AUTOMATION

User Manual

UM EN IL DN BK DI8 DO4-PAC

Order No.: 2910334

Inline Bus Coupler for DeviceNet™
With 8 Digital Inputs and 4 Digital Outputs

PHOENIX CONTACT
INSPIRING INNOVATIONS

DOWNLOADED FROM WWW.SCATTS.CO.UK

AUTOMATION

User Manual

Inline Bus Coupler for DeviceNet™ With 8 Digital Inputs and 4 Digital Outputs

11/2007

Designation: UM EN IL DN BK DI8 DO4-PAC

Revision: 00

Order No.: 2910334

This user manual is valid for:

Designation	Version	Order No.
IL DN BK DI8 DO4-PAC	01/01 or higher	2897211

Please Observe the Following Notes

In order to ensure the safe use of the product described, you have to read and understand this manual. The following notes provide information on how to use this manual.

User Group of This Manual

The use of products described in this manual is oriented exclusively to qualified electricians or persons instructed by them, who are familiar with applicable standards and other regulations regarding electrical engineering and, in particular, the relevant safety concepts.

Phoenix Contact accepts no liability for erroneous handling or damage to products from Phoenix Contact or third-party products resulting from disregard of information contained in this manual.

Explanation of Symbols Used and Signal Words

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

DANGER
This indicates a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING
This indicates a hazardous situation which, if not avoided, could result in death or serious injury.

CAUTION
This indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.

The following types of messages provide information about possible property damage and general information concerning proper operation and ease-of-use.

NOTE
This symbol and the accompanying text alerts the reader to a situation which may cause damage or malfunction to the device, either hardware or software, or surrounding property.

This symbol and the accompanying text provides additional information to the reader. It is also used as a reference to other sources of information (manuals, data sheets, literature) on the subject matter, product, etc.

General Terms and Conditions of Use for Technical Documentation

Phoenix Contact reserves the right to alter, correct, and/or improve the technical documentation and the products described in the technical documentation at its own discretion and without giving prior notice, insofar as this is reasonable for the user. The same applies to any technical changes that serve the purpose of technical progress.

The receipt of technical documentation (in particular data sheets, installation instructions, manuals, etc.) does not constitute any further duty on the part of Phoenix Contact to furnish information on alterations to products and/or technical documentation. Any other agreement shall only apply if expressly confirmed in writing by Phoenix Contact. Please note that the supplied documentation is product-specific documentation only and that you are responsible for checking the suitability and intended use of the products in your specific application, in particular with regard to observing the applicable standards and regulations. Although Phoenix Contact makes every effort to ensure that the information content is accurate, up-to-date, and state-of-the-art, technical inaccuracies and/or printing errors in the information cannot be ruled out. Phoenix Contact does not offer any guarantees as to the reliability, accuracy or completeness of the information. All information made available in the technical data is supplied without any accompanying guarantee, whether expressly mentioned, implied or tacitly assumed. This information does not include any guarantees regarding quality, does not describe any fair marketable quality, and does not make any claims as to quality guarantees or guarantees regarding the suitability for a special purpose.

Phoenix Contact accepts no liability or responsibility for errors or omissions in the content of the technical documentation (in particular data sheets, installation instructions, manuals, etc.).

The aforementioned limitations of liability and exemptions from liability do not apply, in so far as liability must be assumed, e.g., according to product liability law, in cases of premeditation, gross negligence, on account of loss of life, physical injury or damage to health or on account of the violation of important contractual obligations. Claims for damages for the violation of important contractual obligations are, however, limited to contract-typical, predictable damages, provided there is no premeditation or gross negligence, or that liability is assumed on account of loss of life, physical injury or damage to health. This ruling does not imply a change in the burden of proof to the detriment of the user.

Statement of Legal Authority

How to Contact us

Internet

Up-to-date information on Phoenix Contact products and our Terms and Conditions can be found on the Internet at:

www.phoenixcontact.com.

Make sure you always use the latest documentation.
It can be downloaded at:

www.download.phoenixcontact.com.

A conversion table is available on the Internet at:

www.download.phoenixcontact.com/general/7000_en_00.pdf.

Subsidiaries

If there are any problems that cannot be solved using the documentation, please contact your Phoenix Contact subsidiary.
Subsidiary contact information is available at www.phoenixcontact.com.

Published by

PHOENIX CONTACT GmbH & Co. KG
Flachsmarkstraße 8
32825 Blomberg
Germany
Phone +49 - (0) 52 35 - 3-00
Fax +49 - (0) 52 35 - 3-4 12 00

PHOENIX CONTACT
586 Fulling Mill Road
P.O. Box 4100
Harrisburg, PA 17111-0100
USA
Phone +1-717-944-1300

Should you have any suggestions or recommendations for improvement of the contents and layout of our manuals, please send your comments to

tecdoc@phoenixcontact.com.

Table of Contents

1 The Inline DeviceNet™ Bus Coupler

DeviceNet™ Bus Coupler

The DeviceNet™ bus coupler, shown in Figure 1-1, provides an interface between DeviceNet™ and the Phoenix Contact range of Inline I/O modules. It also provides the required bus signal conditioning and the power supply for the connected station components. Bus couplers are currently available for the connection of copper cables.

The bus coupler provides the initial connection for the main supply, U_M, and the segment (I/O) supply, U_S, to the station. You can also provide the main supply U_M and the segment supply U_S using a power terminal and/or a segment terminal respectively.

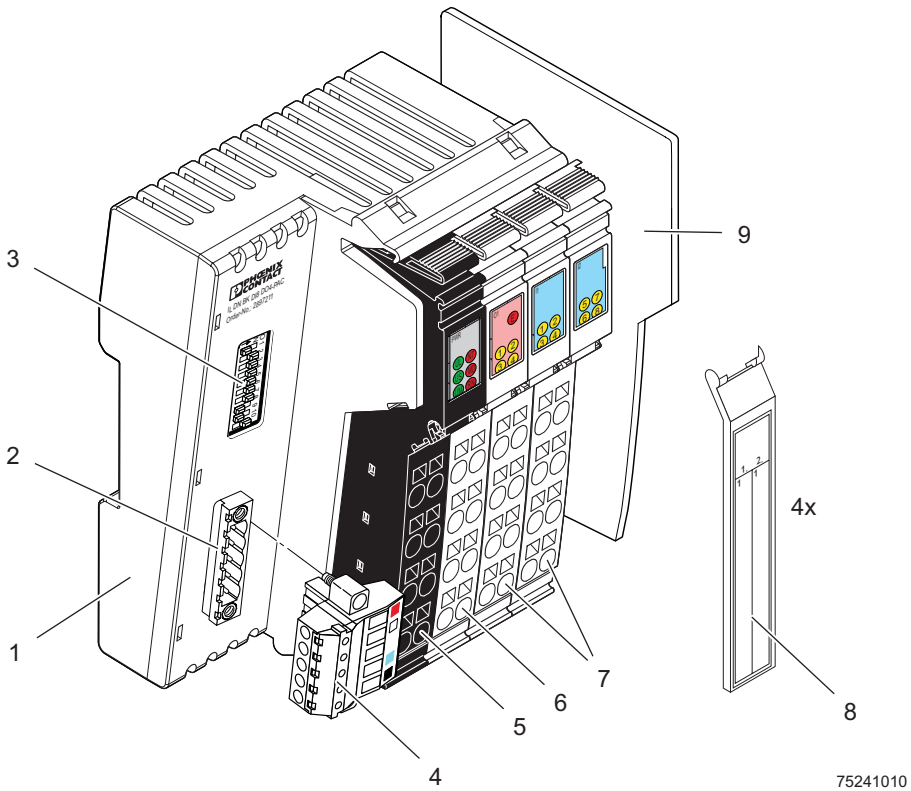


Figure 1-1 Features of the Inline DeviceNet™ bus coupler

Key:

1	Bus coupler	**5**	Power connector
2	DeviceNet™ connection	**6**	Digital output
3	DIP switches	**7**	Digital input
4	Twin COMBICON DeviceNet™ network connector (not included in scope of delivery of IL DN BK DI8 DO4-PAC)	**8**	Labeling field
		9	End plate

The Electronic Data Sheet (EDS) can be found on the Internet at:
www.download.phoenixcontact.com.

Features

The key features of the IL DN BK DI8 DO4-PAC are listed below:

– Module Features

- DeviceNet™ connection via TWIN COMBICON connector
- DIP switch to set the DeviceNet™ address
- Supported DeviceNet™ addresses 0 to 63
- Device description using EDS file
- 8 digital inputs
- 4 digital outputs
- Diagnostic and status indicators
- Automatic baud rate identification in the local bus (500 kbps or 2 Mbps)
- Connection of up to 63 I/O modules
- Can be installed in the field without software for automatic configuration of the station
- Digital inputs with latch function
- Stored (latched) diagnostic data
- Reservation of I/O memory area for future expansions possible (no hardware required)
- Supported Inline module types:
  Digital inputs and outputs
  Analog inputs and outputs
  Intelligent function module

– Features of DeviceNet™

- Faulted node recovery
- Baud rates:
  125 kBaud, 250 kBaud, 500 kBaud
- I/O slave messages:
  Polling, Cyclic, Change-of-State (COS), Bit strobing

For additional information about the supported Inline modules, please refer to the “I/O Modules at Bus Couplers“ application note. It can be downloaded at: www.download.phoenixcontact.com.

2 Installation

2.1 The DeviceNet™ System

DeviceNet™ is a communications link that transmits data between control systems (e.g., PLCs, PCs, VMEbus computers, robot controllers, etc.) and distributed industrial devices (such as switches, sensors, valve manifolds, motor starters, bar code readers, drives, displays, and operator interfaces) to network and eliminate expensive hard wiring.

DeviceNet™ has a linear structure. This structure consists of a main trunk line with drop lines routed to the networked devices. Power and signal is routed on the same network cable.

Communications is very flexible with DeviceNet™. Peer-to-peer , multi-cast (one-to-many), master/slave, bit-strobe, and change-of-state (exception-based) communication options are available when using DeviceNet™.

The DeviceNet™ topology allows for the removal and replacement of powered devices from the network with no interruptions to the rest of the network.

2.2 Example Topology of a DeviceNet™ System

DeviceNet™ utilizes a linear bus topology. See Figure 2-1. Terminating resistors are required at each end of the trunk. Drops, made of trunk or drop cable, may be as long as 6 m (20 feet), and each drop can support one or more nodes.

The total length of trunk and drop cable that can be used on the network depends upon the type of cable and/or the baud rate. See Table 2-1 when determining the length of the network based on the baud rate. See the DeviceNet™ specification for more details on network lengths in relation to current consumption.

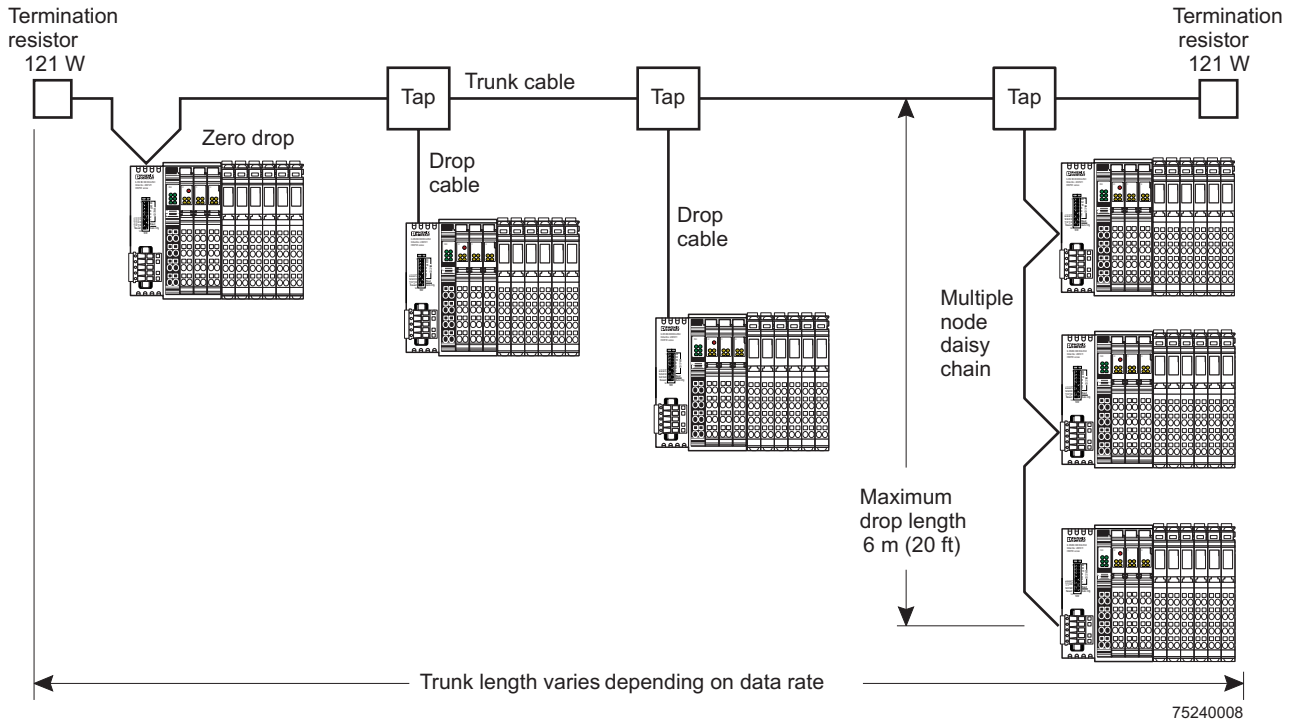


Figure 2-1 DeviceNet™ topology example

Table 2-1 Maximum cable length

Baud-Rate	Baud Rate DIP-Switch Settings	Trunk Distance		Drop Length	
		Thick	Thin	Maximum	Cumulative
125 kBaud	8 OFF, 9 OFF	500 m	100 m	6 m	156 m
250 kBaud	8 OFF, 9 ON	250 m	100 m	6 m	78 m
500 kBaud	8 ON, 9 OFF	100 m	100 m	6 m	39 m

2.2.1 Bus Coupler Module

The first step in setting up a modular I/O station is to connect the bus coupler module to the DeviceNet™ cable. I/O modules may be installed branching off from these bus coupler modules, to create a local bus. The bus coupler module also supplies communications power to the connected I/O modules.

A breakdown of the supply voltage on the bus coupler module stops the communications to the modules connected to the bus coupler and causes an error message for the node.

Tasks of the bus coupler module

- Coupling of DeviceNet™ and the Inline I/O modules
- Supplying the I/O modules with communications power
- Electrical isolation of the local I/O
- Providing diagnostic information from the connected I/O to DeviceNet™

Maximum number of devices

The maximum number of devices that you can connect to a bus coupler is determined by the following parameters:

- Up to 63 devices can be connected to a bus coupler. This number includes all the devices after the bus coupler with input or output data, i.e., the Inline modules and the modules for Fieldline Modular local bus.
- The bus coupler can supply a maximum of 0.8 A for communications power and 0.5 A for analog supply power.
- The current carrying capacity of the voltage jumpers is limited. For the limit values of the individual voltage jumpers, refer to the user manual IL SYS INST UM E

NOTE:

Observe the current consumption of each device when configuring an Inline station. Current consumption specifications can be found in the product-specific data sheets.

2.2.2 Trunk Cable

The trunk cable is used to connect nodes to each other if no drops exist on the network. If drops are present, device taps will be linked via the trunk cable (See Figure 2-1 on page 2-2). The maximum number of nodes on DeviceNet™ is limited to 64.

2.2.3 Drop Cable

The drop cable is used to connect drops to the trunk cable. This connection is made by using a network tap. Drop cable is typically thinner and more flexible than trunk cable.

2.2.4 Termination Resistors

DeviceNet™ requires a terminating resistor to be installed at each end of the trunk. These be 121 Ohm, 1% metal film, and have a power dissipation rating of 0.25 W.

2.3 Inline Station

The Inline product range is a modular automation system. Inline modules are joined together to create functional units that meet the requirements of the application.
See Figure 2-2, shown with the DeviceNet™ bus coupler. Both communication and power routing is accomplished automatically by the physical interconnections between the I/O modules. Additional networking options permit the Inline station to branch out to various machine mounted I/O modules such as Fieldline Modular, or AS-i devices.

For general information on the setup of an Inline station, please refer to the IL SYS INST UM E user manual.

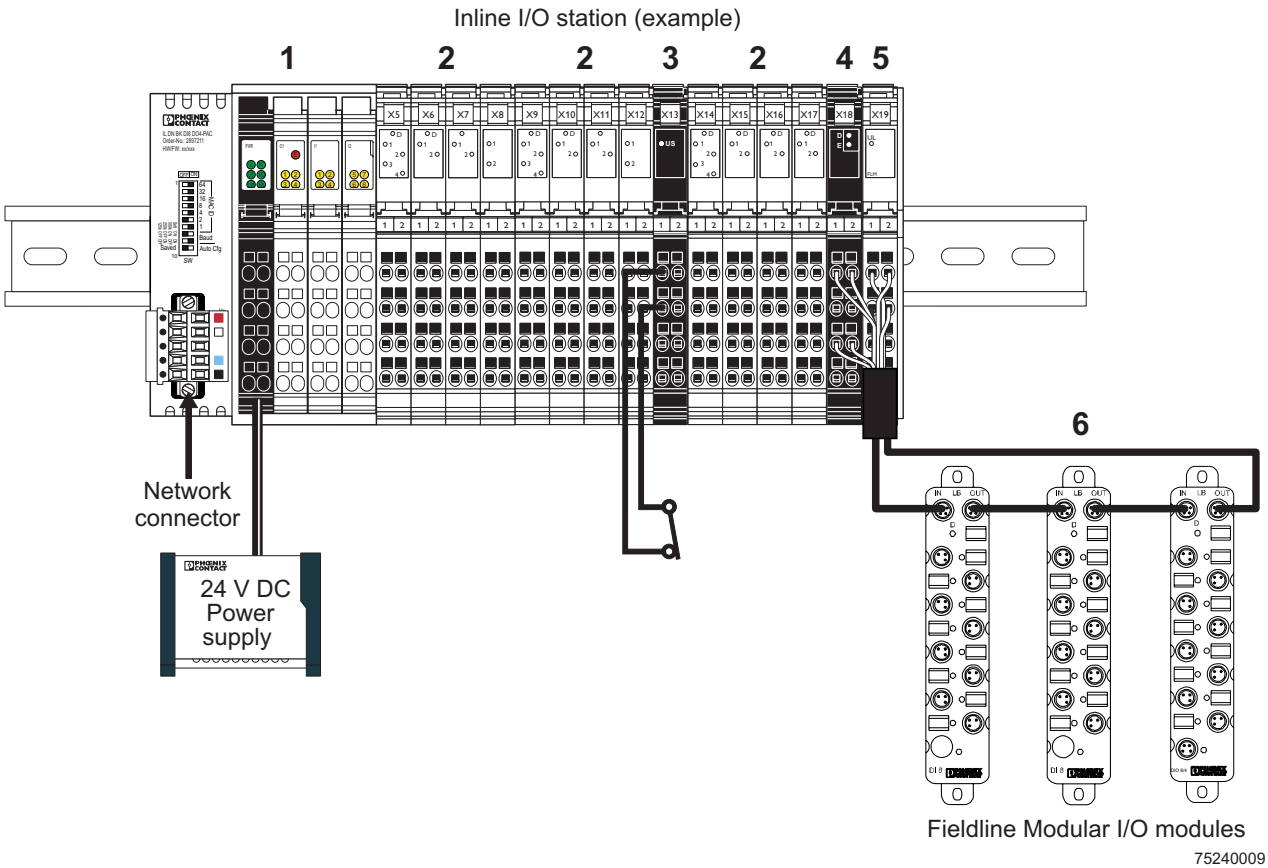


Figure 2-2 Example of a basic Inline station

Key:

1 Bus coupler (here: IL DN BK DI8 DO4-PAC)
2 I/0 modules
3 Power terminal
4 Segment terminal (here: IB IL 24 SEG/F-D)
5 Branch terminal for integrating a Fieldline Modular local bus in an Inline station (here: IB IL 24 FLM-PAC)
6 Fieldline Modular local bus

2.4 Bus Coupler Wiring

2.4.1 Connecting the DeviceNet™ System

Connect the DeviceNet™ system to the bus coupler (see Figure 2-3) via a TWIN COMBICON connector. For the pin assignment, please refer to Table 2-2.

