><td>Error reading I/O connection 2 output size</td></tr>
<tr><td>0x 08</td><td>Error reading vendor id</td><td>0x 15</td><td>Error setting I/O connection 1 packet rate</td></tr>
<tr><td>0x 09</td><td>Invalid device type</td><td>0x 16</td><td>Error setting I/O connection 2 packet rate</td></tr>
<tr><td>0x 0A</td><td>Error reading device type</td><td>0x 17</td><td>M/S connection set sync fault</td></tr>
<tr><td>0x 0B</td><td>Invalid product code</td><td>0x 18</td><td>Error setting Production Inhibit Time</td></tr>
<tr><td>0x 0C</td><td>Error reading product code</td><td>0x 19 - FF</td><td>Reserved</td></tr>
<tr><td rowspan="5">2
high
byte</td><td colspan="4">Status flags: Bits indicating the connection states of the slave's connection 1 and connection 2 inputs.</td></tr>
<tr><td>bits 0-4</td><td colspan="3">Reserved, should be ignored</td></tr>
<tr><td>bit 5</td><td colspan="3">1 = Input area 1 receive idle condition</td></tr>
<tr><td>bit 6</td><td colspan="3">1 = Input area 2 receive idle condition</td></tr>
<tr><td>bit 7</td><td colspan="3">Reserved, should be ignored</td></tr>
<tr><td>3 to 9</td><td colspan="4">Reserved, should be ignored</td></tr>
</table>

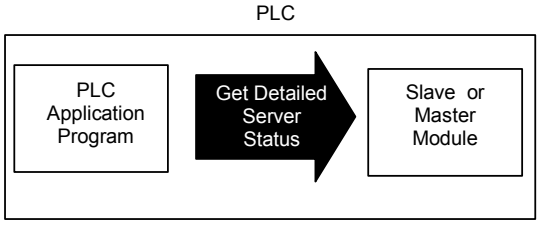
Command Code 5: Getting Status Information of a DeviceNet Slave Module or the Server Function of a Master Module

To retrieve status information about a DeviceNet Slave Module, or a DeviceNet Master module operating in server mode, use ***COMMREQ #5, Get Detailed Server Status***.

This command supplies the following information about the module:

- whether the module is set up for slave operation (its network settings are configured)
- the module's output connection states
- whether the module has sent a DeviceNet explicit message (previously commanded by a Send Server Response COMMREQ).
- how the module's I/O messaging settings are configured.

This function is internal to the PLC; it does not generate a DeviceNet network message.



Get Detailed Server Status, COMMREQ Example

In this example, the application program sends a Get Detailed Served Status COMMREQ to a DeviceNet Master Module that is configured for slave operation.

This COMMREQ specifies the command number and command length, and sets up memory locations for both the COMMREQ status and the reply message.

Word #	*Dec*	*Hex*	*Description*
1	00004	(0004)	**Length of command Data Block.** Always 4 words for this command.
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status segment select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status memory offset:** COMMREQ status words address minus 1 (%R10 for this example)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00005	(0005)	**Command code:** Get Detailed Server Status command (5)
8	00008	(0008)	**Reply segment select:** Memory type for the reply data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply memory offset:** Offset within the memory type for the reply minus 1. For this example, it is %R251.
10	00009	(0009)	**Reply memory size:** Maximum size for the reply (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Maximum 2048 bytes. Note: For command 5 must be 18 bytes (9 words) or more, or an error is returned in the COMMREQ status and the command is ignored.

Get Detailed Server Status, Reply Data Format

The response to a Get Detailed Server Status COMMREQ supplies details of the module's configured Network Settings. It also shows whether the module has sent (on the DeviceNet network) a previously-commanded Send Server Explicit message.

Word #	*Dec/Bin*	*Hex*	*Description*	
1	00005	(0005)	Command number that this data block is replying to. (5)	
2 low byte	00000001	(01)	Indicates whether the module is set up for slave operation (Network Settings configured). For this example, the module is being scanned.	
			0x00	Idle (Group 2 master/slave connection is not allocated, the slave server is not active, no master is scanning).
			0x01	Active (Group 2 master/slave connection allocated, the slave server is active and is being scanned by a master device).
			0x02- 0xFF	Reserved, these bits should be ignored.
2 high byte	Bits indicating the various connection states. In this example, the module's output 1 and 2 connections are both configured for receive idle, and it has an UCMM connection.			
	11100000	(E0)	*bits 0 - 4*	Reserved, these bits should be ignored.
			bit 5	1 = Output connection 1 receive idle condition
			bit 6	1 = Output connection 2 receive idle condition
			bit 7	1 = Group 3 UCMM connection(s) allocated.
3	00000	(0000)	Reserved, these bits should be ignored.	
4 low byte	Bits indicating explicit message status since the last Get Detailed Server Status. These bits are automatically cleared by this COMMREQ in preparation for the next call. For this example, the module has sent the explicit message response on the network.			
	00001	(0001)	*bit 0*	1 = Explicit response sent. Set when the scanner has submitted the explicit response from a Send Server Explicit message, for transmission on the network.
			bits 1 - 7: Reserved	
4 high byte	Bits showing the configured features of the module. For this example, the DeviceNet module slave server is set up for explicit messaging and polled I/O operation.			
	00000011	(03)	*bit 0*	1 = Explicit connection allocated. This bit is always on, regardless of whether the explicit messaging is turned on or off.
			bit 1	1 = Polled I/O connection allocated
			bit 2	1 = Bit-strobed I/O connection allocated
			bit 3	Not used
			bit 4	1 = Change of State I/O connection allocated
			bit 5	1 = Cyclic I/O connection allocated
			bit 6	1 = Acknowledge Suppress Enabled
			bit 7	Not used
5 to 9			Reserved, these bits should be ignored.	

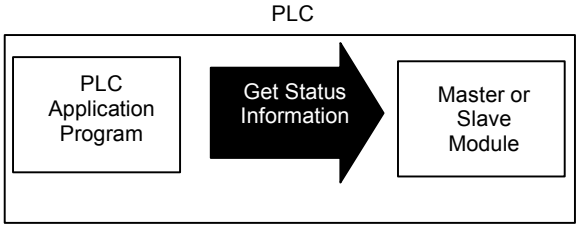
Command Code 6: Getting Input Status from a Device

To read the information normally mapped to a DeviceNet Module's 3 status words, plus all the information normally mapped to the DeviceNet Master Module's 64 device status bits, use ***COMMREQ #6, Get Status Information***. COMMREQ #6 is alternate way to access this information.

The module responds to the command with the following information:

- the network activity status of each MAC ID on the network
- the module's own configured Network Settings.
- the module's current network status.
- the module's firmware ID.

The information read by this command comes directly from the module; this command does not generate a DeviceNet message.



Get Status Information, COMMREQ Example

This example COMMREQ does the following:

- Gets status information from the DeviceNet master.
- Returns the COMMREQ Status Words to %R10-%R13
- Returns the Device Status to %R251-%R260.

Word #	*Dec*	*(Hex)*	*Description*
1	00004	(0004)	**Length of command Data Block:** For Get Status Information, the length is 4 words (8 bytes).
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status segment select:** Memory type of COMMREQ status words (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status memory offset:** COMMREQ status words start address minus 1 (%R10 for this example)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00006	(0006)	**Command code:** Get Status Info command number (6)
8	00008	(0008)	**Reply segment select:** Memory type for the reply data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply memory offset:** Offset within the memory type for the reply (0-based). For this example, it is %R251. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22).
10	00008	(0008)	**Reply memory size:** Maximum size for the reply (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Maximum 2048 bytes. Note: For command 6 must be16 bytes (8 words) or more, or an error will be returned in the COMMREQ status and the command will be ignored.