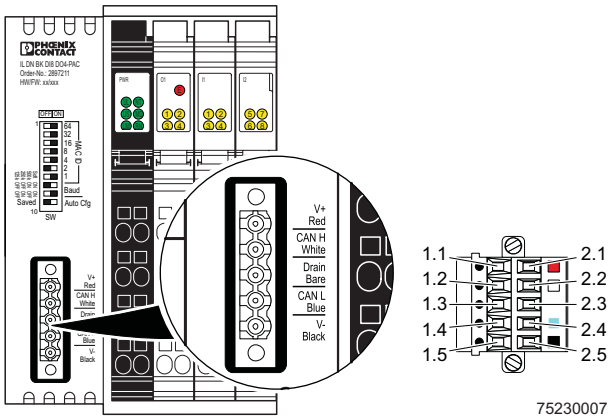


Figure 2-3 Pin assignment of the TWIN COMBICON connector

Table 2-2 Pin assignment of the TWIN COMBICON connector

Pin	Color	Signal	Description
1.1, 2.1	Red	V+	+24 V DC for U_L
1.2, 2.2	White	CAN H	CAN High
1.3, 2.3	Bare	Drain	Shield
1.4, 2.4	Blue	CAN L	CAN Low
1.5, 2.5	Black	V-	GND

2.4.2 Connection of Supply, Actuators, Sensors

Figure 2-4 and Table 2-3 to Table 2-6 lists terminal assignments for the bus coupler connectors. Figure 2-5 shows a wiring schematic for the power connector. Note that the bus coupler provides I/O power to the main (U_M) and segment (U_S) circuits for the Inline station. Communications/logic power and analog power (U_{ANA}) are also supplied by the bus coupler through the U_L connection.
The bus coupler has 3 external power supply connections U_L (logic), U_S (segment) and U_M (main). The 7.5 volt internal U_L (communications) supply and the +24 V U_{ANA} (analog) supply are derived from the external +24 V U_L. The +24 V (U_L) external power supply can be connected to DeviceNet™ or another external supply.

For information on special features of an Inline station, such as voltage supply, voltage distribution, and grounding, please refer to the IL SYS INST UM E user manual.

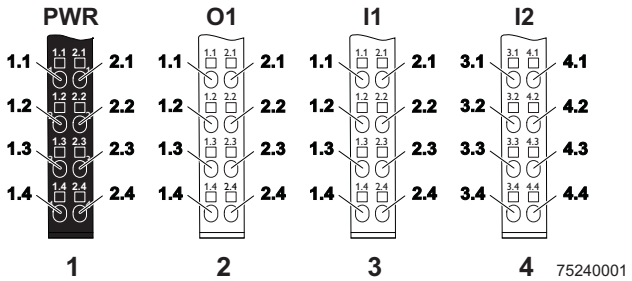


Figure 2-4 Terminal point assignment of the Inline connector

Table 2-3 Terminal point assignment of the power connector (1)

Terminal Points	Assignment	Terminal Points	Assignment
1.1	U_S	2.1	U_M
1.2	U_L	2.2	U_M
1.3	GND U_L	2.3	GND U_M, U_S
1.4	Functional earth ground (FE)	2.4	Functional earth ground (FE)

Table 2-4 Terminal point assignment of the output connector (2)

Terminal Points	Assignment	Terminal Points	Assignment
1.1	OUT1	2.1	OUT2
1.2	PGND	2.2	PGND
1.3	FE	2.3	FE
1.4	OUT3	2.4	OUT4

Table 2-5 Terminal point assignment of the input connector (3)

Terminal Points	Assignment	Terminal Points	Assignment
1.1	IN1	2.1	IN2
1.2	U_S	2.2	U_S
1.3	PGND	2.3	PGND
1.4	IN3	2.4	IN4

Table 2-6 Terminal point assignment of the input connector (4)

Terminal Points	Assignment	Terminal Points	Assignment
3.1	IN5	4.1	IN6
3.2	U_S	4.2	U_S
3.3	PGND	4.3	PGND
3.4	IN7	4.4	IN8

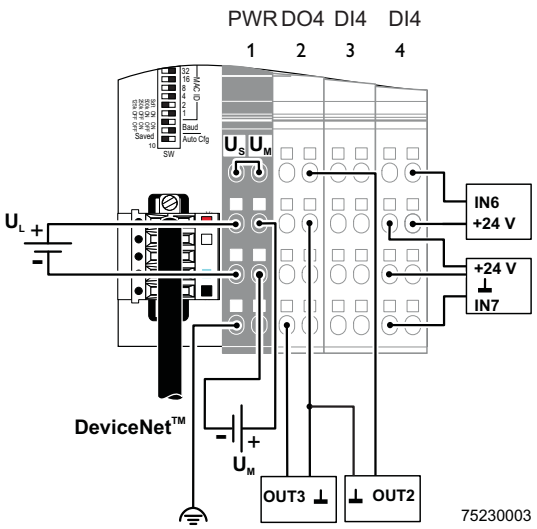


Figure 2-5 Connection example

2.4.3 Wire Type and Strip-Length Requirements

For the bus coupler network connector, use standard 5-wire, DeviceNet™ cabling. The bus coupler power connector uses wires with diameters of 0.22 to 1.5 mm^2 (AWG 24 - 16) and utilizes spring-cage technology for making gas-tight connections. See Figure 2-6 for strip-length requirements. Figure 2-7 shows how to install or remove wires from the terminal.

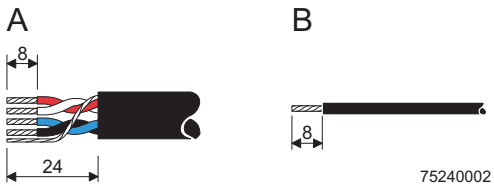


Figure 2-6 Strip-length requirements (dimensions in mm)

A: Network cable strip length

B: Wire strip length for Inline connectors

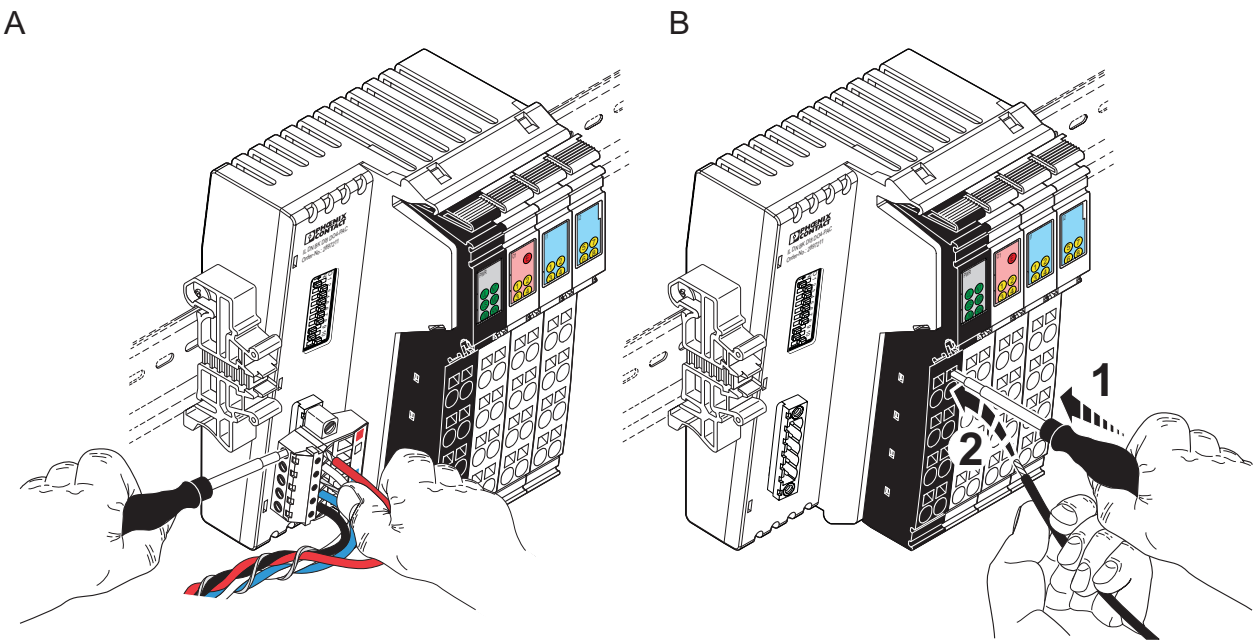


Figure 2-7 Installing and removing cables and wires

2.5 Network Considerations

2.5.1 Wiring

Figure 2-8 shows an example of setting up a standard trunk and drop network.

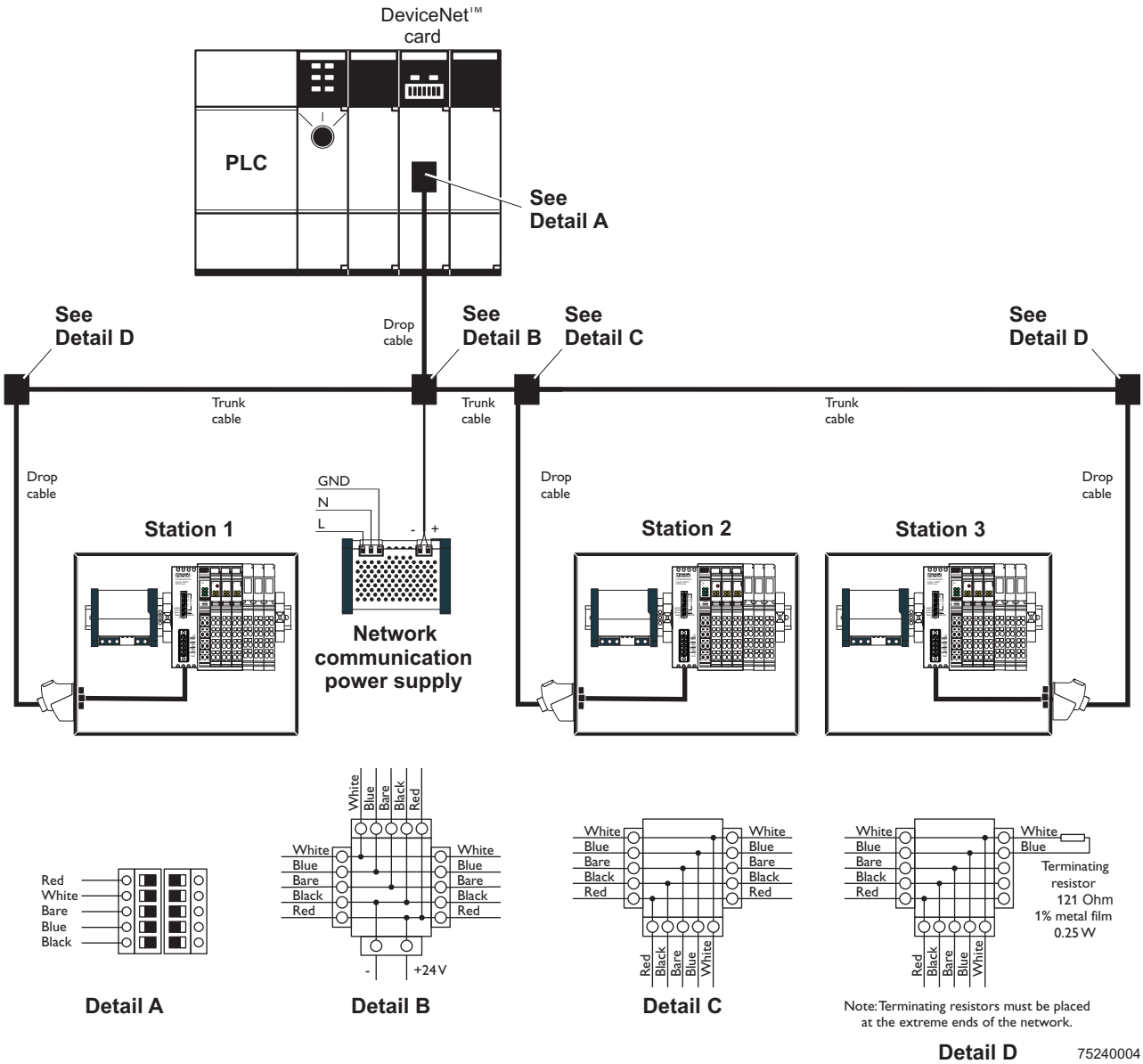


Figure 2-8 Example of setting up network communications and network power in an Inline DeviceNet™ system

Power

DeviceNet™ network power supplied to the bus coupler must be in the range of 11 V DC to 28 V DC. A supplied voltage of less than 11 V DC will generate an Inline status error (see Section 4, "Diagnostics"). Each bus coupler requires approximately 60 mA, at 24 V DC, of the total available network current.

To power Inline I/O module communications, DeviceNet™ power can be jumpered from the network connector, positions 1.1/2.1 (+V) and 1.5/2.5 (-V) to positions 1.2 (+24 V U_L) and 1.3 (U_L return) of the power connector. Inserting this jumper will draw additional current from the network. Refer to the I/O module specific data sheet to determine the exact additional U_L current draw.

Termination Resistors

A 121 Ω, 1% metal film, 0.25 W, termination resistor must be added at the extreme ends of the DeviceNet™ system. See Figure 2-8, Detail D.

Daisy Chaining

If daisy chaining is required, the network cable must be terminated to the network connector prior to installing the connector to the bus coupler.

The bus coupler 10-pin network connector contains two rows of contacts with 5 contacts in each row. Refer to Figure 2-3 on page 2-5. These two rows of contacts provide a means for daisy chaining additional Inline stations. See Figure 2-9.

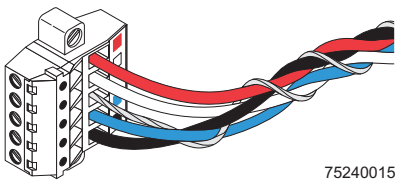


Figure 2-9 Termination of the network cable

2.5.2 Wire and Cable Central Grounding Specifications

Grounding protects human beings and machines from dangerous voltages. To avoid dangerous situations, it is essential that you always follow the grounding procedures outlined in the product-specific data sheet(s), this manual, and local codes.

DeviceNet™ communications power should only be grounded to earth at a single point. Typically this is done in the control cabinet where the DeviceNet™ power supply unit is located (+V, -V).

Grounding requirements for DeviceNet™ such as power return (-V), the drain (bare) wire, and the cable shields all need to be directly connected to earth ground. The ideal spot for making the earth ground connection for the entire network is in the physical center of the network. The ground connection should be made using either a 25 mm wide flat, braided, copper cable or a 10 mm^2 wire. The wire for this connection should be no longer than 3 meters.

Figure 2-10 shows an example of a system grounding scheme using a single power supply. Please note that the power supply's chassis is directly connected to earth ground.

NOTE:
DeviceNet™ components must be grounded to avoid possible signal interferences.

The Inline bus coupler is both an isolated physical layer and an I/O device. The only strictly DeviceNet™ grounding consideration is the drain (bare) wire. It can be terminated in position 1.3 or 2.3 of the network connector. Refer to Figure 2-3 on page 2-5 and Table 2-2 on page 2-5.

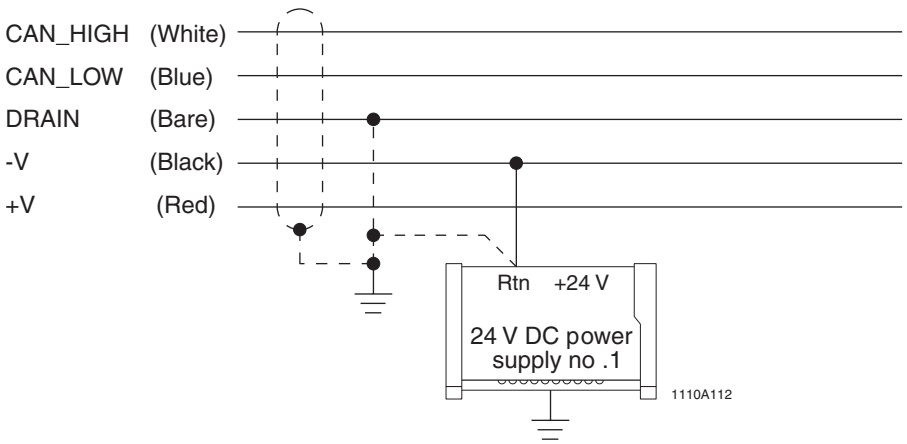


Figure 2-10 DeviceNet™ system grounding scheme

3 Configuration and Operation

3.1 Introduction

This section provides the quickest path to go online with the Inline DeviceNet™ system.

Each procedure in this section deals with information on how to install, operate and troubleshoot an Inline DeviceNet™ network.

Information in this section is presented with the assumption that the user has a working knowledge of DeviceNet™ and the RSNetWorx™ (or equivalent) configuration software. By default the bus coupler will be ready to exchange data in the polled I/O scan mode.

When this unit is first received it will be set up to use address 63 and a baud rate of 125K. This station is delivered for operation in fault response mode 0, Stop on Fail. In the Stop on Fail mode a local I/O failure (except for the peripheral fault) will cause all outputs to turn off. The PF bit is set in the Inline Status word when any output is shorted or during the loss of power to an intelligent segment module.

By default, Inline station diagnostics (Inline Status word) are included in the produced response. The PF bit is included in the diagnostic word and will need to be evaluated by the application software to determine the required action for the local station outputs.

3.2 DIP Switch Settings for Address (MAC ID) and Baud Rate

Each bus coupler has ten DIP switches. These switches are located on the left side of the DeviceNet™ bus coupler. See Figure 3-1. DIP switches 1 through 7 are used to set the MAC ID. DIP switches 8 and 9 are used to set the baud rate. DIP switch 10 sets the bus coupler configuration mode as described in Figure 3-1.

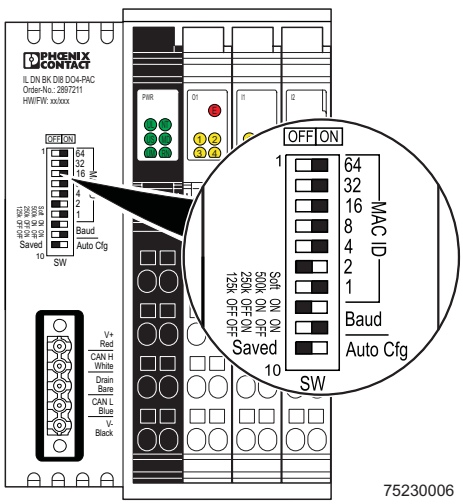


Figure 3-1 DeviceNet™ bus coupler DIP switches

3.2.1 Setting the Node Address (MAC ID)

The node address is set using DIP switches 1 through 7. DIP switch 7 is the least significant digit of the MAC ID and DIP switch 1 is the most significant digit. Valid MAC ID settings range from 0 to 63 .

Table 3-1 Node address (MAC ID)

	SW1	SW2	SW3	SW4	SW5	SW6	SW7
ON	64	32	16	8	4	2	1
OFF	0	0	0	0	0	0	0

Note that DeviceNet™ power needs to be cycled (OFF & ON) in order to implement any changes in a hardware selected node address (MAC ID).

3.2.2 Software Setting of the Node Address (MAC ID) Using Switches 1 Through 7

The software setting of the node address is also set using DIP switches 1 through 7. Any combination of DIP switch settings greater than 63 will allow the MAC ID to be set using software. This software setting is made by sending an explicit message using Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single) to the DeviceNet™ Object. Then set service parameters as follows.

Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single)

Parameter 1	Class Code 3
Parameter 2	Instance 1
Parameter 3	Attribute 1
Parameter 4	Data (Desired address 0 to 63)

The MAC ID software setting will default to the last valid physical setting of the DIP switches.

3.2.3 Setting the Baud Rate

The baud rate is set using DIP switches 8 and 9. DIP switch settings for various baud rates are shown in Table 3-2.

Table 3-2 Baud rates

SW8	SW9	Baud Rate
ON	ON	Software settable
ON	OFF	500 k baud
OFF	ON	250 k baud
OFF	OFF	125 k baud

Note that DeviceNet™ power needs to be cycled (OFF & ON) in order to implement any changes in a hardware selected baud rate.

3.2.4 Software Setting of the Baud Rate

The software setting for the baud rate is also set using DIP switches 8 and 9. Setting DIP switches 8 and 9 to the ON position allows the software to set the baud rate. The software setting is made by sending an explicit message using the Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single) to the DeviceNet™ Object. Then set service parameters as described below.

Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single)

Parameter 1	Class Code 3	
Parameter 2	Instance 1	
Parameter 3	Attribute 2	
Parameter 4	Data:	0 = 125 kbps
		1 = 250 kbps
		2 = 500 kbps

The software setting will default to the last valid physical DIP switch setting.

3.2.5 Configuration

With DIP switch 10, the I/O configuration mode of the bus coupler is set.

ON enables the bus coupler to auto-configure the I/O modules during the next power-up (**Auto Cfg**).

With **OFF**, the bus coupler retains the last valid I/O module configuration (**Saved**).

For configuration, see Section "Auto-Configuration Method" on page 3-8.

3.2.6 Restoring Factory Default Settings

Set DIP switches 1 through 7 to the ON position. Then apply power to the bus coupler to restore all settings to factory default.

3.2.7 Faulted Node Recovery (FNR)

DIP switches can be used to physically set the bus coupler's MAC ID. Faulted Node Recovery gives the user an alternative method by assigning the MAC ID using software. Typically, Faulted Node Recovery is used for devices that do not have mechanical switches to set the MAC ID.

Faulted Node Recovery (FNR) is the ability to recover two or more nodes connected to a DeviceNet™ network that have the same MAC ID. This includes sensors, actuators, drives and all other DeviceNet™ nodes. The advantage of FNR is that all devices on the DeviceNet™ network can be connected using the same MAC ID (same DIP switch settings) and then given a unique MAC ID through configuration software.

Example Procedure Using Inline Bus Couplers

If using FNR, use the following procedure:

1. Set all bus couplers to the same MAC ID. Value must be 64_{dec} or greater.

When installing bus couplers, the user may want to record the serial number of each bus coupler and its location on the network.

2. Install all bus couplers on the network.
3. Using the RSNetworx™ (or equivalent DeviceNet™ configuration package having Faulted Node Recovery capability) Faulted Address Recovery Wizard, go online to view all installed bus couplers.
4. From the displayed list of bus couplers (with serial numbers), use one of the following two methods for identifying and setting the unique MAC ID.

Method 1 - By Serial Number

1. Select the first bus coupler and serial number from the displayed list.
2. Match the serial number to its location on the network.
3. Assign an unused MAC ID to the bus coupler.
4. Continue assigning unused MAC IDs to remaining bus couplers.

Method 2 - By Visual Observation

1. Select the first bus coupler and serial number from the displayed list.
2. Press LED button to blink NT LED for selected bus coupler.
3. Visually search each bus coupler on the network to determine which NT LED is blinking.
4. Assign an unused MAC ID to the bus coupler.
5. Continue assigning unused MAC IDs to remaining bus couplers.

3.3 Determining I/O Module Capacity

The bus coupler is capable of processing the maximum number of instances (points) for DeviceNet™/vendor-specific objects listed below.

Digital Input Points (DIP)	510 instances
Digital Output Points (DOP)	510 instances
Analog Inputs Points (AIP)	128 instances
Analog Output Points (AOP)	128 instances
Special Function Object	63 instances
PCP Special Function Object	8 instances*
Serial Communication Object	8 instances*

*Total instances shared between the PCP Special Function and Serial Communication Objects can not exceed 8.

There are certain considerations that must be observed when determining the number of I/O devices that can be connected to the bus coupler. These considerations are described in the following paragraphs.

1. The bus coupler cannot provide more than 0.8 A of communications/logic power (U_L).

If U_L power requires more than 0.8 A a IB IL 24 PWR IN/R terminal can be inserted to reinject U_L power.

2. The maximum number of devices connected to the bus coupler cannot exceed 63.
3. Analog modules cannot draw more than 0.5 A from the analog supply (U_{ANA}). Note that analog modules also require current from the 0.8 A communications/logic supply (U_L).
4. FLM devices
   a. Only the IB IL 24 FLM-PAC uses the logic supply (U_L).
   b. FLM I/O devices will use current from the segment power (U_S); see documentation for the IB IL 24 FLM-PAC.
   c. The IB IL 24 FLM does not count towards the 63 device maximum.
   d. Each FLM I/O device does count towards the 63 device maximum.

There are 0.8 A of the current available on the Inline communications supply (U_L) and 0.5 A of current available on the analog supply (U_{ANA}). Refer to the specific I/O module's data sheet or the Phoenix Contact Automation Catalog to determine the amount of current draw required for each I/O module or device to be connected to the bus coupler. The total current consumption of all modules/devices cannot exceed the 0.8 A and 0.5 A ratings stated above.

3.4 Configuring the Inline Station

If using a serial or another type of PCP module, read this section then refer to Section "Serial and Other PCP Inline Modules" on page A-1.

3.4.1 Bus Coupler

Configuration of the bus coupler allows it to read and communicate specific information about the I/O modules connected on its local bus (backplane). Specific local bus information includes:

- How many I/O modules are on the local bus
- Position of the I/O modules on the local bus
- How many points or channels (instances) each module contains
- Bytes of produced and consumed data
  Types of I/O modules:
  - Digital input
  - Digital output
  - Analog input
  - Analog output
  - Special function (incremental encoder, absolute encoder, high speed counter, other)
  - PCP and serial (AS-i master, RS-232, RS-485, and others)

Any module that is not recognized will be placed into the special function category.

When configuring the Inline DeviceNet™ bus coupler, it is recommended that you remove the connection to the DeviceNet™ scanner or make sure the scan list for the DeviceNet™ scanner is empty.

3.4.1.1 Configuration Methods

When creating, adding to, or changing an Inline DeviceNet™ station, the I/O configuration stored in the bus coupler must be updated to match the new configuration of the station.

Configure the bus coupler using one of the following 3 methods

- The Electronic Data Sheet (EDS) file
- Auto-configuration (no software required)
- Sending an explicit message

The Electronic Data Sheet (EDS) File Method

The EDS file is the software interface between the bus coupler and a DeviceNet™ configuration software package such as RSNetWorx™. The EDS file contains information about the number of produced and consumed bytes. It also provides the user-settable parameters such as what should be included under poll options.

Section C, "Electronic Data Sheet (EDS) File" in this manual describes the EDS file and each of the EDS file parameters.

EDS file parameter 9, (Add All I/O) must be used for new or changed configurations when I/O modules are to be included in the scan.

The following procedure describes how to configure a bus coupler using the Electronic Data Sheet. Repeat this procedure for each bus coupler on the network.

1. Obtain a list of I/O modules that will be used in the DeviceNet™ station.
2. Determine which types of I/O modules will be included in the scan. By default, all modules will be included. If the default needs to be altered, a new value must be downloaded to the bus coupler through EDS parameter 8 (Add All Mode).
3. Determine whether the "Inline Status" (diagnostic) word needs to be included in the produced size. By default, these 2 bytes will be added to the produced size. By using the produced data channel, the Inline Status word gives the user the ability to locate and define any faults that could occur on the Inline local bus. If the user decides to disable this feature, then the user must download a 0 to the bus coupler using parameter 1 (Use Inline Status) in the EDS file.
4. Determine whether analog or special function modules are used. If used, the user has the option to set parameter 2 (Pad I/O) to a 0 or 1 (default). If not used, proceed to step 5.

Depending on the I/O configuration, the analog or special function data may start on a odd byte. If this is the case, it is possible that this word could span two words in the master scanner. By setting the parameter 2 (Pad I/O) to 1 (default) one byte will be added to the produced/consumed size, thereby, forcing the first word based module to start on an even byte. If this word already starts on an even byte and the user sets parameter 2 (Pad I/O) to 1 (default), no additional bytes will be added to the produced /consumed size.

5. If future system expansions are anticipated, determine if there is a need to reserve digital points (bits) or analog words in the scan. If so, determine the number of points (bits) or words to be reserved, then add to this the number of points or words currently being used. For more information, refer to Section "Reserving I/O Memory for Future System Expansion" on page 3-21. This total number of points or words must be downloaded to the bus coupler. Reserving points or words in local I/O memory is accomplished by using parameter 3 (Reserve Digital Inputs), parameter 4 (Reserve Digital Outputs), parameter 5 (Reserve Analog Inputs) and parameter 6 (Reserve Analog Outputs).
6. Enter the new station configuration parameters into the flash memory of the bus coupler by changing the default value of parameter 9 (Add All I/O) from a 0 (False) to a 1 (True). Depending on the size of the local bus, the user may have to wait until the downloading of the new configuration is completed. Completion of the download can be determined by observing the state of the "MD" LED on the bus coupler and the "D" LEDs on the I/O modules. While downloading, the LEDs will blink. Once the download is completed, the blinking stops and the new configuration is now stored in the flash memory of the bus coupler.

Auto-Configuration Method

Auto-configuration allows for in-the-field configuration of the bus coupler without any software. Defaults settings entered under auto-configuration are:

– Two bytes will be added to the produced size for the Inline Status word (diagnostic word).
– All modules will be added to the produced and consumed response as dictated by their class.
– Pad I/O will be used.
– No digital or analog I/O reservations for future system expansion will be made.

An explanation of each of these EDS parameters can be found in Section C, "Electronic Data Sheet (EDS) File".

The following procedure describes how to use the auto-configuration method.

1. Set DIP switches 1 through 7 to the desired address. Set DIP switches 8 and 9 to the desired baud rate. Set DIP switch 10 to "ON" to allow auto-configuration.
2. Check that all required I/O devices are connected.
3. Power up the bus coupler using network and U_L (communication/logic) power. Note that at initial power up, the new address and baud rate are automatically stored in the bus coupler's flash memory.
4. Set DIP switch 10 to the "OFF" position and power cycle to store current configuration.

At this point both the "MD" LED on the bus coupler and the "D" LEDs on the I/O modules should begin blinking. Once the LEDs stop blinking and remain ON, storage of the I/O configuration in the bus coupler's flash memory is completed.

Sending an Explicit Message Method

When configuring the bus coupler of an Inline station by sending an explicit message, observe the decisions stated under Section "The Electronic Data Sheet (EDS) File Method" on page 3-7. The parameters listed in this section can also be configured by sending multiple explicit messages to the Configuration Object (Class Code 100_{dec}, 64_{hex}).

When using the explicit message method to change defaults, the following command structure must be used to configure the DeviceNet™ bus coupler.

Service Code	16_{dec}, 10_{hex}
Class Code	100_{dec}, 64_{hex}
Instance	1
Attribute	X (X = Attribute to be changed)
Attribute Data	1

If no default settings need to be changed, you still must send one explicit message using the following command structure to configure the DeviceNet™ bus coupler.

Service Code	16_{dec}, 10_{hex}
Class Code	100_{dec}, 64_{hex}
Instance	1
Attribute	9 (Add All I/O)
Attribute Data	1

Setting Attribute 9 (Add All I/O) to a 1 instructs the bus coupler to scan its local bus and store its current configuration into the bus coupler's flash memory. This configuration will remain in flash memory until the next "Add All I/O" is sent or until a different configuration method is used.

The service that allows Attribute 9 to be set is Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single). Object classes, and services are described in Section B, "DeviceNet™ Object Classes, Message Types and Services". The RSNetWorx™ or an equivalent software package can be used to send explicit messages.

3.4.2 Analog Input (AI) Modules

3.4.2.1 General Configuration

Non-multiplexed (standard) analog input (AI) modules default to a unipolar range of 0 V DC to +10 V DC. To change this range, the range attribute can be set in the Analog Input Point (AIP) Object (Class Code 10_{dec}, $0A_{hex}$). Set the range by sending an explicit message using the "Class Instance Editor" in RSNetWorx™. Additionally, attributes 100 and 101 could be used to write a custom configuration value to the analog input module. Refer to Section "Thermocouple and RTD Modules" on page 3-10 to determine how to use attributes 100 and 101. Enabling attribute 101 will override any range settings.

Once the range is set for the new value, the bus coupler will retain that setting in flash memory. If the bus coupler is replaced the configuration will need to be redone.

Section B, "DeviceNet™ Object Classes, Message Types and Services" provides details of the AIP Object (Class Code 10_{dec}, $0A_{hex}$) range settings.

Multiplexed analog input (AI) modules such as the AI8 will appear to have only as many channels as the module has data words. The BK has no way of determining how many channels are contained within those words. By default, the control words for an analog input module are not placed in the poll. However, with multiplexed modules, it is often desirable to change the control word frequently. The user can instruct the BK to place these control words into the poll. This is done by enabling attribute 102 "AIP configuration word in poll" of the AIP object. Enabling attribute 102 will override both attribute 101 and any range settings.

3.4.3 Thermocouple and RTD Modules

3.4.3.1 General Configuration

This section describes how to change default settings for the Inline 2-channel thermocouple module. However, changing the default settings for the 2-channel RTD modules is accomplished in the same manner.

There is no "Thermocouple" or "RTD Object". To change default settings, the Analog Input Points (AIPs) Object (Class Code 10_{dec}, $0A_{hex}$) must be used. Refer to Section B, "DeviceNet™ Object Classes, Message Types and Services".

Default settings for the thermocouple modules are:

Sensor type:	K
Resolution:	0.1°C (1 microvolt)
Output format:	15 bits and 1 sign bit with extended diagnostics
Cold junction:	Internal

In order to change any setting, refer to the thermocouple data sheet to determine the appropriate attribute settings for the AIP Object (Class Code 10_{dec}, $0A_{hex}$).

Keep in mind that thermocouple or RTD instances will appear as analog input instances. There are 2 instances for every thermocouple or RTD module. Channel 0 will be the first instance and channel 1 will be the second.

The user must keep track of which instances are analog inputs and which instances are thermocouple inputs. Figure 3-2 shows a station where instances 1 and 2 of the AIP are used by the 2-channel thermocouple. Instances 3, 4, 5 and 6 of the AIP are used by the next two, 2-channel analog input modules. Instances 7 and 8 of the AIP are used by the next 2-channel thermocouple module. The last two, 2-channel analog input modules occupy instances 9,10,11 and 12.

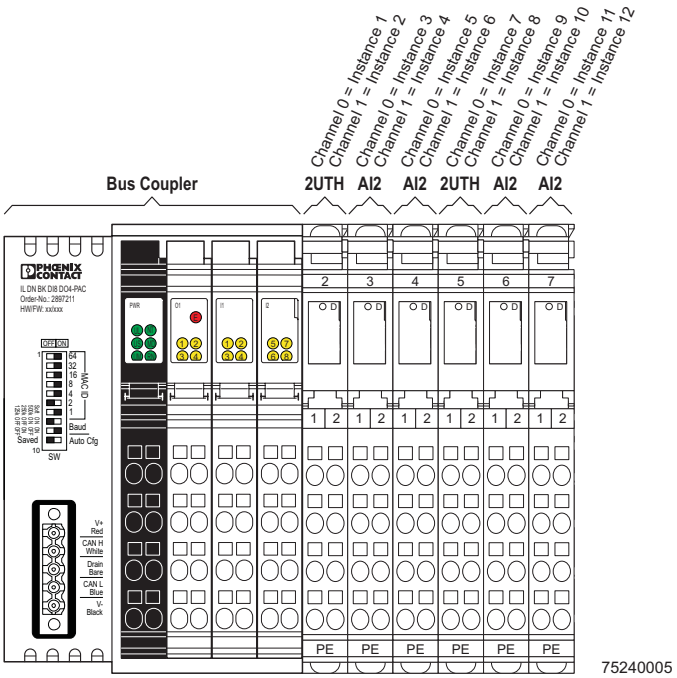


Figure 3-2 I/O station with analog input terminals and thermocouple terminals

Default settings are modified by sending an explicit message to a specific instance. This message can be sent using the "Class, Instance Editor" window in RSNetWorx™.

Using attribute 100 along with the correct instance in the AIP object is one way to determine how a thermocouple, RTD, or analog input module is configured.

If Attribute 100 = 0 (default) in the AIP, the user has access to all standard AIP attributes. If Attribute 100 = 1 the user has access to Attribute 101 (Input Configuration word). By assigning a value to the Attribute 101, the user will be able to configure the thermocouple or RTD module(s). The correct configuration value for attribute 101 can be determined by using the module specific data sheets for the thermocouple, or RTD modules. It is also possible to use the same method as multiplexed modules, whereby the configuration word is placed permanently into the poll data.

Once the new thermocouple setting is made, the new configuration will be stored in the flash memory of the bus coupler. If the bus coupler is replaced the configuration will need to be redone.

AIP Object (Class Code 10_{dec}, $0A_{hex}$) settings are described in Section B, "DeviceNet™ Object Classes, Message Types and Services".

3.4.4 Special Function Modules

3.4.4.1 General Configuration

Special function modules such as the incremental encoder, absolute encoder and the high-speed counter are configurable through the polled data channel by default. Output word(s) that are assigned to the special function module are used to program the terminal. The user must refer to the specific special function module's data sheet or manual to determine what codes need to be written to the associated output word(s).