Get Status Information, Reply Data Format

<table>
<tr><th>Word #</th><th>Description</th></tr>
<tr><td>1</td><td>Command code that this data block is replying to. (6)</td></tr>
<tr><td>2 - 5</td><td>Device Status. Each bit corresponds to an individual device MAC ID. The state of that bit indicates the device's status:
0 = Device is not active (not configured, faulted, etc…)
1 = Device is active, being scanned
For the master's own MAC ID, the status bit is always 0.
<table>
<tr><th></th><th>bit 7</th><th>bit 6</th><th>bit 5</th><th>bit 4</th><th>bit 3</th><th>bit 2</th><th>bit 1</th><th>bit 0</th></tr>
<tr><td>byte 0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>byte 1</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>byte 2</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td>byte 3</td><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td>byte 4</td><td>39</td><td>38</td><td>37</td><td>36</td><td>35</td><td>34</td><td>33</td><td>32</td></tr>
<tr><td>byte 5</td><td>47</td><td>46</td><td>45</td><td>44</td><td>43</td><td>42</td><td>41</td><td>40</td></tr>
<tr><td>byte 6</td><td>55</td><td>54</td><td>53</td><td>52</td><td>51</td><td>50</td><td>49</td><td>48</td></tr>
<tr><td>byte 7</td><td>63</td><td>62</td><td>61</td><td>60</td><td>59</td><td>58</td><td>57</td><td>56</td></tr>
</table></td></tr>
<tr><td>6</td><td>Server Status
<table>
<tr><th></th><th>bit 7</th><th>bit 6</th><th>bit 5</th><th>bit 4</th><th>bit 3</th><th>bit 2</th><th>bit 1</th><th>bit 0</th></tr>
<tr><td>byte 0</td><td>res.</td><td>AKS</td><td>CYC</td><td>COS</td><td>res.</td><td>ST</td><td>P</td><td>EX</td></tr>
<tr><td colspan="9">All these bits are turned off if the COMMREQ is sent to a master device.</td></tr>
<tr><td>byte 1</td><td colspan="5">reserved</td><td>SERA</td><td>IDLE2</td><td>IDLE1</td><td>G3</td></tr>
</table>
Group 2 only I/O connections
AKS Acknowledge suppress enabled
CYC Cyclic I/O connection allocated
COS Change-of-state I/O connection allocated
ST Bit Strobed I/O connection allocated
P Polled I/O connection allocated
Group 2 Explicit Connections EX Explicit connection allocated
Group 3 Connection G3 At least one Group 3 (UCMM) connection allocated
Status Bits IDLE1 Output area 1 receive idle status bit.
IDLE2 Output area 2 receive idle status bit
SERA Server Explicit Request Available. Use Receive server explicit command to retrieve the request</td></tr>
<tr><td>7</td><td>CAN Network Status.
<table>
<tr><th></th><th>bit 7</th><th>bit 6</th><th>bit 5</th><th>bit 4</th><th>bit 3</th><th>bit 2</th><th>bit 1</th><th>bit 0</th></tr>
<tr><td>byte 0</td><td>ML</td><td>RO</td><td>TO</td><td>TA</td><td>A</td><td>BO</td><td>BW</td><td>OL</td></tr>
<tr><td>byte 1</td><td>SA</td><td>O5</td><td>O2</td><td>O1</td><td>RE</td><td>reserved</td><td>BP</td><td>ER</td></tr>
</table>
Application Specific Flags
SA Scanner Active (at least one connection established)
O5 Online at 500 Kbaud
O2 Online at 250 Kbaud
O1 Online at 125 Kbaud
RE Firmware is resetting so DeviceNet I/O data is not valid
Common Flags
BP Bus power present (zero if power sense not supported)
ER CAN communication error
ML Message lost (CAN controller / receive ISR)
RO Receive buffer overrun (host app. too slow emptying receive queue)
TO Transmit failed due to timeout (flooded network)
TA Transmit failed due to ack error (no other nodes connected)
A Network activity detected (messages received or transmitted)
BO Bus off (this node has been disconnected due to excessive errors)
BW Bus warning (this node is experiencing a large number of errors)
OL Online, CAN interface has been initialized</td></tr>
<tr><td>8</td><td>Firmware ID, Minor revision:</td><td>In BCD four hex digits. For example, revision 1.10 = 01 10 hex.</td></tr>
<tr><td></td><td>Firmware ID, Major revision:</td><td>See above.</td></tr>
</table>

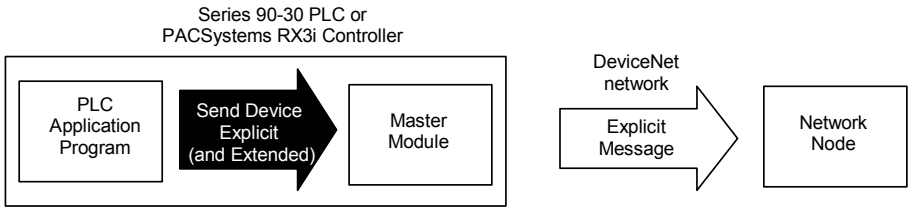
Command Codes 1 & 7: Sending a DeviceNet Explicit Message on the Network

To send a DeviceNet explicit message up to 238 bytes long to a specified device on the network, use ***COMMREQ #1, Send Device Explicit.*** The reply data is limited to 2048 bytes maximum.

To send more than 238 bytes of data or to use a separate data memory area in the PLC, use ***COMMREQ #7, Send Device Explicit Extended***. The reply is limited to 2048 bytes maximum.

Both of these COMMREQs command the DeviceNet Master Module to send a DeviceNet explicit message on the network. The addressed device must be configured for an explicit message connection in the configuration of the DeviceNet Master Module and sufficient buffer memory must be configured to contain the largest message produced by the COMMREQ or the largest reply produced by the device.

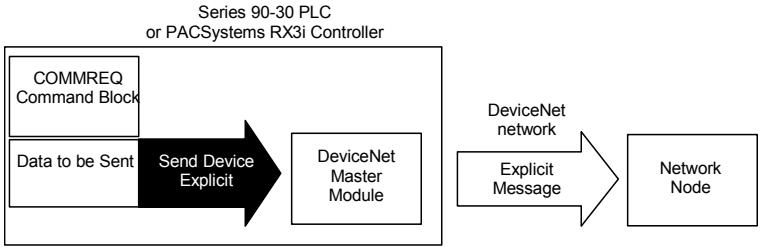
It is important to configure the Series 90-30, DeviceNet slave device (Network Settings, explicit messages tab) of the device addressed, with enough “explicit message size” bytes to contain at a minimum, all the data specified in the data memory size (word 12 of COMMREQ 7) minus the number of bytes skipped in the “data byte offset (command 7, data word 6 of the service data header). For the command 7 example that follows: the device at network address 4 (the S2K controller) must have a minimum of 141 bytes (142 bytes data block minus 1 skipped byte) explicit message size configured.



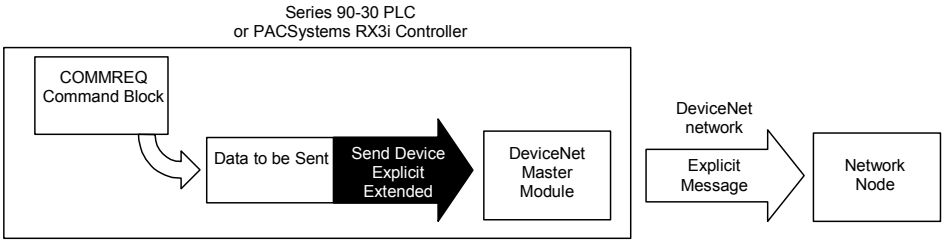
If the device was not configured for explicit messaging or if the number of bytes configured is not enough for the command, the COMMREQ fails with a code of 8 in the COMMREQ Status Word.

Difference Between Send Device Explicit and Send Device Explicit Extended

The difference between Send Device Explicit and Send Device Explicit Extended is how they store the data that will be sent in PLC memory. For Send Device Explicit, the data to be sent is located in the same memory area as the COMMREQ command block.