If the special function module is not included in the poll, it can be programmed through the use of explicit message Special Function Object (Class Code 103_{dec}, 67_{hex}).

The programming of a special function module cannot be stored in the bus coupler flash memory. The user's application will have to implement a programming subroutine.

3.4.5 Using RSNetWorx™

3.4.5.1 Adding the EDS File

The EDS file for the DeviceNet™ bus coupler can be added to the RSNetWorx™ as described in the following procedure.

1. From the RSNetWorx™ menu bar, select "Tools". Then from the "Tools" drop-down list select "Install EDS Wizard".

EDS, BMP and ICO files for DeviceNet™ are available on the Phoenix Contact web site at www.phoenixcon.com, under the InfoService link.

2. Follow the EDS Wizard options to register an EDS file.
3. In the EDS Wizard, browse for and select the bus coupler EDS file. Then select the proper icon. This will add the EDS and visual icon to RSNetWorx™.

3.4.5.2 Setting the Node Address

This section may be skipped if using the DIP switches to set the actual node address.

Make sure DIP switches 1 through 7 are set to a node address that is greater than 63. The following procedure requires that the A-B 1770-KFD module (or equivalent) is connected to the DeviceNet™ bus coupler.

1. From the RSNetWorx™ menu bar, select "Network" then "Online". Form the "Online" window the appropriate interface tool driver can be selected. Figure 3-3.
2. From the "Browse for network" window, select the previously installed in RSLinx™ driver that you want to use. For discussion purposes in this procedure, we will be using the A-B 1770-KFD RS-232 interface V1.50. Once the adapter is selected, click OK.

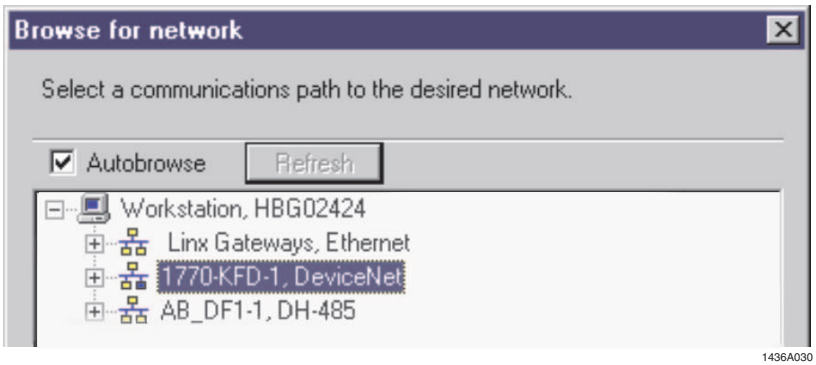


Figure 3-3 RSNetWorx™ Browse for network window

3. Once the driver is selected, click OK. The network browse will start. Make sure the connection status icon (A in Figure 3-4) shows a moving on-line connection. Figure 3-4.
4. Wait for the browse operation to be completed. In our example, the network shows both the icon for the KFD module and the icon for the DeviceNet™ bus coupler. See Figure 3-4. Note the last valid node address setting for the bus coupler (in our example) is 2.

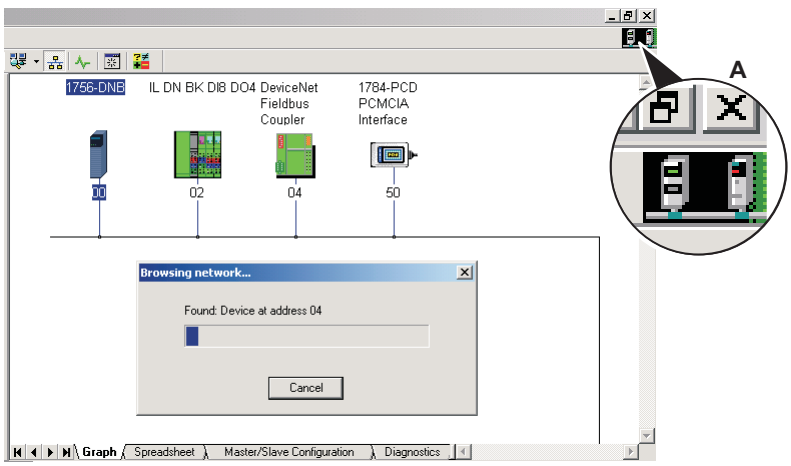


Figure 3-4 Network Browse results

5. Click on the bus coupler icon so that it is highlighted.
6. From the Device menu, choose "Class Instance Editor". At this point the "Class Instance Editor" window will appear. See Figure 3-5.

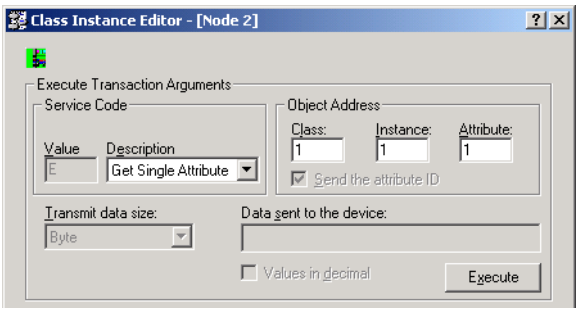


Figure 3-5 Class Instance Editor window

7. From the "Class Instance Editor" window, Locate the proper text boxes and enter the following:

Class:	3
Instance:	1
Attribute:	1

8. Locate the "Data sent to the device" text box and enter your desired address. In our example, the address is being changed to 3. Make sure the data type is sent to a byte.
9. Locate the "Service Code" drop down menu, make sure that it is set to "Set Attribute Single".
10. Click on the "Execute" button to download the new address parameter. Thereafter, the next time you choose the "Browse" option, the new node address will appear.

3.4.5.3 Setting the Baud Rate

This section may be skipped if the DIP switches are being set to the actual baud rate.

With the A-B 1770-KFD module (or equivalent) connected to the Inline DeviceNet™ bus coupler, set DIP switches 8 and 9 to the ON position. Then cycle power. Set the new baud rate as described in the following procedure.

1. From the RSNetWorx™ menu bar, select "Network" then choose "Online". Form the "Online" window (refer to Figure 3-3), select the appropriate interface tool driver.
2. From the "Browse for network" window, select the previously installed in RSLinx™ driver that you want to use. For discussion purposes, this procedure will be using the A-B 1770-KFD RS-232 Interface V1.50. Once the adapter is selected, click OK.
3. Once the driver is selected, click OK. The network browse will start. Make sure the connection status icon shows that there is a moving online connection. Refer to Figure 3-4.

To go online, the initial baud rate must match the last valid DIP switch setting.

4. Wait for the browse operation to be completed. In our example, the network shows both the icon for the KFD module and the icon for the DeviceNet™ bus coupler. Refer to Figure 3-4.
5. Click on the bus coupler icon so that it is highlighted.
6. From the Device menu, choose "Class Instance Editor". At this point the "Class Instance Editor" window will appear. Refer to Figure 3-5.

7. From the "Class Instance Editor" window, locate the proper text boxes and enter the following:

Class:	3
Instance:	1
Attribute:	2

8. Locate the "Data sent to the device" text box. Then enter your desired baud rate. In our example, the baud rate is being changed to 500K.

   0 = 125 kbps

   1 = 250 kbps

   2 = 500 kbps

9. Locate the "Service Code" drop down menu, make sure that it is set to "Set Attribute Single".
10. Click on the "Execute" button to download new baud rate.
11. Reconnect to the bus coupler using the new baud rate as described in Steps 1 to 4.

3.4.5.4 Adding the I/O to the Bus Coupler's Memory

If the bus coupler EDS file is not present in the Device List under Generic\Phoenix Contact\., install the EDS file as previously described.

1. Make an online connection to the DeviceNet™ bus coupler using the A-B KFD module (or equivalent).

NOTE:

To prevent communication errors to the device, make sure that the device is not included in the master controllers scan list, or unplug the network cable from the scanner.

At this point the user must know what EDS parameters are required for the application. When determining parameters, consider parameters 1, 2 and 9. Parameter 9 is mandatory for the I/O configuration using the EDS file. See Section C, "Electronic Data Sheet (EDS) File" for a description of the EDS file parameters.

2. Double click the Inline icon. Then select the "Parameters" tab. At this point the EDS file parameters window will be displayed. See Figure 3-6.

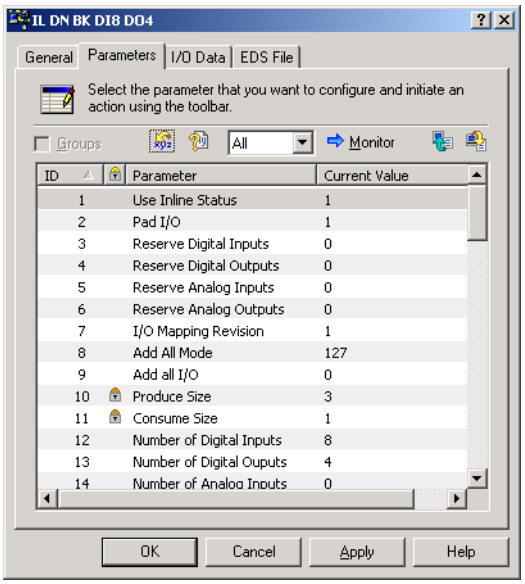


Figure 3-6 EDS File Parameters window

3. Ask yourself the following two questions.

   Question 1: Should the Inline status word (EDS parameter 1) be added to the produced data?

   If the answer is NO, double-click on parameter 1 (Use Inline status). Set the value to a 0. Then download the parameter to the device. Proceed to question 2.

   If the answer is YES, double-click on parameter 1. Set the value to a 1 (default) and download. By default the Inline status word will be added to the produced data size (adds two bytes to the produced size). Proceed to question 2.

   Question 2: Should the word oriented modules start on a even byte in the I/O memory?

   If the answer is NO, double-click on parameter 2. Set value to a 0. Then download to the device. Proceed to Step 4.

   If the answer is YES, double-click on parameter 2 (Pad I/O). Set value to a 1 (default) and download. This may add 1 byte to the produced/consumed size. Proceed to Step 4.

4. Double click on parameter 9 (Add All I/O). Set to 1, then download to the device.

The bus coupler will retain its last known valid I/O configuration. The last configuration is stored in flash memory. If an
I/O module has been added or deleted from the Inline station, the DeviceNet™ bus coupler must be reconfigured using parameter 9 (Add All I/O). Parameter 9 will ensure that the new configuration is stored in flash memory. Optionally, DIP switch 10 can be set to the ON position and the station can be power cycled to auto-configure the bus coupler.

3.5 Understanding I/O Memory Mapping

3.5.1 Bus Coupler Mapping

The I/O image in the bus coupler flash memory contains all produced data (input data) and consumed data (output data) derived from the I/O modules connected to it. I/O image data is added to the poll through the use of parameter 9 (Add All I/O). Configuration through EDS and RSNetWorx™ is explained in Section 3.4, "Configuring the Inline Station".

An I/O image could contain the following produced and consumed elements in the priority order listed below.

Produced

1. Inline Status
2. DIP Faults
3. DOP Faults
4. AIP Faults
5. AOP Faults
6. Special Function Faults
7. DIP States (values)
8. DIP Reserve
9. PAD Byte (optional)
10. AIP States (values)
11. AIP Reserve
12. Special Function
13. PCP Special Function (process & fragment)
14. Serial Communication (process & fragment)

Consumed

1. Inline Control Byte
2. DOP States (values)
3. DOP Reserve Points
4. PAD Byte (optional)
5. AOP States (values)
6. AOP Reserved Words (optional)
7. AIP Configuration Words (optional)
8. Special Function
9. PCP Special Function (process & fragment)
10. Serial Communication (process & fragment)

All produced elements of the same priority will be mapped together regardless of their location. However, their relative location to the BK will be used to determine their instance values (sequential ordering). This same approach applies to consumed elements.

Analog channels will start at the first completely unused byte after the last digital module. If the total number of digital points of the same image is not modulus 8, there will be unused bits between the digital data area and the analog data area.

Depending on what I/O modules are connected to the bus coupler determines whether or not analog data starts on an even or odd byte. In those cases when analog data starts on an odd byte, analog data will span two words in the master scanner. If you prefer to have analog data to start on an even byte, set the EDS parameter 2 (Pad I/O) to a 1. Then download to the bus coupler.

This will prevent analog data from starting on an odd byte without regard to the I/O modules connected to the bus coupler. Once parameter 2 is set, one byte of unused I/O data may be added to the produced/consumed size. This byte of unused I/O data will force the analog word to always start on an even byte in the master scanner. If the physical configuration dictates that the analog word starts on an even byte and parameter 2 (Pad I/O) will not add a byte of data to the I/O data size.

The physical order of data in the I/O table is determined by the position of the modules on the local bus. The first module connected to the Inline DeviceNet™ bus coupler will reside in the first I/O byte (keeping in mind the "which data type comes first" rule). Furthermore, the LSB of the first module will be assigned to the first instance. The next module of the same type and image will line up next to the first module without leaving any “gaps” in the I/O table.

The example in Figure 3-7 consist of the following modules:

– Inline bus coupler with 4 digital outputs and 8 digital inputs onboard
– Inline terminal with 8 digital outputs
– Inline terminal with 2 digital output
– Inline terminal with 1 analog output.

Figure 3-7 shows an example I/O table (memory map) for the station shown below. The total amount of input bytes (Inline Status Word) would be 3 and the total amount of output bytes would be 4.

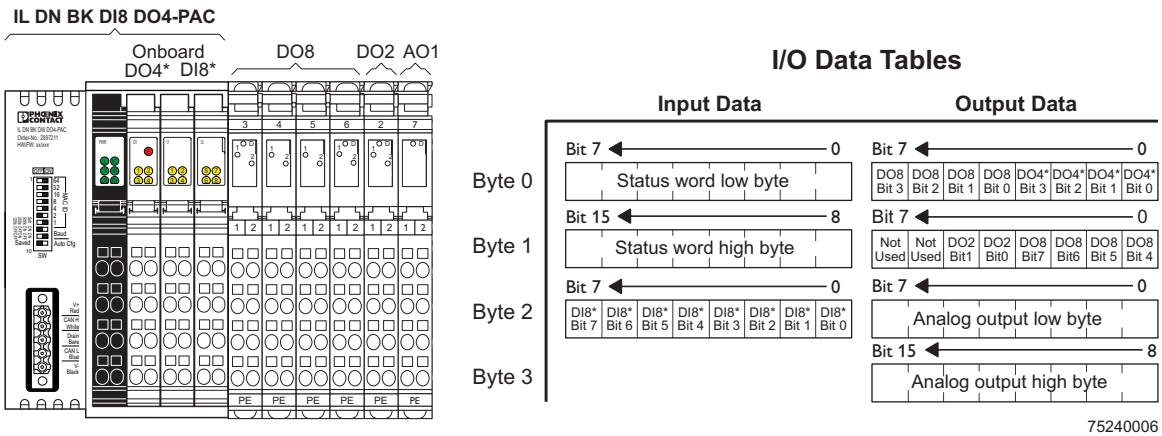


Figure 3-7 Example of an I/O table (memory map) consisting of analog and digital output modules

Figure 3-8 shows an Inline station and an example I/O table. The station consists of the following modules:

– Inline terminal with 2 digital inputs
– Inline terminal with 1 analog output
– Inline terminal with 2 digital outputs
– Inline terminal with 4 digital inputs
– Inline terminal with 2 analog inputs
– Inline terminal with 4 digital outputs
– Inline counter terminal
– Inline power terminals

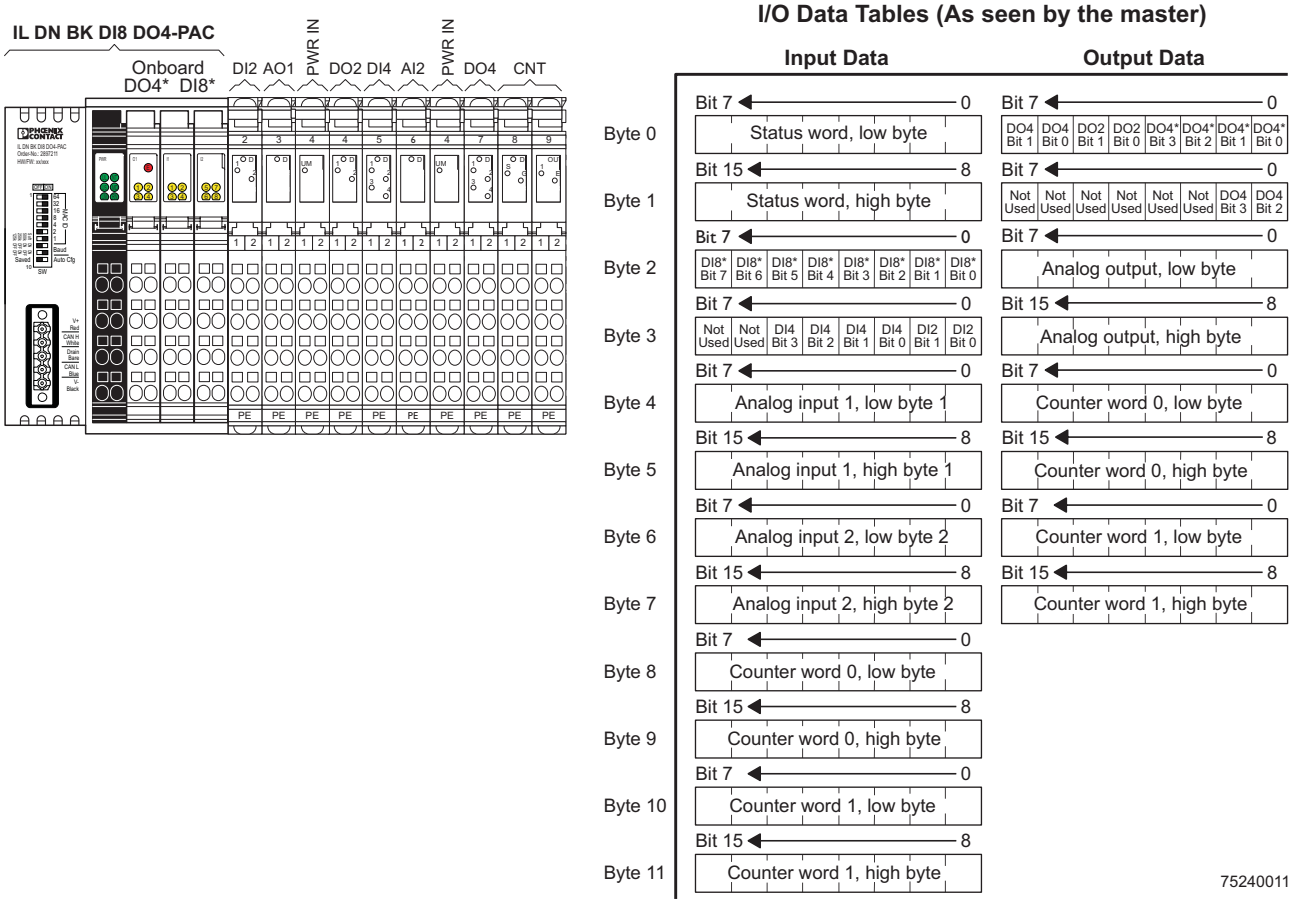


Figure 3-8 Example of an I/O table (memory map) consisting of digital and analog, input and output modules

The I/O configuration is for data mapping example only. Please follow the installation guidelines in the IL SYS INST UM E user manual.

Figure 3-8 shows that the total number of input bytes is 12 (byte 0 through byte 11). This includes the Inline Status word. Figure 3-8 also shows that the total number of output bytes is 8 (byte 0 to byte 7).

In our example, 11 and 8 were entered in the "Polled" box of the "Edit I/O Parameters" window. See Figure 3-9. The 11 (bytes) would be entered in "Rx" text box and the 8 (bytes) would be entered in the "Tx" text box.

The "Edit I/O Parameters" window can be opened as described in the following procedure:

1. Right click the scanner icon.
2. Select "Properties".
3. Select "Scanlist" tab then click "Upload".
4. Add an available device or select a device from the scan list to edit.
5. Select "Edit I/O Parameters".

When determining the total number of bytes to be configured for the DeviceNet™ bus coupler, the Inline status word must be added to the number of input bytes. Configured produced and consumed size can be read online by double-clicking on the device icon then uploading the parameters.

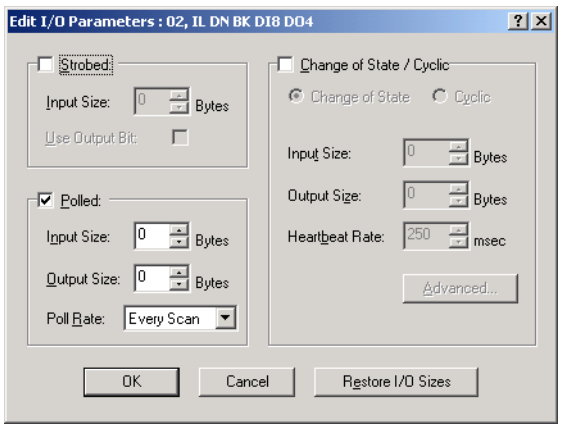


Figure 3-9 Edit I/O Parameters window

3.5.2 Reserving I/O Memory for Future System Expansion

Memory reservation is only available for digital and analog modules. It is not required for special function modules.

Rules for Reserving I/O Memory

1. Reserved I/O points will take up physical space in the produced and/or consumed data.
2. After reserving digital and/or analog I/O, any new modules added must be connected after (anywhere to the right of) the last digital/analog module of the same type and image (input or output) on the local bus.
3. If special function modules are added, they must be added after the right most special function module on the station. After adding the special function module, the station must be reconfigured to add the special function data to the scan. This additional data will not effect existing mapping of the master scanner.
4. If you want to reserve space for analog input configuration words, you must add the number of analog input configuration words to be reserved to the number of analog output words to be reserved. This will be the total number of analog output words to be reserved.
   When adding modules to this shared reserve space, analog output words will be added to the lower end of this space and analog input configuration words will be added to the upper end of this space until the entire space is used. For information about analog input configuration words, refer to the Section "Analog Input (AI) Modules" on page 3-9.

Ways to Reserve I/O Memory

The bus coupler can reserve digital or analog I/O in either the input or output image. This will allow for future system expansion(s) without having to change the master scanner's I/O tables. The actual reservation can be done in the following two ways:

1. By using EDS file parameter 3 (Reserve Digital Inputs), parameter 4 (Reserve Digital Outputs), parameter 5 (Reserve Analog Inputs), parameter 6 (Reserve Analog Outputs). The entry downloaded to the bus coupler will be equal to the current physical number of I/O points on the local bus, plus the number of I/O points to be reserved.
2. By sending an explicit message to the Configuration Object (Class Code 100_{dec}, 64_{hex}). The user can reserve a digital bit by writing to parameters 22 (Reserve Digital Inputs), 23 (Reserve Digital Outputs), 35 (Reserve Analog Inputs) and 36 (Reserve Analog Outputs). The entry downloaded to the bus coupler will be equal to the current physical number of I/O points on the local bus, plus the number of I/O points to be reserved.

3.5.3 I/O Mapping Revision

This parameter controls which DeviceNet™ I/O class Inline modules will be placed into. The default (1) will use the latest information stored in the firmware. A Mapping Revision of "0" will use mapping information that is compatible with the IL DN BK 1 DeviceNet™ bus coupler. For example, on the IL DN BK1, the IB IL 24 AI8, and IB IL AO2/BP modules would be classified as Special Function modules, whereas, on the IL DN BK DI8 DO4-PAC they would be considered analog input and output modules respectively. Any modules not identified will be placed into the Special Function (SF) category.

Table 3-3 Inline module IDs mapped to DeviceNet™ objects

Module ID Code (Dec)	Module ID Code (Hex)	Type of Inline Module	Mapping Revision 0 IL DN BK1 (FW 2.xx & 5.xx)	Mapping Revision 1 IL DN BK DI8 DO4-PAC (FW 1.xx)	Mapping Revision 2 IL DN BK DI8 DO4-PAC (FW 1.xx)
83	53	Analog input/output with parameter data	SF	AIP, AOP	AIP, AOP
91	5B	Analog output with parameter data	SF	AOP	AOP
95	5F	Analog input with parameter data	SF	AIP	AIP
99	63	Analog input/output with parameter data	SF	AIP, AOP	AIP, AOP
102	66	ENCOM appliance	SF	AIP	AIP
103	67	ENCOM appliance	SF	SF	SF
107	6B	Analog output with parameter data	SF	AOP	AOP
111	6F	Analog input with parameter data	SF	AIP	AIP
113	71	Analog output	AOP	AOP	AOP
114	72	Analog input	AIP	AIP	AIP
115	73	Analog input/output	AIP	AIP	AIP
117	75	Analog output for Safety	AOP	AOP	AOP
118	76	Analog input for Safety	SF	AIP	AIP
119	77	Analog input/output for Safety	SF	SF	SF
121	79	Analog output with profile	SF	AOP	AOP
122	7A	Analog input with profile	SF	AIP	AIP
123	7B	Analog input/output with profile	SF	SF	SF
125	7D	Analog output	AOP	AOP	AOP
126	7E	Analog input	AIP	AIP	AIP
127	7F	Analog input/output	AIP	AIP	AIP
173	AD	Digital output for Safety	SF	DOP	DOP
174	AE	Digital input for Safety	SF	DIP	DIP
175	AF	Digital input/output for Safety	SF	DIP, DOP	DIP, DOP
177	B1	Digital output	DOP	DOP	DOP
178	B2	Digital input	DIP	DIP	DIP
179	B3	Digital input/output	DIP, DOP	DIP, DOP	DIP, DOP
181	B5	Digital output with profile	SF	DOP	DOP

Table 3-3 Inline module IDs mapped to DeviceNet™ objects (continued)

Module ID Code (Dec)	Module ID Code (Hex)	Type of Inline Module	Mapping Revision 0 IL DN BK1 (FW 2.xx & 5.xx)	Mapping Revision 1 IL DN BK DI8 DO4-PAC (FW 1.xx)	Mapping Revision 2 IL DN BK DI8 DO4-PAC (FW 1.xx)
182	B6	Digital input with profile	SF	DIP	DIP
183	B7	Digital input/output with profile	SF	SF	SF
185	B9	Digital output with bus master special treatment	SF	DOP	DOP
186	BA	Digital input with bus master special treatment	SF	DIP	DIP
189	BD	Digital output	DOP	DOP	DOP
190	BE	Digital input	DIP	DIP	DIP
191	BF	Digital input/output with < 1 word of data, Special function with ≥ 1 word of data	SF	SF	DIDO, SF

3.6 I/O Data Transfer

A detailed explanation of the following objects and their attributes can be found in Section B, "DeviceNet™ Object Classes, Message Types and Services".

I/O data transfer can be accomplished by establishing a I/O connection (implicit) or by establishing a message connection (explicit).

Implicit

An implicit connection provides a dedicated path from a producing application to one or more consuming applications and is typically handled in a master with slave(s) relationship.

Explicit

An explicit connection is a generic connection between two devices where a request is sent and a acknowledgement is expected.

The bus coupler supports the Unconnected Message Manager (UCMM), which provides for the dynamic establishment of messaging connections.

3.6.1 I/O Scan Methods

The DeviceNet™ master will scan the bus coupler through the use of several implicit/explicit I/O scan types. The following scan methods are available to the user:

- Polled
- Explicit
- Change-of-State (COS)
- Cyclic
- Bit strobe
- I/O peer-to-peer
- Explicit peer-to-peer

The bus coupler can support 10 DeviceNet™ connections:

- 3 dynamic I/O (I/O peer-to-peer)
- 3 dynamic explicit (explicit peer-to-peer)
- 1 master /slave explicit
- 1 polled
- 1 COS/cyclic
- 1-bit strobe

Polled

Typically polled I/O is used with DeviceNet™ networks.

The poll command is an I/O message that is transmitted by the master. A poll command is directed towards a specific node. The master must transmit a separate poll command for each node that is included in its scan list.

The poll response is an I/O message generated by the slave and sent to the master after the poll command is received.

Change of State (COS) or Cyclic

A change of state or cyclic message is used to receive produced data from a slave device. This message is triggered by either a change of state or by the expiration of the cyclic time period. The response size is equal to the number of digital, analog and special function modules input bytes, plus two bytes for the Inline Status word by default.

By default, the change of state response is triggered by either digital inputs or the Inline Status word. If the trigger determination for the COS message needs to be modified the Build COS Mask Mode attribute must be given the proper value to set the desired response trigger event(s). After the Build COS Mask Mode is set, a Build COS Mask must occur for the changes to go into effect. These attributes can be set by either using the EDS file or by sending explicit messages to the COS Mask Object (Class Code 104_{dec}, 68_{hex}).

If analog inputs are being used to trigger the COS response, the user has the ability to mask the lower bit weights to eliminate heavy network traffic due to fluctuating LSBs that may trigger the COS response. The Analog Mask attribute can be found in the EDS file or can be set by sending an explicit message to the COS Mask object (Class Code 104_{dec}, 68_{hex}). The default mask for analog inputs, RTD and thermocouples is one byte.

Specific bits can be set to be the trigger by using the COS Mask Index and COS Mask Byte Value attributes. An example is shown in Section D, "Editing the COS Mask".

Bit Strobe

The bit strobe is for inputs only. The bit-strobe command, transmitted by the master, has multicast capabilities. Multiple nodes can receive and respond to the same bit-strobe command.

When using the bit-strobe command, the value of this 1-bit command sent to the bus coupler is ignored. After which, the response is generated by the node and is sent to the master.

DeviceNet™ will use eight bytes of consumed data for all devices that use the bit-strobe connection.

The produced size of the response is limited to eight bytes. By default, this is equal to the number of digital, analog and special function module input bytes, plus two bytes for the Inline Status word.

I/O Peer-to-Peer

I/O peer-to-peer allows nodes to communicate in an implicit manner, without the need for a master or as an additional connection besides the master to slave connection. The bus coupler can offer this type of connection to another I/O peer-to-peer device that is able to establish the connection. An example of this relationship would be a motion controller using the I/O peer-to-peer connection to read inputs connected on an Inline station for control decisions.

Explicit Peer-to-Peer

Explicit peer-to-peer allows nodes to communicate in a explicit manner, without the need for a master or as an additional connection besides the master to slave connection. The bus coupler can offer this type of connection to another explicit peer-to-peer device that is able to establish the connection. An example of this relationship would be a configuration tool sending a message to the bus coupler.

3.6.2 I/O Communications Objects

Bus coupler I/O communication objects can be accessed through the use of the explicit messages. Listed below are the objects used to transfer I/O data for the bus coupler. If data needs to be transferred to a device that is not in a scan list, a “Get” or “Set” explicit message service can be sent to the proper class, instance and attribute in question.

3.6.2.1 Digital Input Point (DIP) Object (Class Code 08_{dec}, 08_{hex})

The DIP object models digital inputs in the bus coupler. There is a separate instance for each digital input point available on the device. Attributes include Value and Status.

Setting DIP Inputs to Latch

Each DIP point can be independently configured to latch on a desired state. This is accomplished by using attributes 100 and 101. Figure 3-10 shows how "actual input" or "latched" data is selected.

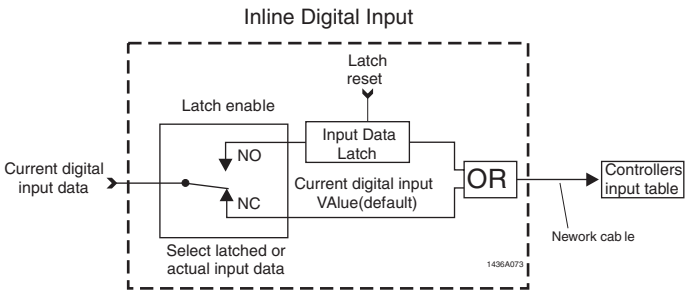


Figure 3-10 Latched or current data selection

Attribute 100 is used it to enable the latching feature for any specific DIP. When set to a logic 0 (default), the latching feature is OFF. When set to logic 1, the latching feature is ON (enabled).

Attribute 101 determines the latch level of a specific DIP. Setting attribute 101 to a logic 0 enables the DIP to select a low-level latch. Setting attribute 101 to a logic 1 enables the DIP to select a high-level latch.

Enabling the latch and setting the desired latch state must be done by sending an explicit message.

Resetting the latched condition must be done by setting bit 1 in the Inline Control byte. This clear will effect all latched inputs. After the latches are reset, bit 1 in the Inline Control byte must be set back to 0 to allow for the next latched condition to occur when the control byte is in the consumed data.

By default the Inline Control byte is not included in the consumed data command. The user can add this to the consumed data by issuing an explicit message to the Configuration Object (Class Code 100_{dec}, 64_{hex}, Attribute 32).

If the user does not want to clear the latches through the polled I/O then an explicit message to the Inline Interface Object (Class Code 101_{dec}, 65_{hex}, Attribute 20) can be sent. Setting attribute 20 to a 2 will reset all latches enabling the next input latch. It will also automatically reset the attribute 20 value to 0.

Latch values are retained during operation and will not be cleared until the latches are reset. Once a reset is received the latches will re-initialize to the value that allows the input level to be captured. This initialization depends on the value determined by attribute 101, Latch Level.

3.6.2.2 Digital Output Point (DOP) Object (Class 09_{hex})

The DOP object models digital outputs in the DeviceNet™ bus coupler. There is a separate instance for each digital output point available on the device. However, the value of the Status is the same for all the given points on a particular I/O module. Other attributes include: Value, Status, Fault State, Fault Value, Idle State and Idle Value.

3.6.2.3 Analog Input Point (AIP) Object (Class $0A_{hex}$)

The AIP object models analog inputs in the DeviceNet™ bus coupler. There is a separate instance for each analog input point available on the device. Attributes include: Value, Status and Range.

3.6.2.4 Analog Output Point (AOP) Object (Class $0B_{hex}$)

The AOP object models analog outputs in the DeviceNet™ bus coupler. There is a separate instance for each analog output point available on the device. Attributes include: Value, Output Range, Value Data Type, Fault State, Idle State, Fault Value and Idle Value.

3.6.2.5 Accessing Analog and Digital Instances 1 through 510

The bus coupler automatically supports the following number of instances for the specific object types when accessed using produced/consumed data that is mapped to a scanner.

Digital Inputs	510 instances
Digital Outputs	510 instances
Analog Inputs	128 instances
Analog Outputs	128 instances

However, if explicit messages are being used to retrieve I/O data the user will need to access the Configuration Object (Class Code 100_{dec}, 64_{hex}), attribute 33 to determine if instances 1 through 255 or 256 through 510 can be accessed. (There is no bank select for special function data.)

By default, Attribute 33_{dec} (I/O Bank Select) is set to a 0, which allows the user to explicitly access instances 1 through 255. If the user needs to access instances 256 through 510 explicitly, the I/O Bank Select (Attribute 33) must be set explicitly in the Configuration Object (Class Code 100_{dec}, 64_{hex}).

Example: To access the 256th digital input point, using an explicit message, proceed as follows:

1. The user must set the I/O Bank Select to a 1.
2. A Get (Get_Attribute_Single) to Instance 1 of the DIP object accesses the value for Instance 256.

Setting the bank select to a 1 adds 255 instances to the instance called out in the explicit message.

3.6.2.6 Inline Special Function Object (Class 43_{hex})

The Inline special function object gives the user the ability to control and monitor the below listed modules and any other module that does not map to a standard DeviceNet™ object.

– Incremental encoder
– Absolute encoder
– High Speed counter

3.6.2.7 PCP Special Function Object

By default, the PCP Special Function Object contains one instance for each PCP module. If the bus coupler detects that the modules is designed for serial communications, it will also create an instance in the Serial communications Object. An example of a PCP module is:

– AS-i master

3.6.2.8 Serial Communications Object

The Serial Communications Object contains one instance for each PCP module that is designed for serial communications. It is possible to access an instance of the Serial Communication Object into an instance in the PCP Special Function Object. Examples of Serial Communications PCP modules are:

– RS-232
– RS-485

3.7 Fault Response Modes

Fault response is a configurable setting that will control the behavior of the Inline station in the event of one of the 3 below listed errors occur:

Module Change Error Bit 3 in the Inline Status word

Configuring Error Bit 4 in the Inline Status word

Module Connection Error Bit 5 in the Inline Status word

The bus coupler has the ability to run in any of the fault response modes described below.

Fault Response Mode 0: Stop on Fault (Default) Outputs are turned OFF automatically when a local failure occurs. Once the failure(s) is fixed, either a reconfiguration (Class 64_{hex}, Instance 1, Attribute 6) or a power cycle (U_L and DeviceNet™ power) needs to occur before the station becomes operational again.

Fault Response Mode 1: Auto Restart Outputs are turned OFF automatically when a local failure occurs. Once failure is fixed, the station will begin to update I/O automatically.

Fault Response Mode 2: Goto Fault State Outputs are turned OFF automatically for up to two seconds when a local failure occurs. After two seconds the station will attempt to set the outputs to the pre-programed DeviceNet™ fault states. The station will not recover its lost I/O, even after the fault condition has been cleared, without a reconfiguration or power cycle (U_L, DeviceNet™). The default for a preprogrammed fault state is OFF.

Fault Response Mode 3: Continue on Fault Outputs are turned OFF automatically for up to two seconds when a local failure occurs. After two seconds the station will continue to update all I/O that is still on line. The station will not recover its lost I/O, even after the fault condition is cleared, without a reconfiguration or power cycle (U_L, DeviceNet™).

Changing the Fault Response Mode

Fault response mode is a flash memory setting in the bus coupler. A fault response mode cannot be changed by reconfigurating the I/O station or by cycling power (U_L, DeviceNet™).

Changing the fault response from the default (mode 0) to a different mode can be accomplished in two ways:

1. an explicit message is sent to the Configuration Object (described in Section B, "DeviceNet™ Object Classes, Message Types and Services");
2. EDS parameter 26 (Fault Mode) is set to desired fault response mode number (described above) then downloaded to the bus coupler.