For Send Device Explicit Extended, the data to be sent is located in a separate memory area, which is indicated by a pointer in the COMMREQ command block. This makes it possible to store and send more data or to have the data separate from the command memory.



Send Device Explicit, COMMREQ Example

The Send Device Explicit COMMREQ command block contains the data to be sent in the explicit message (the data may optionally be offset from the end of the command block as explained below). For this example, there are multiple channels in the VersaPoint analog module to configure. The application program can repeat the message with a different instance [another channel]. Having the PLC application check the COMMREQ status is important even if there is only one COMMREQ, to be sure it has worked. The VersaPoint DeviceNet NIU (network interface unit) may be off-line for example. When sequencing multiple commands to the same device (MAC ID) it is critical to test for successful command completion prior to executing a subsequent command.

This example COMMREQ does the following:

- Sends an explicit message to device # 4 (a VersaPoint DeviceNet NIU)
- Returns the COMMREQ Status Words to %R10-%R13
- Sets Analog Input 1 to the 4-20mA range.

Word #	Dec	(Hex)	Description
1	00012	(000C)	**Length of command:** Length of the command block for this COMMREQ. For the Send Device Explicit (command 1) the command length is 10 words plus the number of words of Service Data. The COMMREQ header (words 1–6) is not counted in the command length. Note: Service Data is in bytes, divide by 2 and round up for words. Service data length will vary depending on message executed; consult vendor documentation of the addressed server device. For this example: 12 words = Service Data is 3 bytes (rounded up to 2 Words) + 10 words command length.
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status Segment Select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status Memory Offset:** COMMREQ status words start address minus 1. (%R10 for this example)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00001	(0001)	**Command Code:** Send Device Explicit command number (1)
8	00008	(0008)	**Reply Segment Select:** Memory type for the reply data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply Memory Offset:** Offset within the memory type for the reply minus 1. Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22. For this example, it is %R251.
10	00006	(0006)	**Reply Memory Size:** Maximum size required to hold the reply to the command: in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22) For command 1 the size must be 10 bytes (5 words) or more, or an error will be reported in the COMMREQ status word and the request will be ignored. Note: The size needed for the reply depends on the service used and the instance accessed. Consult the server device documentation. Add 10 bytes (5 words) to the server reply data for the reply header. The reply memory size can be larger than the reply data of a particular message it must not be smaller.
11	00004	(0004)	**MAC ID:** of the device to send the message to (0 - 63). For this example, the VersaPoint Network Interface Unit uses MAC ID #4.
12	00002	(0002)	**Service Data Size:** Number of Service Data bytes being sent. This needs to be determined from the documentation of the DeviceNet server to which the message is being sent. For the example, 2 bytes = 1 attribute byte + 1 bytes data. Note: For service codes 0x10 or 0x0E the attribute byte is contained in the service data at byte zero.
13	00016	(0010)	**DeviceNet service code:** See the vendor documentation for the server device. In this example, the Service Code for the VersaPoint DeviceNet NIU is 0x10 (Set Attribute Single Service) to write data. Another service code often used is 0x0E (Get Attribute Single Service) to read data.
14	00010	(000A)	**Class/Object:** The object class to which this is requested. See the vendor documentation for the server device. For this example, the object class is 0x0A (Analog Input Point Object).
15	00001	(0001)	**Instance:** The specific instance of the object class to which this request is directed. See vendor documentation for the server device. For the example the instance represents which VersaPoint analog channel to set.
16	00001	(0001)	**Service Data Byte Offset:** If the offset is 0, then the service data is located immediately after this data word in memory (at word 17, see below). The value entered here is the number of bytes between this word and the beginning of the service data. For example, if the offset were 2, then two bytes would be "skipped" and the service data would begin at word 18.
17	1792	(00)	**Skipped byte(s):** This byte is skipped because the data byte offset in word 16 is a “1” for this example. Multiple bytes may be skipped. Data in skipped bytes is ignored.
		(07)	**Service Data byte 0, Attribute:** Attribute is used in service code 0x10 and 0x0E messages. The attribute is a one-byte field always at byte zero of the service data when used. The “Attribute” field is not used by other message services. This byte is the actual beginning of the service data since the data byte offset caused a one-byte skip. For the example attribute 7 is the VersaPoint, Analog Input Point Object, Range setting.
18	00003	(0003)	**Service Data:** Offset of the start of this data depends on entry for Service Data Byte Offset. Service data to is limited to 238 bytes maximum for command 1. For the example “Range” 3 is the vendor code for the VersaPoint Analog Input 4-20ma setting and the data type is USINT (2 bytes). Note: It is important to know the type of the data to properly calculate the length setting of word 1 and word 12 of the COMMREQ.
19 to end			**Service Data:** Additional service data as required by the message. In the example this is unused space.

Send Device Explicit Extended, COMMREQ Example

The Send Device Explicit Extended COMMREQ command block contains a pointer to the data to be sent in the explicit message. The programmer can use this functionality to point to different stored messages without recalculating command length each time. Command 7 additionally avoids the 238-byte service data limit of command 1 by increasing the maximum size for the service data. This command is only valid to a Master Module.

This example COMMREQ does the following:

- Sends an explicit message to Mac ID 4 (a GE Fanuc S2K DeviceNet Motion Controller)
- Returns the COMMREQ Status Words to %R10-%R13
- Sets (writes) an array of data (32 DINT) variables to the S2K integer memory (VI registers).

	Word #	*Value*	*Description*
COMMREQ Header	1	00007	**Command Length:** Length of the command block for the Send Device Explicit Extended command. Always 7 words.
	2	00000	Always 0 (no-wait mode request)
	3	00008	**Status Memory Select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
	4	00009	**Status Memory Offset:** COMMREQ status words start address minus 1 (%R10 for this example)
	5		Reserved
	6		Reserved
COMMREQ Command	7	00007	**Command Code:** Send Device Explicit Extended command number (7)
	8	00008	**Reply Segment Select:** Memory type for the reply data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
	9	00250	**Reply Memory Offset:** Offset within the memory type for the reply minus 1. For this example, it is %R251. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22).
	10	00005	**Reply Memory Size:** Maximum size required to hold the reply for the command: (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Add 10 bytes to expected reply size. Note: must be 10 bytes (5 words) or more, or an error will be reported in the COMMREQ status word and the request will be ignored. Actual length needed will vary depending on which message is sent; consult vendor information for the target device. Maximum 2048 bytes.
	11	00008	**Data Segment Select:** Memory type for the service data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
	12	00300	**Data Memory Offset:** Offset within the specified memory type for the service data start address minus 1. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22). For this example, it is %R301.
	13	00071	**Data Memory Size:** Size of the data to be sent, in units of the selected type (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Must be large enough to contain the entire explicit data block. The entire data block calculation is; the service data header 12 bytes (6 words) + skipped bytes (specified in word 6 of the service data header + the service data. Note: It is important to know the type of the data used in the service to calculate the minimum length accurately. The attribute byte when used is always byte 0 of the service data and must be added to the data length. Round size up as needed. For this example we have 71 words; 6 service data header words + 1 skipped byte + 1 attribute byte + 32 DINT data (64 words) service data.

Send Device Explicit Extended, Data Block Format

The following data must be placed in the PLC memory location specifed in the command by the data memory offset.

One use of the data byte offset (see below) would be to “point “ to a start location within a large array of data in the PLC memory. In the following example the data byte offset is used to maintain word boundary location of the data within the PLC memory even though we require the service data to contain the attribute value.