4 Diagnostics

4.1 Diagnostic and Status Indicators

All modules are provided with diagnostic and status indicators for rapid local error detection.

Diagnostics Indicators

The diagnostic indicators (red or green) show the state of the Inline modules. When a module is operating normally, all its diagnostic LEDs are green.

After an error is detected, the indicators immediately display the current status.

Status Indicators

The status indicators (yellow) display the status of the relevant inputs/outputs or of the connected device.

Each different type of module has different diagnostic and status indicators.

Refer to the module-specific data sheet to see which diagnostic and status LED indicators apply to that module.

4.2 Inline DeviceNet™ Bus Coupler LEDs

Figure 4-1 and Table 4-1 provides the different LED states that can be read from the bus coupler.

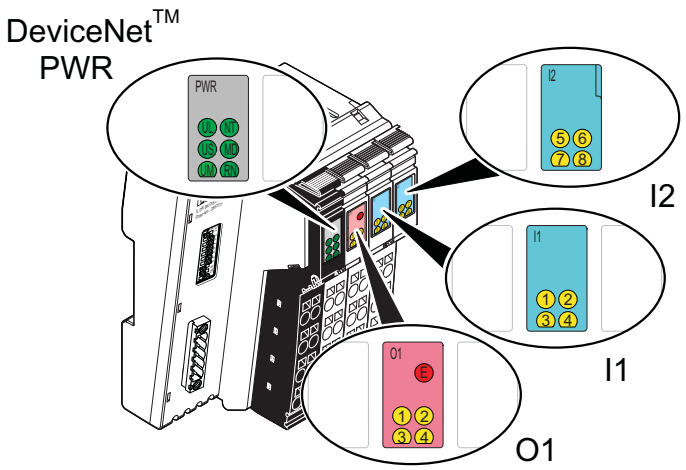


Figure 4-1 Diagnostic and status indicators on the bus coupler

Table 4-1 Indicators on the bus coupler

LED	Color	Meaning	State	Description of the LED States
PWR, DeviceNet™				
UL	Green	$\mathbf{U}_{Logic}$	ON	24 V bus coupler supply 7.5 logic voltage is present ($\mathbf{U}_{Logic}$)
			OFF	24 V bus coupler supply/ 7.5 logic voltage is not present
US	Green	$\mathbf{U}_{Segment}$	ON	24 V supply of the segment circuit ($\mathbf{U}_{Segment}$) is present
			OFF	24 V supply of the segment circuit is not present
UM	Green	$\mathbf{U}_{Main}$	ON	24 V supply of the main circuit ($\mathbf{U}_{Main}$) is present
			OFF	24 V supply of the main circuit is not present
NT	Green/ red	**Net**work status		Network status
			Green ON	Device is online and has established a connection Group 2 devices are assigned to a master
			Red ON	Error preventing communication with the network (e.g., bus is offline or duplicate MAC ID).
			Green flashing	Device online, connections not established. Device has finished the "duplicate MAC ID" test but has not established connections to other nodes. Group 2 devices are **not** assigned to a master.
			Red flashing	One or more connections are in the timeout state.
			Red-green flashing	Network access error
			OFF	Device not online The device has not yet finished the "duplicate MAC ID" test. No DeviceNet™ power is present.
MD	Green/ red			Device status (**M**odule **D**iagnostics)
			Green ON	Normal operation
			Red ON	Unrecoverable error
			Green flashing	– Device not configured, or device configuration not complete or faulty. – Device in standby mode
			Red flashing	Recoverable error
			Red-green flashing	Selftest
			OFF	No DeviceNet™ power is present.

Table 4-1 Indicators on the bus coupler (continued)

LED	Color	Meaning	State	Description of the LED States
RN	Green	**R**un		Status of the local Inline communication
			Green ON	Local bus communication present
			Green flashing 10 Hz	Initialization phase
			1 x Green flashing	"Module Change" alarm during startup
			2 x Green flashing	Connection error
			Green flashing 20 Hz	Loop diagnostics (e.g., for connected FLM or AS-i modules)
			3 x Green flashing	"Module Change" alarm during operation
			Green flashing 1 Hz	Configuration error (without other error)
			4 x Green flashing	Operation with error data
			Green flashing 2 Hz	Peripheral fault (without other errors)
			5 x Green flashing	Error of supply voltage
			OFF	The bus coupler only operates with integrated inputs and outputs. No other Inline terminals are snapped on.
O1				
E	Red	**E**rror	ON	Peripheral fault present, short-circuit or overload at one of the outputs
			OFF	No short circuit or overload at one of the outputs
1 - 4	Yellow	Output **1** to Output **4**		Status indicators of the outputs
			ON	Output active
			OFF	Output not active
I1, I2				
E	Red	**E**rror	ON	Peripheral fault present, short circuit or overload of the sensor supply
			OFF	No short circuit or overload of the sensor supply
1 - 8	Yellow	Input **1** to Input **8**		Status indicators of the inputs
			ON	Input active
			OFF	Input not active

4.3 Available Network Diagnostics

A detailed explanation of object classes can be found in Section B, "DeviceNet™ Object Classes, Message Types and Services".

Diagnostic information is made available through several mechanisms. The EDS file allows the user to read the Inline Status word and condition of the standard DIPs, DOPs, AIPs, AOPs and special function modules. This Inline Status word and I/O point/channel status can also be mapped directly to the polled I/O. The final method of retrieval is to explicitly query the attributes from the Configuration Object (Class Code 64_{hex}) and Inline Object (Class Code 65_{hex}).

4.3.1 Inline Status Word

By default, Inline Status word data is made available to the user as two bytes of diagnostic data in the polled scan. These two bytes contain the Inline fault code (byte 0) and the number of the first module in the local bus that is faulted (byte 1). The status word adds 2 bytes to the produced data size by default. The status word updates status automatically so when an error is cleared the status will be set back to a 0.

If the user needs to remove this data from the poll, the status word can be disabled by sending a 0 in the "Use Inline Status" parameter during an "Add All I/O" using either the EDS file or an explicit message to the Configuration Object (Class Code 64_{hex}).

4.3.2 Major/Minor Faults

By default all Inline Status word fault bits (byte 0, bits 0, 2-6) except for bit 1 are considered major recoverable faults, as defined in the Identity Object state and status attributes, and will flash the red MD LED on the bus coupler when that specific type of failure occurs. Bit 1, peripheral fault, is assigned as a minor recoverable (default value) fault and will not flash the MD LED when in a faulted condition. These default values can be changed by using the EDS file or by sending an explicit message to the Configuration Object (Class Code 64_{hex}). Possible fault action setting are:

0 None

1 Minor Recoverable Fault

2 Major Recoverable Fault

4.3.3 Bit Meanings for Inline Status Word (Byte 0)

Bit 0 CRC Error

The CRC error bit will be set when a data transmission error occurs due to unwanted interference on the Inline local bus.

Bit 1 Peripheral Fault

The Peripheral Fault bit will be set when any output is shorted or a loss of power to an intelligent segment module.

Bit 2 Power Fault

The Power Fault bit will be set when any of the power supplies (U_L, U_S, U_M, DeviceNet™) are in an under voltage condition (less than 11 V DC).

Bit 3 Module Change

The Module Change bit will be set when the configuration present on the Inline local bus does not match the configuration that was stored in flash during the last configuration cycle.

Bit 4 Configuring Error

The Configuring Error bit will be set when the bus coupler is not able to talk to the first I/O module connected to it. Possible failures include the bus coupler itself or the first I/O module connected to it. Power down and reconnect the I/O to the bus coupler.

Bit 5 Module Connection Error

The Module Connection Error bit will be set when the bus coupler is no longer able to talk to the modules connected to it and can determine the failure position. This failure occurs due to a broken data path. The exact path "between what two modules" can be read from the Inline Interface Object (Class Code 65_{hex}).

Bit 6 Outputs Set to Preprogram DeviceNet™ Fault State

This bit can only be set in the Fault Response mode 2 (described in Section 3.4, "Configuring the Inline Station"). It is made available to let the application know that the local outputs have gone to their preprogrammed DeviceNet™ fault state and will no longer respond to the controller.

Bit 7

Reserved for future use

4.3.4 Bit Meanings for Inline Status Word (Byte 1)

Contains the first failed device number. The device number determines the position on the Inline station where a failure or warning has occurred. These positions are numbered starting at the bus coupler being assigned with a 1. The numbering will continue to the right up to 64, which is the maximum number of devices that can be connected to an Inline station (63 I/O devices + 1 bus coupler).

Inline local errors will not be sent over the network unless the Inline Status word is in the poll or an explicit message to the Inline Object is sent periodically.

These errors by default are considered a major (except for a peripheral fault) error and the MD LED on the bus coupler will blink red. A determination must be made regarding the Inline Status word and its desired effect on the network and/or failing node through the users application.

4.3.5 Latched Diagnostics

The bus coupler will latch the last occurring Inline Status word fault, module number and connection point 1 and 2 failures. This benefits the user by capturing any fault that may be occurring intermittently and that is occurring too quickly to be updated by the DeviceNet™ poll or by an explicit message. These latched values can not be cleared until the station is reconfigured. The following latched diagnostics are available through the EDS file or by sending an explicit message to the Inline Interface Object (Class Code 65_{hex}).

Latched Inline Status Word

This parameter will contain the last reported Inline station failure. The bit weights signify the same failures as described in the Inline Status word (byte 0).

Latched Faulted Module

This parameter will contain the first failed module location that was reported during the last Inline station fault. The bit weights signify the same failures as described in the Inline Status word (byte 1).

Latched Connection Failure Endpoint 1

This parameter will contain the number of the module that was reported on the first end of a connection failure.

Latched Connection Failure Endpoint 2

This parameter will contain the number of the module that was reported on the other end of a connection failure.

4.3.6 Inline Control Byte

The Inline Control Byte is used to acknowledge latched peripheral faults (bit 0) or to clear latched inputs states (bit 1).

For an explanation of latching input states, refer to Section "I/O Data Transfer" on page 3-24.

By default, the Inline Control Byte is not added to the poll. It can be added by setting Instance 1, Attribute 32 of the Configuration Object (Class Code 64_{hex}) to a 1. If the user would rather access this byte through an explicit message, a Get or Set can be sent to the Inline Interface Object (Class Code 65_{hex}), Instance 1, Attribute 20.

– **Bit 0:** When set to a 1, will attempt to clear all latched peripheral faults.
– **Bit 1:** When set to a 1, will clear all latched input states.

The latched peripheral fault can only be generated by certain Inline modules. Examples of this type of module are the IB IL SEG-ELF and the IB IL 24 EDI 2-DESINA.

4.3.7 I/O Point/Channel Status

The fault status of a digital, analog, or special function point is either 0 (functioning) or 1 (failed). The fault status can be added to the poll through the EDS file or solicited by issuing an explicit message to the Configuration Object (Class Code 64_{hex}), see Section B, "DeviceNet™ Object Classes, Message Types and Services". When adding this status to the poll, a bit for each point or channel will be assigned to the input image. This will occur before the mapping of the actual point or channel. The mapping assignments of these status bits will occur in order of their instance on the local bus.

I/O status bits can be added to the poll through the use of the EDS file. Parameters 16 through 20 (described in the following paragraphs) allow for respective status bits to be added to the poll.

Parameter 16 Selects the number of DIP faults added to the poll response on a point basis.

Parameter 17 Selects the number of DOP faults added to the poll response on a point basis.

Parameter 18 Selects the number of AIP faults added to the poll response on a channel basis.

Parameter 19 Selects the number of AOP faults added to the poll response on a channel basis.

Parameter 20 Selects the number of special function faults added to the poll response on a channel basis.

4.3.8 Fault/Idle State and Value

The bus coupler supports the standard DeviceNet™ DOP (Digital Output Points), AOP (Analog Output Points), fault or idle states, and values. These values can be set and read by the use of an explicit message. Fault states will only occur during a network error. They will not occur after an Inline local error. The default value for the AOPs and DOPs is zero. Idle states and values will occur when a PLC is taken out of the run state. The default value for the idle state is also zero.

Digital Output Support:

- Holds last state
- Turn off during a faulted condition (default)
- Turn on during a faulted condition

Analog Output Support:

- Hold last value
- Set to low limit
- Set to high limit
- Set to value determined by the fault value attribute

Section B, "DeviceNet™ Object Classes, Message Types and Services" will detail the DOP (Class Code 09_{hex}) and AOP (Class Code $0B_{hex}$) fault/idle values and states.

4.3.9 Analog Input, Thermocouple and RTD Fault Codes

Inline analog inputs, thermocouples and RTDs can report diagnostic codes. These codes must be read from the produced response or the AIP detailed in Section B, "DeviceNet™ Object Classes, Message Types and Services". A list of these codes, in Inline format "IL", is shown in Table 4-2.

Error codes are dependent on the type of module and how that module's format is configured. By default error codes are received in the Inline "IL" format and can be viewed as shown in Table 4-2. If the format has been changed, the user must refer to the module specific data sheet to determined what error code has been received.

Table 4-2 Error messages of analog input modules

Code (hex)	Error Message
8001	Under-range
8002	Open circuit
8004	Measured value invalid
8008	Cold junction defective
8020	I/O supply voltage faulty
8010	Configuration invalid
8040	Module defective
8080	Over-range

4.3.10 Error History

Error history provides access to the last ten errors that have been stored in the bus coupler. These errors can be accessed using either the EDS file (Parameter 56 (most recent) through Parameter 65 (oldest) or by using the Inline Interface Object Class 101, Instance 1, Attribute 21 (most recent) to attribute 30 (oldest). As error values are added, existing values will be shifted to older parameters. The new value is then placed in the "Most Recent" parameter and the value in the "Last Saved" parameter is discarded.

The error history entry will contain the faulted module number in the high byte and the Inline status code in the low byte.

A "0" for an error history entry represents a point where an error was removed.

An error history value may be recorded at a point where an error is detected but is not yet localized. When the error is localized, a new error history value will be added.

5 Technical Data and Ordering Data

Data presented in the following tables is based on the product being mounted in the preferred vertical mounting position. Please refer to the DeviceNet™ specification and to the module-specific data sheets for additional information or to confirm the accuracy of technical data presented.

5.1 Technical Data

For technical data of the Inline system, please refer to the IL SYS INST UM E user manual. For technical data of other product ranges, please refer to the corresponding user documentation.

General Data	
Housing dimensions (width x height x depth)	80 mm x 121 mm x 70 mm
Weight	255 g (with connectors)
Ambient temperatures (operation)	-25°C to +60°C
Ambient temperature (storage/transport)	-25°C to +85°C
Permissible humidity (operation/storage/transport)	10% to 95%, according to EN 61131-2
Permissible air pressure (operation/storage/transport)	70 kPa to 106 kPa (up to 3000 m above sea level)
Degree of protection	IP20 according to IEC 60529
Class of protection	Class 3 in acc. with VDE 0106, IEC 60536
Connection data for Inline connectors	
Connection type	Spring-cage terminals
Conductor cross-section	0.2 mm^2 to 1.5 mm^2 (solid or stranded), 24 - 16 AWG

DeviceNet™ System Data	
Transmission speed	500 kbps, 250 kbps, 125 kbps
Transmission reliability	CR check
Maximum transmission distance	Up to 500 m (depends on transmission speed)
Protocols	Control and information (**C**ommon **I**ndustrial **P**rotocol) with CAN as transport medium
Nodes	64, max.
Voltage range of the bus coupler (+V, -V)	11 V - 25 V
Nominal current of the bus coupler (+V, -V)	60 mA
Bus coupler certification	ODVA

System Limits of Bus Coupler	
Transmission rate on the local bus	500 kbps or 2 Mbps (automatic detection)
Number of I/O devices within an Inline system	63, max.
Maximum number of process data	Limited by Class-Instance maximums Observe the 2 byte status data of the bus coupler.
Maximum number of PCP devices	8
Maximum power supply at U_L (7.5 V)	0.8 A, max.
Maximum power supply at U_{ANA}	0.5 A, max.
Maximum power supply at U_S	8 A, max. (total current with U_M)

System Limits of Bus Coupler (Continued)

Maximum power supply at U_M	8 A, max. (total current with U_S)
Maximum current consumption of the I/O terminals	See terminal-specific data sheet

NOTE:
Observe the current consumption of each device when configuring an Inline station. The current consumption is indicated in each terminal-specific data sheet. The current consumption can differ depending on the individual terminal. If the maximum current capacity of a power supply is reached, a new power terminal must be used.

DeviceNet™ Interface

CAN bus, connected via TWIN COMBICON connector

Common Data for 24 V Supply U_L, U_S, U_M

Connection	Via power connector; for terminal point assignment see Section "Connection of Supply, Actuators, Sensors" on page 2-6
Connection method	Spring-cage terminals
Recommended cable lengths	30 m, max.; routing cables through outdoor areas is not admissible
Continued	Via potential routing
Nominal value	24 V DC
Tolerance	-15%/+20% (according to EN 61131-2)
Ripple	± 5%
Permissible range (according to EN 61131-2)	19.2 V to 30 V
Safety equipment	
Surge voltage, polarity reversal	Yes, suppressor diode parallel to supply voltage

NOTE: The bus coppler is damaged in the event of overload
The 24 V area must be protected externally. The power supply unit must be able to supply 4 times the nominal current of the external fuse, to ensure that it trips in the event of an error. Recommended fuse 4 A medium blow.

U_L (7.5 V) and U_{ANA} (24 V) are generated from U_L (24 V).

Current Consumption / Power Consumption

Current consumption from U_{LOGIC} (24 V)	
Current consumption local bus (800 mA load at 7.5 V)	0.4 A, max.
Current consumption from U_{ANALOG} (24 V)	0.5 A, max.
Current consumption from U_{LOGIC} total	0.9 A, max.
Current consumption from U_S (24 V)	8 A, max.
Current consumption from U_M (24 V)	8 A, max.
Power dissipation entire device	3.5 W, max.

Digital Outputs

Number	4
Connection method for actuators	2 and 3-wire technology
Nominal output voltage U_{OUT}	24 V DC
Differential voltage for I_{nom}	< 1 V
Nominal current I_{nom} per channel	0.5 A
Total current	2 A
Nominal load	
Ohmic	12 W
Lamp	12 W
Inductive	12 VA (1.2 H)
Switching frequency with nominal inductive load	0.5 Hz (1.2 H), maximum
Response time	1.2 ms, typical

Digital Outputs (Continued)	
Overload response	Auto restart
Response with inductive overload	Output may be damaged
Reverse voltage protection against short pulses	Protected against reverse voltages
Resistance to permanently applied reverse voltages	Protected against reverse voltages, permissible current 2A, maximum
Response upon power down	The output follows the supply voltage without delay
Limitation of the voltage induced on circuit interruption	-30.0 V, approximately
Safety equipment	Short circuit; overload
Type of safety equipment	Integrated free running circuit in the output chip
Maximum output current when switched off	10 µA
Error message to the higher-level control system	Short circuit; overload

When not loaded, a voltage can be measured even at an output that is not set.