	Word #	*(Hex)*	*Description*
Service Data Header	1	(0004)	**MAC ID:** Address of the device to send the message to (0 - 63).
	2	(0081)	**Number of Service Data bytes:** This needs to be determined from the vendor documentation of the DeviceNet server to which the message is being executed. For the example Service Data 0x81 (129 bytes) = 1 byte attribute + 128 bytes (32 DINT) of data.
	3	(0010)	**DeviceNet service code:** See the vendor documentation for the server device. In this example, the Service is 0x10 (Set Attribute Single Service) to write data.
	4	(0004)	**Object Class:** to which this is requested. See the documentation for the server device. For this example, the object class is 0x04 (S2K Assembly Object).
	5	(0300)	**Instance:** of the object class to which this request is directed. See documentation for the server. In this example Instance 768 decimal (0300h) points to VI001 in the S2K as the first of 32 DINT variables to write.
	6	(0001)	**Data Byte Offset:** The number of bytes between this word and the beginning of the service data to be sent. If the offset is 0, the service data is located immediately after this data word (at word 7, see below). For example, if the offset were 2, then two bytes would be "skipped" and the data would begin at word 8.
	7	(00)	**LSB: Skipped** - Least significant byte “skipped” because of setting in word 6.
Service Data		(03)	**Service Data Byte 0, Attribute** – An attribute is used in service 0x10 and 0x0E messages. See documentation of targeted server device for meaning of specific attributes. Since word 6 “skipped” a byte this is the actual beginning of the service data. Locate data for messages without an attribute to start data here. May be at a different location depending on the value of word 6.
	8, 9	DINT	**Service Data:** May be located at a different offset based on word 6. Using the offset in word 6 allowed, in this example, the DINT data to be aligned on a word boundary.
	10 to end	-	**Service Data:** For this example the end of the service data is located at word 71 [6 header words + 1 skipped byte + 1 attribute byte + 64 data words].

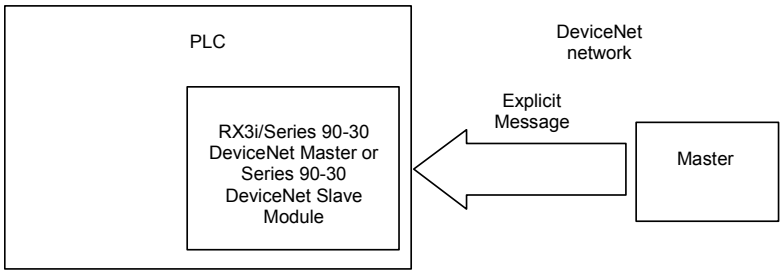
Send Device Explicit (& Extended), Reply Data Format

The DeviceNet Master Module replies to a Send Device Explicit or Send Device Explicit Extended COMMREQ. The reply data has the following format:

<table>
<tr><th>Word #</th><th colspan="4">Description</th></tr>
<tr><td>1</td><td colspan="4">Command code that this data block is replying to. (1 for Send Device Explicit, or 7 for Send Device Explicit Extended)</td></tr>
<tr><td rowspan="4">2</td><td colspan="4">Status of the explicit message. Bits 0 and 1 should both be 0.</td></tr>
<tr><td>bit 0</td><td colspan="3">1 = Explicit message response truncated to fit in shared memory buffer. The configured size of the explicit buffer of the device is too small.</td></tr>
<tr><td>bit 1</td><td colspan="3">1 = Explicit message response truncated to fit in Reply Memory. The reply buffer allocated by the COMMREQ is too small.</td></tr>
<tr><td>bits 2 - 15</td><td colspan="3">Reserved, should be ignored.</td></tr>
<tr><td>3</td><td colspan="4">MAC ID of the device producing this reply.</td></tr>
<tr><td>4</td><td colspan="4">Number of reply data bytes consumed. Note: if allocated buffers are not large enough this value should indicate the actual size of the reply data. Allocate reply size at least 10 bytes (for reply words 1-5) larger than the service data.</td></tr>
<tr><td>5</td><td colspan="4">DeviceNet service code / internal result code.
Values less than 0xFF: The service code low byte, in explicit message replies contains the same service that is returned on the DeviceNet network. Since the message is in reply to the explicit service issued by the COMMREQ, the high bit of the low byte is set to a 1. For example:
GET_ATTRIBUTE_SINGLE is service code 0x0E
The DeviceNet response will have the high bit set: 0x8E
SET_ATTRIBUTE_SINGLE is service code 0x10
With the high bit set on response: 0x90
DeviceNet errors use service code 0x14, and since errors are responses, the high bit will be set: 0x94. For example:
GET_ATTRIBUTE_SINGLE: 0x0E
DeviceNet error response: 0x94
(With following bytes of main code and additional code)</td></tr>
<tr><td></td><td>Value</td><td>Error</td><td>Value</td><td>Error</td></tr>
<tr><td></td><td>0x00 - 01</td><td>Reserved</td><td>0x12</td><td>Reserved</td></tr>
<tr><td></td><td>0x02</td><td>Resource needed for the object to perform the requested service not available.</td><td>0x13</td><td>The service did not supply enough data to perform the requested service</td></tr>
<tr><td></td><td>0x03 - 07</td><td>Reserved</td><td>0x14</td><td>Attribute specified in the request is not supported</td></tr>
<tr><td></td><td>0x08</td><td>Requested service not implemented or not defined for the object class/instance</td><td>0x15</td><td>The service supplied more data than was expected</td></tr>
<tr><td></td><td>0x09</td><td>invalid attribute data detected</td><td>0x16</td><td>The specified object does not exist in the device</td></tr>
<tr><td></td><td>0x0A</td><td>Reserved</td><td>0x17</td><td>Reserved</td></tr>
<tr><td></td><td>0x0B</td><td>Object is already in requested mode or state requested by the service</td><td>0x18</td><td>Attribute data of the object was not stored prior to the requested service</td></tr>
<tr><td></td><td>0x0C</td><td>Object cannot perform the requested service in its current mode / state</td><td>0x19</td><td>Attribute data of this object not saved by the object</td></tr>
<tr><td></td><td>0x0D</td><td>Reserved</td><td>0x1A - 1E</td><td>Reserved by DeviceNet</td></tr>
<tr><td></td><td>0x0E</td><td>Request to modify a non-modifiable attribute was received</td><td>0x1F</td><td>Vendor specific error</td></tr>
<tr><td></td><td>0x0F</td><td>Permission/privilege check failed</td><td>0x20</td><td>Invalid parameter</td></tr>
<tr><td></td><td>0x10</td><td>Device's current mode or state prohibits the requested service</td><td>0x21 - CF</td><td>Reserved</td></tr>
<tr><td></td><td>0x11</td><td>Data to be transmitted is larger than the allocated response buffer</td><td>0xD) - FF</td><td>Vendor specific object and class errors</td></tr>
<tr><td></td><td colspan="4">Values above 0xFF are internal DeviceNet Master Module codes (see below).</td></tr>
<tr><td></td><td>0x0100</td><td colspan="3">Explicit connection is not established</td></tr>
<tr><td></td><td>0x0101</td><td colspan="3">Explicit body format cannot represent requested class. (i.e. class > 255 and connection body format is 8/8 or 8/16)</td></tr>
<tr><td></td><td>0x0102</td><td colspan="3">Explicit body format cannot represent requested instance. (i.e. instance > 255 and connection body format is 8/8 or 8/16)</td></tr>
<tr><td></td><td>0x0103</td><td colspan="3">Resources not available to send explicit message</td></tr>
<tr><td></td><td>0x0104 - FFFF</td><td colspan="3">Reserved</td></tr>
<tr><td>6 - end</td><td colspan="4">Optional data as required by the service. The size of this data is indicated by word 4</td></tr>
</table>

Command Codes 2, 3 & 8: Reading and Responding to Client Explicit Messages

A client on the network may send DeviceNet explicit messages to a DeviceNet Slave Module or to a DeviceNet Master Module when explicit messaging is enabled for the server.



Objects that Do Not Require PLC Application Programming

Some client explicit messages are serviced directly by the DeviceNet module and do not require PLC application program interaction. The objects defined in the module firmware and reserved that do respond automatically are:

- Identity Object (0x01)
- Message Router Object (0x02)
- DeviceNet Object (0x03)
- Assembly Object (0x04)
- Connection Object (0x05)
- PLC Data Object (0x64).

You do not need COMMREQ commands to reply to services associated with these objects, the reply is generated by the DeviceNet module.

For many applications, client messages to the automatic message objects will achieve the application goal. For example, the PLC Data Object allows a device like a personal computer or another PLC to read or write an array of PLC data memory where the DeviceNet module is installed. This could be used to send a recipe to PLC %R memory or to retrieve an array of status data on demand. With a minimum of PLC programming to unpack or pack data within a single PLC data type, it is manageable to generate almost any data structure needed and access it as needed from the client. Making use of the automatic responses typically simplifies the PLC application program. It should be considered as a first option when designing the overall system.

Custom Explicit Messaging Using COMMREQs

The COMMREQ commands described in this section allow the PLC hosting the DeviceNet module to reply to “custom” explicit messages. One use of custom-explicit messaging is to allow the DeviceNet module to respond to existing client messages, or to emulate the response of a particular DeviceNet device template. For example, the DeviceNet module could emulate the response expected from a DeviceNet motor control center or position controller device. The only restriction is that the reserved automatic objects mentioned above may not be used for custom-explicit class object identifers. A DeviceNet object class identifier is specified by a numeric value in the range of 1-65535. With just six reserved identifiers, there is a lot of room to work with custom-explicit identifiers. In general, use object values higher than 0x64 for custom-explicit messages to minimize conflicts with DeviceNet objects in common use.

Through the use of custom-explicit messaging, the PLC program can implement application objects and allow other client devices on the DeviceNet network to read and write parameters within these objects.The application program in the PLC CPU can read and respond to DeviceNet custom-explicit messages as explained below.

How the Module Handles a Custom-Explicit Message

When the DeviceNet server receives a custom-explicit message from the network, it stores the messages in an internal queue. At any given time, up to 255 connections are available in the message queue. However, this message queue is used by all the client or server message functions within the module. For example the DeviceNet Master module’s polled connection to a slave is a client connection. Any explicit message to the DeviceNet module server from a client device will open an active server connection until that client message is replied to, or times out. After the connection is idle for a short time (2-3 seconds is typical) it is released back to the queue for a subsequent message to use. Should the same client device produce a second explicit message to the DeviceNet module before the idle timeout, the same connection is reused for the new message.

Additionally the Server Status word SERA bit is set logically ON for at least one PLC sweep. The Server Status word is mapped to PLC I/O via configuration and is optionally available via COMMREQ # 6. The application should monitor the Server Status word instead of constantly generating command 6 COMMREQs to search for the arrival of a custom-explicit message. Application logic in the PLC can monitor the SERA bit to determine when a client custom-explicit message is present in the queue. The PLC application program must process the client message and produce a reply before the client times out waiting for a response.

Therefore, the first step in responding to custom-explicit messages is for the PLC application program to read the next available custom-explicit request it has received from a client device on the network, using ***COMMREQ #2, Receive Server Explicit***.

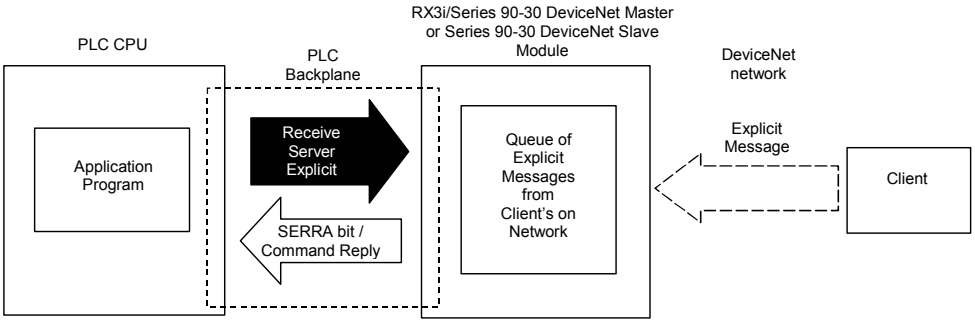
After reading an explicit message from the DeviceNet module, the PLC application program replies to the custom-explicit client request using ***COMMREQ #3, Send Server Explicit,*** for messages up to 238 bytes, or ***COMMREQ #8, Send Server Explicit Extended***, for longer messages.

To determine whether the DeviceNet module has completed sending the reply to the custom-explicit message, the program logic can subsequently use ***COMMREQ #5, Get Detailed Server Status***. The reply data to command five indicates whether the module

has sent a DeviceNet explicit reply message that was previously commanded by a Send Server Explicit COMMREQ.

Reading a DeviceNet Custom-Explicit Message from the DeviceNet Module

The Receive Server Explicit COMMREQ commands a DeviceNet Slave Module or a DeviceNet Master Module in slave mode to return a DeviceNet the most recent explicit client message it has received from the network.



A DeviceNet module can receive multiple explicit request messages from a network master before the first response message is generated.

Command code 2: Receive Server Explicit, Example COMMREQ Format

This example Receive Server Explicit COMMREQ specifies the command number (2), and sets up in the PLC CPU a memory type, starting address and length for the reply data that will be returned by the module. Test the SERA bit to initiate this command.

Word #	Dec	(Hex)	Description
1	00004	(0004)	**Command Length:** of command Data Block. For the Receive Server Explicit command the length is 4.
2	00000	(0000)	Always 0 (no-wait mode request)
3	00008	(0008)	**Status segment select:** Memory type of COMMREQ status words (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	00009	(0009)	**Status memory offset:** COMMREQ status words start address minus 1 (%R10 for example)
5	00000	(0000)	Reserved
6	00000	(0000)	Reserved
7	00002	(0002)	**Command code:** Receive Server Explicit (2)
8	00008	(0008)	**Reply segment select:** Memory type for the reply data. (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	00250	(00FA)	**Reply memory offset:** Offset within the memory type for the reply minus 1. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22). For this example, it is %R251.
10	00010	(000A)	**Reply size:** Maximum size required to hold the reply for the command: (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Note: It must be 15 bytes or more, or an error will be reported in the status word and the request will be ignored.

Receive Server Explicit, Reply Data Format

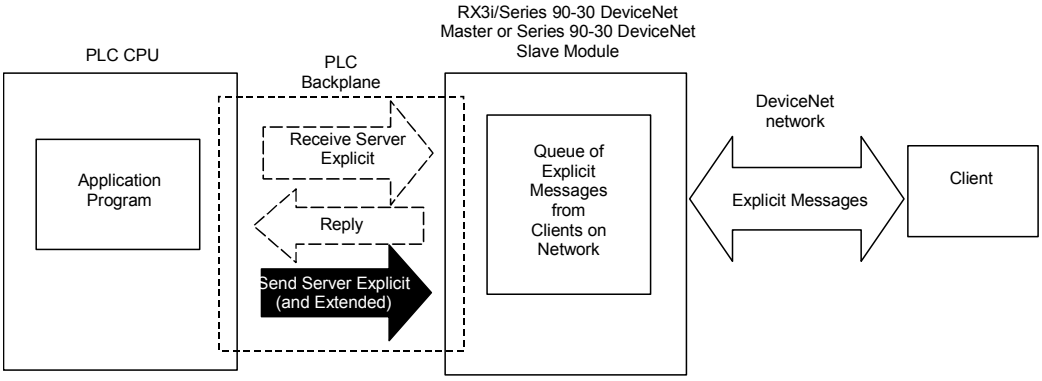
When it receives a Receive Server Explicit COMMREQ, the DeviceNet module returns a reply containing either the next explicit request in its internal memory , or an indication that there is no explicit request in the queue to process.

The reply contains information about the device that sent the explicit request, and a description of the service requested. The application program must be set up to respond appropriately to the request.