Digital Inputs	
Number	8
Connection method for sensors	2 and 3-wire technology
Input design	According to EN 61131-2, type 1
Switching threshold definition	
Maximum low-level voltage	$U_{Lmax} < 5$ V
Minimum high-level voltage	$U_{Hmin} > 15$ V
Common potentials	Sensor supply U_S, ground
Nominal input voltage U_{IN}	24 V DC
Permissible range	-30 V $< U_{IN} < +30$ V DC
Nominal input current for U_{IN}	3 mA, typical
Current flow	Limited to a 3 mA, maximum
Delay time	1.2 ms, typical
Permissible cable length to the sensor	100 m
Use of AC sensors	AC sensors in the voltage range $< U_{IN}$ are limited in application
Safety equipment	Polarity reversal
Type of safety equipment	Serial diode for polarity reversal protection
Error message to the higher-level control system	Sensor supply not present

Mechanical Requirements	
Vibration test sinusoidal vibrations according to IEC 60068-2-6; EN 60068-2-6	5g load, 2 hours in each space direction
Shock test according to IEC 60068-2-27; EN 60068-2-27	25 g load for 11 ms, half sinusoidal wave, 3 shocks in each space direction and orientation

Conformance With EMC Directive 89/336/EEC or 2004/108/EG

Noise Immunity Test According to EN 61000-6-2

Electrostatic discharge (ESD)	EN 61000-4-2/ IEC 61000-4-2	Criterion B 6 kV contact discharge 8 kV air discharge
Electromagnetic fields	EN 61000-4-3 IEC 61000-4-3	Criterion A Field strength: 10 V/m
Fast transients (burst)	EN 61000-4-4/ IEC 61000-4-4	Criterion A All interfaces: 1 kV Criterion B All interfaces: 2 kV
Surge voltage	EN 61000-4-5/ IEC 61000-4-5	Criterion B Supply cables DC: 0.5 kV / 1 kV (symmetrical/asymmetrical) Fieldbus cable shield 1 kV
Conducted interference	EN 61000-4-6 IEC 61000-4-6	Criterion A Test voltage 10 V

Noise Emission Test According to EN 61000-6-4

Noise emission of housing	EN 55011	Class A

Approvals

Information on latest approvals can be found on the Internet at www.download.phoenixcontact.com at www.eshop.phoenixcontact.com.

5.2 Ordering Data

5.3 Bus Coupler

Description	Type	Order No.	Pcs./Pck.
Inline bus coupler for DeviceNet™ with 8 digital inputs and 4 digital outputs including accessories (end plate, connector and labeling fields)	IL DN BK DI8 DO4-PAC	2897211	1

5.4 Accessories

Description	Type	Order No.	Pcs./Pck.
DeviceNet™ connector	TMSTBP 2,5/ 5-STF-5,08 GNAU CP	2862576	1
Connector set as replacement item	IL BK DIO-PLSET	2878599	1
Extended double signal connector for inputs, 4 signals with 3-wire connection method	IB IL SCN-12-ICP	2727611	10
Extended double signal connector for outputs, 4 signals with 3-wire connection method	IB IL SCN-12-OCP	2727624	10
Keying profile	IL CP	2742683	100
Zack marker strip to label the terminals	ZB 6 ... see CLIPLINE catalog	1051003	10
Zack marker strip to label the terminals	ZB 12 ... see CLIPLINE catalog	0812120	10
DIN EN 50022 DIN rail, 2 meters	NS 35/7,5 PERFORATED NS 35/7,5 UNPERFORATED	0801733 0801681	1 1
End clamp	CLIPFIX 35-5	3022276	50

5.5 Additional System Components

Description
Power supply units (INTERFACE catalog) for supplying the bus coupler

5.6 Documentation

Description	Type	Order No.	Pcs./Pck.
User Manual: Automation Terminals of the Inline Product Range	IL SYS INST UM E	2698737	1
Application Note: I/O Modules at Bus Couplers	AH IL BK IO LIST	9015358	1

A Serial and Other PCP Inline Modules

A 1 General

This section provides the following information on the use of the Inline serial modules and any other module that uses the Peripheral Communications Protocol (PCP):

- Communication methods for the Inline serial modules (RS-232 or RS-485/RS-422)
- Communication methods for any other module that supports the PCP protocol.
- Serial and Generic PCP Modules Produced and Consumed Sizes
- I/O memory mapping
- RS-232 and RS-485/RS-422 configuration brief

A 2 Communications Methods

Communications to an Inline serial module can be accomplished in two ways. Likewise, communications to a generic PCP module can be accomplished in two ways. The type of module being used, either serial or generic PCP, determines what types of communication methods are available to the user (by default).

The first method available to the user is the sending or receiving of data using process/cyclic data channel (fragmentation) and the second is by sending explicit messages. Choosing between the two methods is a matter of the capabilities of the DeviceNet™ scanner and/or personal preference.

When using the Inline RS-232 or RS-485/RS-422 modules, simplified methods 1 or 2 can be used. If using any module (including serial) that supports PCP, methods 3 or 4 can be used.

Serial modules are not supported under mapping revision 0.

- Method 1: Transfer of Serial Data Using Serial Fragmentation
- Method 2: Transfer of Serial Data Using Explicit Messages
- Method 3: Transfer of Generic PCP Data Using PCP Fragmentation
- Method 4: Transfer of Generic PCP Data Using Explicit Messages

Supported serial modules are:

- IB IL RS 485/422 (Inline RS-485/422 module)
- IB IL RS 232 (Inline RS-232 module)

An example of a “Generic” PCP Inline module is the:

- ASI MA IB IL (Inline AS-i Gateway)

1. PCP is not required when using the ASI MA IB IL, if the AS-i branch has less than 32 slaves.
2. If the PCP module looses power, a reset service, with a data value of "1", can be issued to the Inline Interface object (101_{dec}, 65_{hex}) or the BK can be power cycled.
3. You can access a serial module as a PCP Special Function module. To do this, first disable the module's instance in the Serial Communication Object. Then, remove the module's serial process and fragment data from the poll using the Serial Communication Object. Finally, add the PCP special function process and fragment data into the poll using the PCP Special Function Object.

If the autoconfiguration switch is ON, at next power-up the BK will erase any custom settings described in the note 3 above.

The bus coupler can hold up to 64 bytes of incoming and 64 bytes of outgoing data before data will overflow internal buffers.

A 2.1 Method 1: Transfer of Data Using Serial Fragmentation

This method defines how serial data is exchanged using process/cyclic data. The messages that are required to read and write data are encoded into the high-speed data stream. The protocol is handled using a series of message fragments that are initiated by a client request and then followed up with a server response.

Each fragment contains 8 bytes. Every fragment includes a control byte for a request (output data) and a status byte for a response (input data).

These fragments were specifically designed for the serial Inline modules and will not work for other PCP based Inline modules.

Depending on the amount of data to be sent, the number of fragments required to read/write data can vary. Fragments are eight bytes in length. Each fragment contains a request format and a response format. These formats are detailed in the following paragraphs.

A 2.1.1 Format of Fragmented Serial Output Data

Consumed Request

Byte 0	Control Byte (See)
Bytes 1 to 7	Data block, if necessary

Table A-1 Control Byte

7	6	5	4	3	2	1	0
Res.	Res.	Receive Ack.	Transmit Request	Reset Request	Number of Data Bytes to Transmit (Per Fragment)		

Bit 7	Reserved
Bit 6	Reserved
Bit 5	Receive Ack (RxAck)
	This bit must be set to the value of the Receive Request bit (RxReq) in order for the module to receive more data. This tells the module that the “master” has received the data and has processed it.
Bit 4	Transmit Request
	This bit must be toggled to signal the module that there is new data to transmit. On devices that cannot ensure data consistency, the user should first set the number of bytes and place the proper data into the TxData mapping before toggling this bit.
Bit 3	Reset Request
	When this bit is set, all errors are cleared, the buffers are flushed, and the RxReq and TxAck bits are cleared. This allows re-sync of the protocol.
Bit 2 to 0	Number of Bytes to Transmit
	This tells the module how many bytes of the data are valid and should be transmitted.

A 2.1.2 Format of Fragmented Serial Input Data

Produced Response:

Byte 0 Status Byte (See

Byte 1 to 7 Data block, if necessary

Table A-2 Status Byte

7	6	5	4	3	2	1	0
Error	Res.	Receive Request	Transmit Ack.	Reset Ack.	Number of Data Bytes Received (Per Fragment)		

Bit 7 Error

When this value is set, an error has occurred such as parity or overrun. The user should query the status parameter for more information.

Bit 6 Reserved

Bit 5 Receive Request

This value is toggled to indicate new data has been received. The user must acknowledge the reception of data by echoing back this value in the Receive Ack bit.

Bit 4 Transmit Ack.

When this value is equal to the Transmit Request, it indicates that the output data has been queued into the output buffer. Once they are equal the user can then send more data.

Bit 3 Reset Ack.

If a Reset has been requested, this value will be set to 1 to indicate that the serial port has been reset and the buffers have been flushed. This causes the TxAck and RxReq to be reset to 0 allowing re-sync of protocol.

Bits 2 to 0 Number of Bytes Received

Indicates the number of valid data bytes that are in the data section of the input data.

A 2.1.3 Serial Fragmentation Examples

The following examples will show how to read, write and handle errors using serial process data fragmentation.

– Fragmented Read
– Fragmented Write
– Error Handling, Communications Backplane Break
– Error Handling, Host Communications Loss

Fragmented Read Example

Figure A-1shows an example I/O data table that contains only eight bytes of input data and eight bytes of output data. This I/O is shown in an event by event sequence that demonstrates how these 8 bytes of I/O are updated when reading data using Serial PCP fragmentation. The sequence works on this following basic principle:

Event x. Client issues server 8 bytes of output data

Event x+1. Server responds by updating 8 bytes of input data

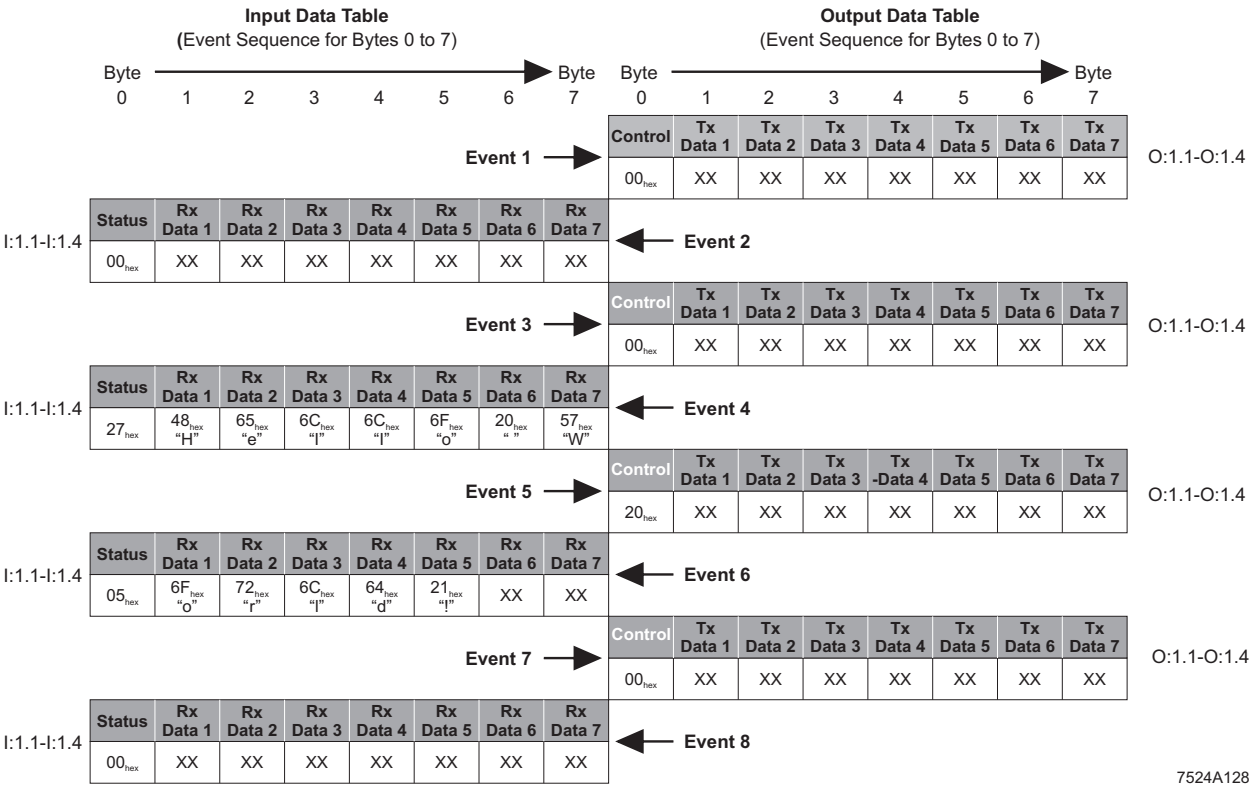


Figure A-1 Serial fragmented read example

The following paragraphs explain those events listed in Figure A-1.

Table A-3 to Table A-9 demonstrate the order of events when issuing a fragmented read service to a PCP device. Each “Event” should be referenced to the I/O data table shown in Figure A-1.

Event 1

Table A-3 shows the transmission from a master to slave when the serial fragmentation is in “Idle” mode.

Table A-3 Master to slave idle transmission

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 2

Table A-4 shows the slaves response to the idle state by replying with a 00_{hex}.

Table A-4 Slave to master idle response

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 3

Master still sending the idle message.

Event 4

Table A-5 shows that the slave has received 7 bytes of new data. This indication is explained as follows:

Table A-5 Slave to master indication that new data has been received

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
27_{hex}	48_{hex} „H“	65_{hex} „e“	$6C_{hex}$ „l“	$6C_{hex}$ „l“	$6F_{hex}$ „0“	20_{hex} „ “	57_{hex} „W“

Status, byte 0 27_{hex}

Bit 5 = 1 Receive Request being toggled Indicates that new data is present.

Bits 2 to 0 = 7 Shows the number of bytes received.

Event 5

Table A-6 shows the master to slave acknowledgment of a Receive Request indication.

Table A-6 Master to slave, receive data acknowledgement

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
20_{hex}	XX	XX	XX	XX	XX	XX	XX

Control, byte 0 20_{hex}

Bit 5 = 1 After the 7 bytes have been received and processed the Receive Ack. Bit is set to the same value as the Receive Request bit to signal the module that the master is ready to receive more data.

Event 6

Table A-7 shows that the slave has received 5 additional bytes of data:

Table A-7 Slave to master indication that more data is present

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
05_{hex}	$6F_{hex}$ „0“	72_{hex} „r“	$6C_{hex}$ „l“	64_{hex} „d“	21_{hex} „!“	XX	XX

Status, byte 0 05_{hex}

Bit 5 = 0 Receive Request being toggled
(In event 6 bit 5 = 0) Indicates that new data is present.

Bits 2 to 0 = 5 Shows the number of bytes received.

Event 7

Table A-8 shows the master to slave acknowledgment of a Receive Request indication.

Table A-8 Master to slave, receive data acknowledgement

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Control, byte 0 00_{hex}

Bit 5 = 0 After the 5 bytes have been received and processed the Receive Ack. Bit is set to the same value as the Receive Request bit to signal the module that the master is ready to receive more data.

Event 8

Table A-9 shows the slave to master indication that no more data is present.

Table A-9 Slave to master, no more data indication

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Fragmented Write Example

Figure A-2 shows an example I/O data table that contains only eight bytes of input data and eight bytes of output data. This I/O is shown in an event by event sequence that demonstrates how these eight bytes of I/O are updated when writing data using Serial PCP fragmentation. The sequence works on this following basic principle:

Event x. Client issues server 8 bytes of output data

Event x+1. Server responds by updating 8 bytes of input data

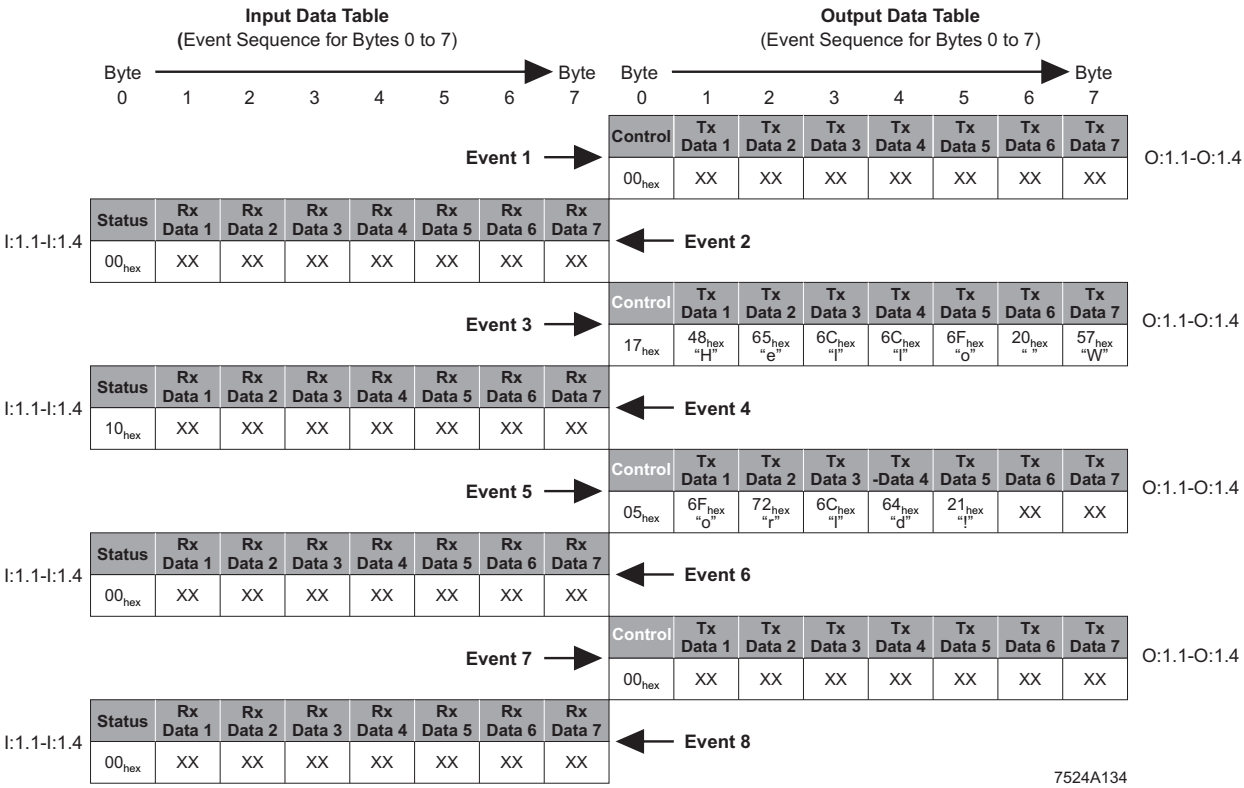


Figure A-2 Serial Fragmentation Write Example

The following paragraphs explain in detail those events listed in Figure A-2.

Table A-10 to Table A-15 demonstrate the order of events when issuing a fragmented write service to a PCP device. Each “Event” should be referenced to the I/O data table shown in Figure A-2.

Event 1

Table A-10 shows the transmission from a master to slave when the serial fragmentation is in “Idle” mode.