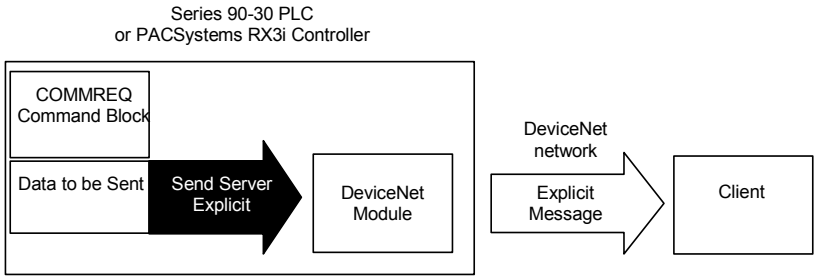
Word #	*Description*	
1	Command number that this data block is replying to. (2)	
2	Status of Receive Server Explicit command. If any of the following bits are set, the corresponding condition is true. A value of zero indicates that a message was retrieved.	
	bit 0	1 = No explicit request available. All remaining reply data is set to zero by the module.
	bits 1 - 15	Reserved.
3	**Connection ID:** The connection ID associated with the request.	
4	**Size:** Number of service data bytes	
5	**Service:** DeviceNet service code being requested.	
6	**Object Class:** The object class to which this is directed.	
7	**Instance:** The specific instance of the object class to which this request is directed.	
8 - end	**Service Data:** Optional data as required by the service. The size of this data is indicated by word 4.	

Command Codes 3 & 8: Replying to a DeviceNet Explicit Message

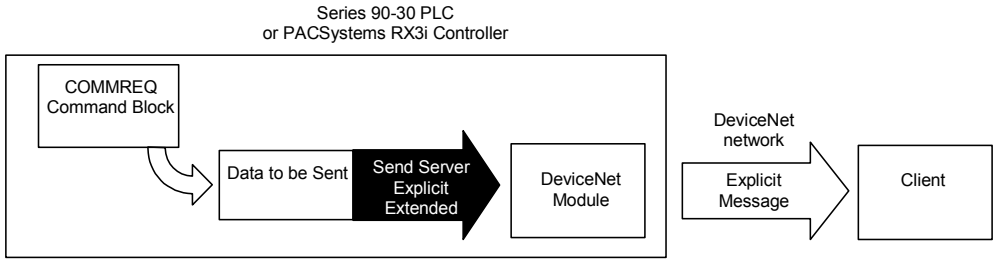
After retrieving a DeviceNet explicit message from a DeviceNet Slave Module or Master Module in slave mode, the application program can respond to the network master using ***COMMREQ #3, Send Server Explicit***. If the content of the response is more than 238 explicit data bytes, use ***COMMREQ #8, Send Server Explicit Extended*** instead.



The difference between Send Server Explicit and Send Server Explicit Extended is how they store the data that will be sent in PLC memory. For Send Server Explicit, the data to be sent is located in the same memory area as the COMMREQ command block.



For Send Server Explicit Extended, the data to be sent is located in a separate memory area, specified by a pointer in the COMMREQ command block. This makes it possible to store and send more data.



Command Code 3: Send Server Explicit, COMMREQ Description

For the Send Server Explicit COMMREQ there is no reply to the command. Care should be taken to verify the success of the command via the COMMREQ status word. You must have the appropriate connection ID from command 2 prior to executing this command.

Word #	*Description*
1	**Command Length:** Length of command. For this command, the length is 5 words plus the number of words of Service Data (Note: Service Data is in bytes, divide by 2 and round up). For this example Service Data is 5 bytes = 3 Words
2	Always 0 (no-wait mode request)
3	**Status segment select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	**Status memory offset:** COMMREQ status word address minus 1 (%R10 for this example)
5	Reserved
6	Reserved
7	**Command Code:** Send Server Explicit (3)
8	**Connection ID:** required for request / response matching. From the command 2 COMMREQ used to service this client reply.
9	**Size:** Number of service data bytes
10	**Service:** DeviceNet service code.
11	**Service Data Offset:** The number of bytes between this word and the beginning of the service data to be sent. If the offset is 0, the service data is located immediately after this data word (at word 12, see below). For example, if the offset were 2, then two bytes would be "skipped" and the data would begin at word 13.
12	**Skipped (optional):**
	Service Data Byte 0, Attribute:
13 to end	**Service Data:**

Command Code 8: Send Server Explicit Extended, COMMREQ

Word #	Description
1	**Command Length:** Always 4 words for command 8
2	Always 0 (no-wait mode request)
3	**Status segment select:** Memory type of COMMREQ status word (%R for this example). (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
4	**Status memory offset:** COMMREQ status word address minus 1 (%R10 for this example)
5	Reserved
6	Reserved
7	**Command Code:** Send Server Explicit Extended (8)
8	**Data segment select:** Memory type for the data to be sent. (8 = R, 10 = AI, 12 = AQ, 16 = I, 18 = Q, 20 = T, 22 = M)
9	**Data memory offset:** Offset within the memory type for the data to be sent. (Word offset for memory types: 8, 10, 12; byte offset for memory types: 16, 18, 20, 22)
10	**Data memory size:** Size data block to send (in words for memory types: 8, 10, 12; in bytes for memory types: 16, 18, 20, 22). Must be large enough to specify the entire explicit data block defined below. In bytes, the size needed is “Number of Service Data bytes + 8 + Service Data Byte offset.” Unit conversions are required for non-byte units.

Send Server Explicit Extended, Data Format

The following data format must be written to the PLC memory location specifed in the command.

Word #	Description
1	**Connection ID:** The server connection ID is required for request / response matching. From the command 2 used to initiate service to this request.
2	**Service Data Size:** Number of service data bytes
3	**Service Code:** DeviceNet service code.
4	**Service data byte Offset:** The number of bytes between this word and the beginning of the service data to be sent. If the offset is 0, the service data is located immediately after this data word (at word 7, see below). For example, if the offset were 2, then two bytes would be "skipped" and the data would begin at word 8.
5	**Skipped (optional):**
	Service Data byte 0, Attribute:
6 to end	**Service Data:**

Chapter 7 DeviceNet Objects

This chapter describes the DeviceNet Objects that are defined for the DeviceNet Master Module and the DeviceNet Slave Module.

These modules use the following types of Objects in DeviceNet information exchanges:

- Identity Object
- Message Router Object
- DeviceNet Object
- Assembly Objects
- Connection Object
- PLC Data Object

Identity Object

Object Class 0x01

The Identity Object contains the module's identifying information.

There is only one instance of this object.

Class Attributes

Id	Description	Get	Set	Limits
1	Revision	O	O	
2	Max Instance	O	O	
6	Max ID of class attributes	O	O	
7	Max ID of instance attributes	O	O	
● Supported O Not supported				

Class Services

Service		Param. Options
Get_Attributes_All	O	
Reset	O	
Get_Attribute_Single	O	
Find_Next_Object_Instance	O	
● Supported O Not supported		

Instance Attributes

Id	Description	Get	Set	Limits
1	Vendor	●	O	326
2	Device Type	●	O	0x0C
3	Product Code	●	O	Master: 2 Slave: 1
4	Revision[1]	●	O	
5	Status	●	O	
6	Serial Number	●	O	
7	Product Name	●	O	Master: “90-30 DeviceNet Master” Slave: “90-30 DeviceNet Slave”
8	State	O	O	
● Supported O Not supported 1 DeviceNet specifies that Major and Minor Revision may not be 0, the lowest valid minor revision is 1				

Instance Services

Service		Param. Options
Reset	●	
Get_Attribute_Single	●	
● Supported O Not supported		

Message Router Object

Object Class 0x02

The Message Router Object routes the explicit messages it receives to other Objects.

There is only 1 instance of this object.

Class Attributes

Id	Description	Get	Set	Limits
1	Revision	O	O	
4	Optional Attribute List	O	O	
5	Optional Service List	O	O	
6	Max ID of class attributes	O	O	
7	Max ID of instance attributes	O	O	
● Supported O Not supported				

Class Services

Service		Param. Options
Get_Attributes_All	O	
Get_Attribute_Single	O	
● Supported O Not supported		

Instance Attributes

Id	Description	Get	Set	Limits
1	Object List	O	O	
2	Maximum connections supported	O	O	
3	Number of active connections	O	O	
4	Active connections list	O	O	
● Supported O Not supported				

Instance Services

Service		Param. Options
Get_Attributes_All	O	
Get_Attribute_Single	O	
● Supported O Not supported		

DeviceNet Object

Object Class 0x03

The DeviceNet Object contains the parameters for DeviceNet operations.

There is only 1 instance of this object.

Class Attributes

Id	Description	Get	Set	Limits
1	Revision	●	O	
● Supported O Not supported				

Class Services

Service		Param. Options
Get_Attribute_Single	●	
● Supported O Not supported		

Instance Attributes

Id	Description	Get	Set	Limits
1	MAC ID	●	O	
2	Baud Rate	●	O	
3	BOI	O	O	
4	Bus-off counter	O	O	
5	Allocation information	●	O	
6	MAC ID switch changed	O	O	
7	Baud rate switch changed	O	O	
8	MAC ID switch value	O	O	
9	Baud rate switch value	O	O	
● Supported O Not supported				

Instance Services

Service		Param. Options
Get_Attribute_Single	●	
Allocate M/S connection set	●	
Release M/S connection set	●	
● Supported O Not supported		

Assembly Object

Object Class 0x04

Assembly Objects group the data from multiple applications objects into a single message.

There are up to 4 instances of this object.

Class Attributes

Id	Description	Get	Set	Limits
1	Revision	O	O	
2	Max Instance	O	O	
● Supported O Not supported				

Class Services

Service		Param. Options
Get_Attributes_All	O	
Get_Attribute_Single	O	
● Supported O Not supported		

Instance Attributes

Id	Description	Get	Set	Limits
1	Number of Members in List	O	O	
2	Member List	O	O	
3	Data	●	●*	
● Supported O Not supported * Only for non-Established output connections				

Instance Services

Service		Param. Options
Get_Attribute_Single	●	
Set_Attribute_Single	●*	
● Supported O Not supported * Only for non-Established output connections		

Connection Object

Object Class 0x05

The Connection Object handles the module connection. Each Connection Object represents the end points of a connection. In an I/O message, all the information about what to do with the data in the message is contained in the Connection Object for that I/O message.

There can be a maximum of 255 instances of the Connection Object in any combination of connection types.

Class Attributes

Id	Description	Get	Set	Limits
1	Revision	O	O	
● Supported O Not supported				

Class Services

Service		Param. Options
Reset	O	
Create	O	
Delete	O	
Get_Attribute_Single	O	
Find_Next_Object_Instance	O	
● Supported O Not supported		

Instance Attributes

Id	Description	Get	Set	Attribute Limits/Fixed Values									
				1	2	3	4	5	6	7	8	9	10
1	State	●	O										
2	Instance Type	●	O	0	0	1	1	1	1	1	1	1	1
3	Transport class trigger	●	O	0x23	0x83	0x22	0x83	0x80	0x83	0x03 or 0x13	0x03 or 0x13	0x00 or 0x10	0x00 or 0x10
4	Produced connection ID	●	O										
5	Consumed connection ID	●	O									0xFFFF	0xFFFF
6	Initial comm. characteristics	●	O			0x10	0x01	0xF0	0x02	0x01	0x01	0x0F	0x0F
7	Produced connection size	●	O	0XFFFF	0xFFFF								
8	Consumed connection size	●	O	0xFFFF	0xFFFF								
9	Expected packet rate	●	●	2500	2500	250		250		250	250	250	250
12	Watchdog timeout action(1)	●	●	0x01	0x01 or 0x03	0	0	0	0	0	0	0	0
13	Produced connection path length	●	O	0	0								
14	Produced connection path	●	O	-	-								
15	Consumed connection path length	●	O	0	0						4	0	0
16	Consumed connection path	●	O	-	-						0x20 0x2B 0x24 0x01	-	-
17	Production Inhibit Time (2)	●	●	-	0xFFFF	-	-	-	-	-	-	-	-

● Supported O Not supported

(1) set attribute service supported only by the server explicit messaging connection

(2) set attribute service supported on client trigger connections only

Connection types:

1. Explicit Client Connection Attribute Limits
2. Explicit Server Connection Attribute Limits
3. Poll Client Connection Attribute Limits
4. Poll Server Connection Attribute Limits
5. Strobe Client Connection Attribute Limits
6. Strobe Server Connection Attribute Limits
7. COS/Cyclic Client Connection Attribute Limits (acknowledged)
8. COS/Cyclic Server Connection Attribute Limits (acknowledged)
9. COS/Cyclic Client Connection Attribute Limits (unacknowledged)
10. COS/Cyclic Server Connection Attribute Limits (unacknowledged)

Instance Services

Service		Param. Options
Reset	O	
Delete	O	
Apply_Attributes	O	
Get_Attribute_Single	●	
Set_Attribute_Single	●	
● Supported O Not supported		

PLC Data Object

Object Class 0x64

Number of Instances: 7

Specific Services

Service Code	Implemented for		Service Name
	Class	Instance	
0x32	Yes	No	GET_PLC_DATA
0x33	Yes	No	SET_PLC_DATA

Specific Instances

The DeviceNet modules support access to PLC memory using the following segment selectors. The segment selectors are set in the Instance ID of the Explicit request.

Name	Selector Code / Instance ID (dec)	Description
REGISTER_TBL_WORD	8	R memory
ANALOG_IN_TBL_WORD	10	AI memory
ANALOG_OUT_TBL_WORD	12	AQ memory
DISCRETE_IN_BYTE	16	I memory in byte mode
DISCRETE_OUT_BYTE	18	Q memory in byte mode
DISCRETE_TEMP_BYTE	20	T memory in byte mode
DISCRETE_INTERNAL_BYTE	22	M memory in byte mode

GET_PLC_DATA Service (0x32)

Service Specific Request Data

Name	Data Type	Description
Offset	UINT2	Offset into memory area. Units are dependent on Seg_Selector. Offset is 1-based, not 0-based.
Length	UINT2	Length of data to read. Units are dependent on Seg_Selector. A maximum of 255 bytes per service request can be read. If more than 255 bytes are requested, GET_PLC_DATA will return error 0x11 0xFF (reply data too large).

Note: The Service Request Data for this service does not include an attribute byte.

Service Specific Success Response Data

On success, the Explicit Service field will be set to 0xB2 and will contain the following data.

Name	Data Type	Description
Data	Array	Data requested

SET_PLC_DATA Service (0x33)

Service Specific Request Data

Name	Data Type	Description
Offset	UINT2	Offset into memory area. Units are dependent on Seg_Selector. Offset is 1-based, not 0-based.
Length	UINT2	Length of data to write. Units are dependent on Seg_Selector. A maximum of 255 bytes per service request can be written. If more than 255 bytes are sent, SET_PLC_DATA will return error 0x15 0xFF (too much data).
Data	UNIT[Length]	Data to be written. Size of data is Length * size of Unit.

Note: The Service Request Data for this service does not include an attribute byte.

Service Specific Success Response Data

On success, the Explicit Service field will be set to 0xB3 and will contain no data.

Service Specific Error Response Data

Error responses contain 0x94 in the Service Code field of the explicit message, and an Error Code and Additional Code in Bytes 1 and 2 respectively. The following error codes may be returned:

Byte 1	Byte 2	Description
0x08	0xFF	Service not supported (invalid Service Code)
0x0C	0xFF	Object state conflict (i.e. internal mail system unavailable)
0x11	0xFF	Reply data too large (the data to be transmitted is larger than the internal transmit buffer (255 bytes))
0x13	0xFF	Not enough data
0x15	0xFF	Too much data
0x16	0xFF	Object does not exist (invalid Instance ID or Class ID)
0x20	0x01	Register base address is not in the proper form
0x20	0x02	Register quantity is not in the proper form
0x20	0x03	Register base address out of range for this CPU model
0x20	0x04	Register quantity out of range for this CPU model or DN module
0x20	0x05	Sum of register base and quantity are out of range for this CPU
0x20	0x06	Error communicating with PLC on GET_PLC_RGS service
0x20	0x07	Error communicating with PLC on SET_PLC_RGS service

Appendix A

DeviceNet EDS Files

This section contains the Electronic Datasheet (EDS) Files that are defined for the DeviceNet Master Module and the DeviceNet Slave Module. These files are included only for reference; an electronic version of the most up-to-date EDS file is provided as part of the GE Fanuc programming software Machine Edition Logic Developer. The EDS files (and any updates) are also on the GE Fanuc web Site www.gefanuc.com.

Electronic Datasheet File for the DeviceNet Master Module

```
$ DeviceNet Electronic Data Sheet

$ Copyright (C) 2002 GE Fanuc Automation North America
[File]
        DescText   = "90-30 DeviceNet Master";
        CreateDate = 06-05-2002;
        CreateTime = 12:59:12;
        ModDate    = 10-23-2002;
        ModTime    = 12:34:56;
        Revision   = 1.04;

[Device]
        VendCode    = 326;
        VendName    = "GE Fanuc Automation NA Inc.";
        ProdType    = 12;
        ProdTypeStr = "Communication Adapter";
        ProdCode    = 32;
        MajRev      = 1;
        MinRev      = 16;
        ProdName    = "90-30 DeviceNet Master";
        Catalog     = "IC693DNM200";
```

Electronic Datasheet File for the DeviceNet Slave Module

```
$    DeviceNet Electronic Data Sheet
$ Copyright (C) 2002 GE Fanuc Automation North America

[File]
        DescText   = "90-30 DeviceNet Slave";
        CreateDate = 06-05-2002;
        CreateTime = 12:59:12;
        ModDate    = 10-23-2002;
        ModTime    = 12:34:56;
        Revision   = 1.04;

[Device]
        VendCode    = 326;
        VendName    = "GE Fanuc Automation NA Inc.";
        ProdType    = 12;
        ProdTypeStr = "Communication Adapter";
        ProdCode    = 31;
        MajRev      = 1;
        MinRev      = 16;
        ProdName    = "90-30 DeviceNet Slave";
        Catalog     = "IC693DNS201";
```

A

ACK timeout, 3-5, 4-5
Add Device errors, 6-10
Adding slaves to the network, 3-8
Additional error codes, 6-11
Assembly Object, 6-24, 7-5

B

Baud Rate, 3-5, 4-5
Bit-strobed I/O mode, 3-12, 3-18, 4-10
Bus
  cable specifications, 2-3
  connectors, 2-3
  length, 2-4
  termination, 2-3
Bus off error detection, 5-3

C

Cable connection to module, 2-5
Cable specifications, 2-3
CAN acknowledgment error counter, 6-9
CAN communication error counter, 6-9
CAN lost messages counter, 6-9
CAN Network Status word, 6-9
CAN receive counter, 6-9
CAN receive queue overrun counter, 6-9
CAN status, 5-6
CAN transmit counter, 6-9
Change of State I/O mode, 3-13, 3-19, 4-11
COMMREQs, 6-1
  #1, Send Device Explicit, 6-18
  #2, Receive Server Explicit, 6-25
  #3, Send Server Explicit, 6-25
  #4, Get Detailed Device Status, 6-12
  #5, Get Detailed Server Status, 6-14, 6-25
  #6, Get Status Information, 6-16
  #7, Send Device Explicit Extended, 6-18
  #8, Send Server Explicit Extended, 6-25
  #9, Read Module Header, 6-8
  error detection and handling, 6-5
  general information
    programming recommendations, 6-5
  ladder instruction, 6-3
  status word, 6-7
Configuration
  Master module, 3-1, 4-1
Configuration file errors, 6-10
Configuration size, 3-2
Configuring a DeviceNet Slave Module, 4-2
Configuring DeviceNet Master Module, 3-2
Configuring network devices, 3-8
Configuring network nettings, 3-2
Configuring network settings, 3-9, 3-16
Connection Object, 6-24, 7-6
Connectors, 2-3
COS. *See* change of state
CPU fails to read inputs, 5-3
Current limits for cable, 2-2
Custom Explicit Messaging, 6-25
Cyclic I/O mode, 3-14, 3-20, 4-12

D

Data Areas tab, 3-6, 4-5
Device status bits, 5-7
DeviceNet communications, 1-8
DeviceNet data rates, 1-3
DeviceNet Master Module, 1-3
DeviceNet message field, 1-8
DeviceNet network, 1-7
DeviceNet network power, 2-2
DeviceNet Object, 6-24, 7-4
DeviceNet object class identifier, 6-25
DeviceNet Objects, 7-1
DeviceNet serial number, 6-9
DeviceNet Slave Module, 1-5
Download EDS files, 3-6
Download names and descriptions, 3-5
Drops, 2-5

E

EDS File, 3-8
EDS files, A-1
EDS files, downloading, 3-6
Error Code., 6-9
Explicit Messaging, 1-8
Explicit Messaging configuration, 3-15, 3-21

F

Fatal Error codes, 6-11
Fatal module error, 5-3
Fault state transmission, 3-4, 4-5
Fault Table entries, 3-5
Fault Table screens, 5-5
Firmware ID, 5-6
Firmware upgrade, 2-9

G

Get Detailed Device Status, 6-2
Get Detailed Server Status, 6-2
Get Status Info, 6-2
Grounding, 2-6

H

Hand-Held Programmer, 1-4, 1-6
Headend Fault codes, 6-10

I

I/O Messaging, 1-8
  configuration, 3-10, 3-16
I/O Messaging configuration, 4-8
Identity Object, 6-24, 7-2
Idle packets, 3-4
Initialization file errors, 6-10
Input data amounts, 1-3, 1-5
Inputs on loss of slave, 3-5
Installation procedures, 2-1
  installing the module in the rack, 2-7

K

Kernel identification, 6-9

L

LEDs, 2-8
Logicmaster, 1-4, 1-6

M

MAC ID, 3-4, 4-5
Machine Edition Logic Developer, 1-4, 1-6
Message Router Object, 6-24, 7-3
Module Header file, 6-11
Module ID, 6-9
Module operation, 5-1, 5-2
Module revision, 6-9
Module serial number, 6-9
Module Status LED red, 6-11
Module Type, 6-9

N

Network Status Data address, 3-4
Network termination, 2-5

O

Online errors, 6-10
Open DeviceNet Vendors Association (ODVA), 1-1
Operation in Program mode, 5-2
Output data amounts, 1-3, 1-5
Outputs on loss of master, 4-5

P

PLC Data Object, 6-24, 7-8
PLC power supply load, 2-2
PLC status references, 5-6
Polled I/O mode, 3-10, 3-11, 3-16, 3-17, 4-8, 4-9
Power Consumption, 1-4, 1-6
Power Consumption tab, 3-6, 4-6
Program mode transmission, 3-4, 4-5

R

Read Module Header, 6-2
Receive Server Explicit, 6-2
Reconnect time, 3-5
Removing the module from the rack, 2-7
RS-232 serial port, 2-9

S

Scan interval, 3-5
Send Device Explicit, 6-2
Send Device Explicit Extended, 6-2
Send Server Explicit, 6-2
Send Server Explicit Extended, 6-2
Send Zeroed Data, 5-2
SERA bit, 6-25
Serial cable, 2-9
Server operation, 5-4
Server status, 5-6
Slave Status Bit address, 3-4
Slave Status Fault Table entries, 4-5
Specifications
  environment, 1-4, 1-6
  power consumption, 1-6
  slave module, 1-6
Start Scan errors, 6-10
Status Address, 4-5
Status bits for network devices, 5-7
Status Word codes, 6-7
Status words, 5-6
Strobed I/O mode, 3-12, 3-18, 4-10

T

Taps, 2-5
Terminating the bus, 2-3
Tick Interval, 6-9
Transmit Idle, 5-2

U

UCMM, 1-3, 1-5

V

VersaPro, 1-4, 1-6

W

Window size, 6-9