Table A-10 Master to slave idle transmission

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 2

Table A-11 shows the slave's response to the idle state by replying with a 00_{hex}.

Table A-11 Slave to master idle response

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 3

Table A-12 shows the master to slave transmission of 7 bytes of data.

Table A-12 Master to slave data transmission

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
17_{hex}	48_{hex} „H“	65_{hex} „e“	$6C_{hex}$ „l“	$6C_{hex}$ „l“	$6F_{hex}$ „0“	20_{hex} „ “	57_{hex} „W“

Control, byte 0 17_{hex}

Bit 4 = 1 Transmit request is toggled indicating that new data is being transmitted.

Bits 2 to 0 = 7 Shows the number of bytes to transmit.

Event 4

Table A-13 shows that the BK has received the data.

Table A-13 Slave to master acknowledgement of data transmission

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
10_{hex}	XX	XX	XX	XX	XX	XX	XX

Status, byte 0 10_{hex}

Bit 4 = 1 Transmit Acknowledge is being set to the same value as Transmit Request to indicate that the module is ready to receive more data.

Bits 2 to 0 = 0 0 bytes are being received at this time

Event 5

Table A-14 shows the master to slave acknowledgment of Transmit Request indication.

Table A-14 Master to slave indication for data transmission

Control	Tx Data 1	Tx Data 2	Tx Data 3	Tx Data 4	Tx Data 5	Tx Data 6	Tx Data 7
05_{hex}	$6F_{hex}$ „0“	72_{hex} „r“	$6C_{hex}$ „l“	64_{hex} „d“	21_{hex} „!“	XX	XX

Control, byte 0 05_{hex}

Bit 4 = 0 Transmit Request has been toggled indicating that there is more data to be transmitted.

Bits 2 to 0 = 5 Shows 5 bytes to transmit.

Event 6

Table A-15 shows that the slave has received 5 additional bytes of data. This indication is explained as follows:

Table A-15 Slave to master output data is "Queued" response

Status	Rx Data 1	Rx Data 2	Rx Data 3	Rx Data 4	Rx Data 5	Rx Data 6	Rx Data 7
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Status, byte 0 00_{hex}

Bit 4 = 0 Transmit Acknowledge is being set to the same value as Transmit Request to indicate that the module is ready to received more data.

Bits 2 to 0 = 5 No data is being received at this time

Event 7 and Event 8

Master to slave and slave to master idle mode. (No more data to send.)

Fragmented Error Handling, Loss of Inline Backplane Communications Example

This serial fragmentation error handling example, shown in Figure A-3, will show how the fragmentation will react during a break in the communications path on the Inline station's backplane. The error-handling sequence is also applicable for a write sequence.

Events 1 to 4 Refer to the fragmented read example for an explanation of events 1 to 4.

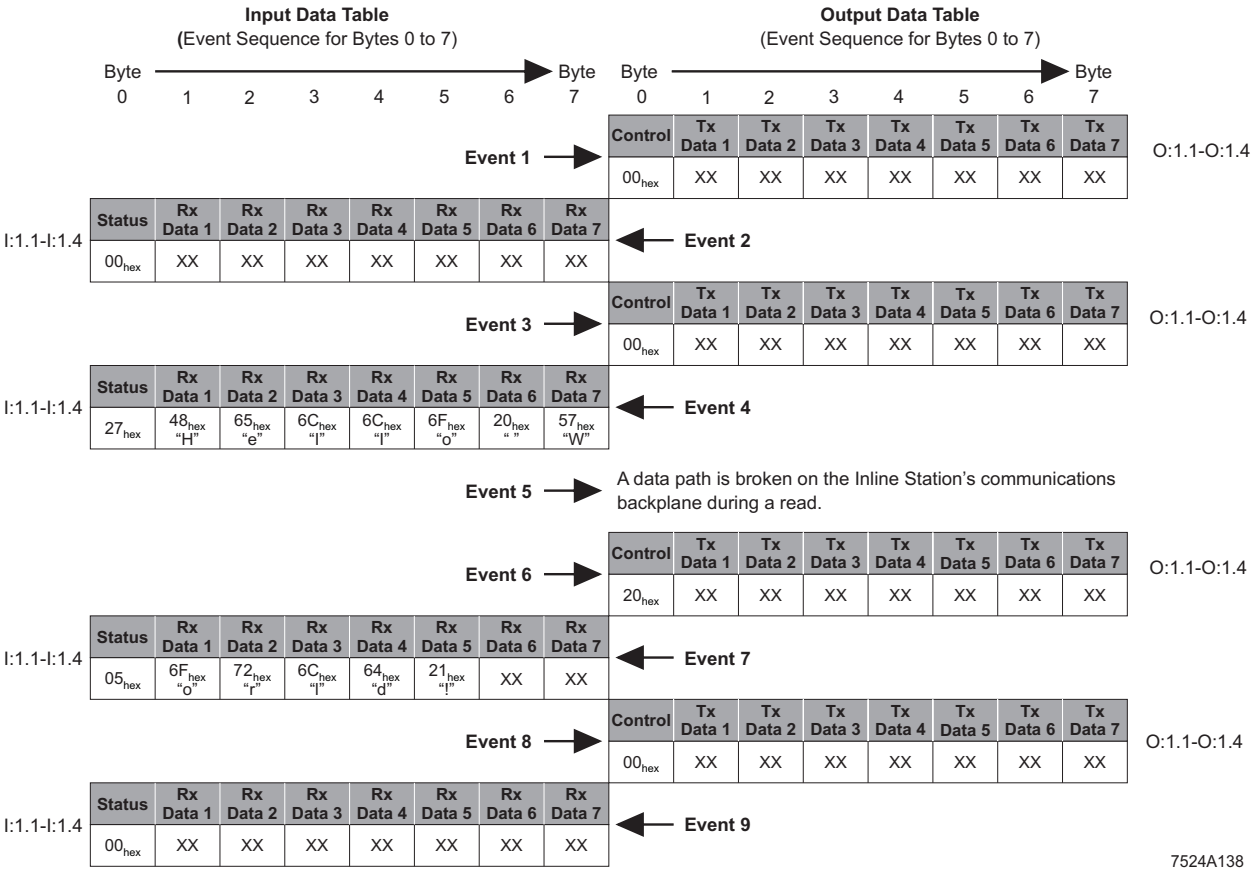


Figure A-3 Serial framentation error example 1

Event 5 An error has occurred. The communications path between the bus coupler and its I/O has been broken. If incoming serial data overflows the buffer on the serial module, overflow data will be lost.

Events 6 to 9 Error is repaired and the connection is reestablished thus data continues to be received.

Fragmented Error Handling, Loss of Communications from the Host to the Bus Coupler

If communications is lost between the host controller an the Inline bus coupler during a read or write service the bus coupler will wait for the error to be corrected (network cable is repaired) and then continue to finish the serial fragmentation transaction.

A 2.2 Method 2: Transfer of Serial Data Using Explicit Messages

The ability to send explicit messages is a function of the DeviceNet™ I/O scanner. Not all scanners have an explicit messaging channel available to the user. This manual will not document the mechanics of the actual sending of an explicit message. Information of this type must be provided in the documentation for the I/O scanner.

Configuration software can be an option to understand the structure of an explicit message before the message is actually integrated into the control program.

Explicit messages can be sent as an alternative to using fragmentation and the high-speed data channel. For this method the Serial Communications Object, Class Code 106_{dec} ($6A_{hex}$) will need to be directly accessed using an explicit message.

The Serial Communications Object contains attributes used for the sending and receiving of data to or from a serial module. When using Method 2, the 8 bytes of fragmentation produced and consumed data, as well as the Status and Control words can be removed from the scan. For easy access, it may be advantageous to allow the Status and Control words to remain in the scan.

A 2.2.1 Receiving Serial Data

Attribute 7 of the Serial Communications Object (SCO) is the Receive Data parameter. Using this attribute along with the status word, bit 0 “Receive Buffer is not Empty” , is required to receive data from a serial module. The status word (and a control word) from the serial module(s) is automatically added to the DeviceNet™ scan by default. (See the respective serial modules data sheet for more information on specific control functions and status capabilities.) The user will need to monitor bit 0. When bit 0 is set, there is data present from the serial module. At this point the user will send an explicit message with the following parameters:

Node address

Service Code	14_{dec} (Get_Attribute_Single)
Class =	106_{dec} (Serial Communications Object)
Instance =	1 (In this case the 1st occurrence of a serial module)
Attribute =	7 (Receive data)

Instance is determined by the physical location on the Inline station. The 1st instance will be assigned to the serial module (RS-232 or RS-485/RS-422) located closest to the bus coupler and the last instance (maximum of 8) will be assigned to the right most module.

When this message is sent, the bus coupler will send its response back to the sender. Typically an I/O scanner control bit is set when an explicit response is present. At that point the response can be read. The response will include a positive or negative confirmation.

The first data byte returned by bus coupler will be the number of bytes to follow. These “following” bytes are the actually data that was received by the serial module. Keep in mind that the user will need to understand the explicit message response format dictated by the I/O scanner. It is probable that several bytes of data pertaining to the response “header” will be returned before the actual data returned from the serial module is present. This header may include such information as: transaction ID, command, nodes address and confirmation (positive or negative).

A 2.2.2 Transmitting Serial Data

Attribute 8 of the Serial Communications Object (SCO) is the Transmit Data parameter. This attribute is required to transfer data to a serial module. To transmit data to the serial module the user will send an explicit message with the following parameters:

Node address

Service Code =	16_{dec} (Set_Attribute_Single)
Class =	106_{dec} (Serial Communications Object)
Instance =	1 (In this case the 1st occurrence of a serial module)
Attribute =	8 (Transmit data)

Instance is determined by the physical location on the Inline station. The 1st instance will be assigned to the serial module (RS-232 or RS-485/RS-422) located closest to the bus coupler and the last instance (maximum of 8) will be assigned to the right most module.

When this message is sent, the bus coupler will send its response back to the sender. Typically an I/O scanner control bit is set when an explicit response is present. At that point the response can be read. When sending a transmit command the user can expect a positive or negative confirmation in return.

The first byte transmitted to the bus coupler will be the number of bytes to follow. These “following” bytes are the actual data that is being sent to the serial module.

A 2.3 Method 3: Transfer of Data Using PCP Fragmentation

This method defines how PCP data is exchanged, using process/cyclic data, with an Inline module that supports the Peripherasl Communication Protocol (PCP) and is not a serial module. An example of this type of module would be the Inline AS-i-Gateway (ASI MA IB IL).

Typically the ASI MA IB IL will not require the use of PCP data exchange. However PCP data exchange will be required if there are more than 31 slaves on the AS-i “subnetwork”.

The messages that are required to read and write data are encoded into the high-speed data stream. The protocol is handled using a series of message fragments that are initiated by a client start request and then followed up with a server response. These fragments were specifically designed to be used with any Inline PCP modules.

These process data messages are used to read or write to a specific slave device’s memory location that is access by an Index and subindex designation. Beside the exchange of normal I/O data PCP process data communications can be used to parameterize an Inline module or retrieve informative data.

Information pertaining to the supported indexes, subindexes and Invoke ID can be found in the specific module’s data sheet and/or manual.

For each PCP Inline module, by default, 8 bytes (1 fragment) are added to the DeviceNet™ produced size and eight bytes are added to the consumed size. These eight bytes can only be used to send PCP data messages and are in addition to any other I/O data that might also be added into the scan. This type of information can be found in the specific module's data sheet.

An example of this type of information would be a status and control word. These two bytes would also be added to the produced size and to the consumed size in addition to the eight bytes allocated for the “I/O messaging” connection. For this example there would be a total of 10 produced bytes and 10 consumed bytes allocated for this one Inline PCP module. When using PCP fragmentation a request will be sent and a response will be returned the format of a request and a response is as follows:

Request (Output Data) – NO Invoke ID

Byte 0	Service
Byte 1	Module number
Byte 2	Index low
Byte 3	Index high
Byte 4	Subindex
Byte 5	Length
Byte 6 to N	Data block, if necessary

Request (Output Data) – With Invoke ID

Byte 0	Service
Byte 1	Invoke ID
Byte 2	Module number
Byte 3	Index low
Byte 4	Index high
Byte 5	Subindex
Byte 6	Length
Byte 7 to N	Data block, if necessary

Successful Response: (Input Data) – NO Invoke ID

Byte 0	Service
Byte 1	Status
Byte 2	Length
Byte 3 to N	Data block, if necessary

If the response was not successful the response will be returned in the following format:

Errored Produced Response: (Input Data) – NO Invoke ID

Byte 0	Service
Byte 1	Status
Byte 2	Error class
Byte 3	Error code
Byte 4	Additional error code 1 LSB, if necessary
Byte 5	Additional error code 1 MSB, if necessary
Byte 6	Additional error code 2 LSB, if necessary
Byte 7	Additional error code 2 MSB, if necessary

Successful Response: (Input Data) – With Invoke ID

Byte 0	Service
Byte 1	Invoke ID
Byte 2	Status
Byte 3	Length
Byte 4 to N	Data block, if necessary

If the response was not successful the response will be returned in the following format:

Errored Produced Response: (Input Data) – With Invoke ID

Byte 0	Service
Byte 1	Invoke ID
Byte 2	Status
Byte 3	Error class
Byte 4	Error code
Byte 5	Additional error code 1 LSB, if necessary
Byte 6	Additional error code 1 MSB, if necessary
Byte 7	Additional error code 2 LSB, if necessary
Byte 8	Additional Error Code 2 MSB, if necessary

The Response Service byte is a reflection of the request service byte, with the exception of the request/response bit (see the definition of the Start Fragment for more information on this bit).

It is important to keep track of the "client-service" relationship between the master and the slave. For example, a read request involves a single non-fragmented service. The slave responds with a completely new service to transfer the response data. This new service begins with a start fragment and ends (if enough data was requested) with a last fragment. The write request on the other hand, starts with a start fragment and ends (if enough data was sent) with a last fragment. The slave responds with a non-fragmented start fragment.

A 2.3.1 Fragment Types

Four transfer-fragment types are distinguished by the service-byte (byte 0 of each fragment). The types are as follows:

– Start fragment
– Middle fragment
– Last fragment
– Abort/Error fragment

Start Fragment

To begin any message a start fragment must be sent. The start fragment's eight bytes have the following format:

Byte 0	Service
Byte 1	Module number
Byte 2	Index low
Byte 3	Index high
Byte 4	Subindex
Byte 5	Length
Byte 6	Data block, if necessary
Byte 7	Data block, if necessary

Byte 0, Service Definition of the Start Fragment (Shown in Table A-16)

Table A-16 Service byte definition of the start fragment

7	6	5	4	3	2	1	0
Request/ Response	0	0	No Fragment/ Fragment	PCP Service			

Bit 7 Request/Response

The request response bit is set when the actual Inline PCP module responds to the PCP service that the user requested. The amount of time that it takes to process this service depends on the PCP channel size, the number of Inline modules, and the actual service requested. The user can use this bit to know when the service is actually processed by the Inline PCP module. For a read request, this bit will be set by the bus coupler as soon as the read request is called. For a write request, this bit is set as soon as all data reaches the PCP module.

0	Request (Service in process)
1	Response (Service is processed)

Bit 6 to 5 Fragment Type

For a start fragment these bits will always be a zero.

00	Start - fragment

Bit 4 Fragmented

Bit 4 informs the slave as to whether the message contains more than eight bytes (7 data bytes) or not.

0	Does not fragment
1	Fragments

Bit 3 to 0 PCP Service

Bits 3 to 0 informs the slave of the type of message (service) being sent. The means are described as follows:

00_{hex}	No action (Clears input bytes in response)
01_{hex}	Read
02_{hex}	Write
03_{hex}	Read PDU length
04_{hex} to $0F_{hex}$	Reserved

Middle Fragment

Any write request or read response requires more than the available number of data bytes in both a start fragment and a last fragment, then a middle fragment must be used. Middle fragments have the following format.

Byte 0	Service
Byte 1	Data block, if necessary
Bytes 2 to 7	Data blocks, if necessary

Byte 0, Service Definition of a Middle Fragment (Shown in Table A-17)

Table A-17 Service byte definition of the middle fragment

7	6	5	4	3	2	1	0
Request/ Response	0	1	Fragment Number (01_{hex} to $1F_{hex}$)				

Bit 7 Request/Response
See start fragment for definition.

Bit 6 to 5 Fragment Type
For a middle fragment these bits will always be a 01.

01 Middle fragment

Bit 4 to 0 Count
Bits 4 to 0 keep track of how many middle fragments have been sent. 31 is the maximum number of middle fragments that can be counted (01_{hex} to $1F_{hex}$). If more fragments are needed, the fragment number will roll over to 0 and fragments can continue to be sent.

Last Fragment

To recognize the end of a fragmented message, a last fragment must be issued. A last fragment has the following format:

Byte 0	Service
Byte 1 to 7	Data block, if necessary

Byte 0, Service Definition of the Last Fragment (see Table A-18)

Table A-18 Service byte definition of the last fragment

7	6	5	4	3	2	1	0
Request/ Response	1	0	Reserved				

Bit 7 Request/Response
See start fragment for definition.

Bit 6 to 5 Fragment Type
For a last fragment these bits will always be a 10.

10 Last fragment

Bit 4 to 0 Reserved

Abort/Error fragment: If a transmission error is detected an abort/error fragment will be generated.

Byte 0, Service Definition of a PCP Abort/Error Fragment (see Table A-19)

Table A-19 Service byte definition of the PCP abort / error fragment

7	6	5	4	3	2	1	0
Request/ Response	1	1	Reserved				

Bit 7 Request/Response
See start fragment for definition.
0 Request
1 Response

Bit 6 to 5 Fragment Type
For an abort/error fragment, these bits will always be an 11.
11 Abort / Error fragment

Bit 4 to 0 Reserved

Byte 1, Status Definition of a Abort/Error Fragment (See Table A-20)

Table A-20 Service byte definition of the PCP abort/error fragment

7	6	5	4	3	2	1	0
Reserved				Fragmen tation Error	PCP Channel Busy	PCP Error	Module Comm. Error

Bit 7 to 4 Reserved

Bit 3 Fragmentation Error:
An error has occurred with either the type or sequence of the fragments (i.e. middle received after last, middle fragment 2 received before 1, etc).

Bit 2 PCP Channel Busy
A PCP transaction is already in progress, such as from an explicit request.

Bit 1 PCP Error
A PCP service-specific error has occurred. See the Error Class and Error Code bytes.

Bit 0 Module Comm. Error
When set, communication with the module is no longer possible.

Byte 2-7, Error Class and Error Code

Bytes 2 and 3 will display the error class and error code. If there is any addition error information it will be displayed in bytes 4 to 7. To interpret the error information, a PCP reference manual must be consulted.

A 2.3.2 PCP Fragmentation Examples

The following examples will show how to read, write and handle errors using PCP process data fragmentation. The examples to follow are:

– Fragmented Read
– Fragmented Write
– Error Handling, Communications Backplane Break
– Error Handling, Host Communications Loss

Fragmented Read Example

Figure A-4 shows an example I/O data table that contains only eight bytes of input data and eight bytes of output data. This I/O is shown in an event by event sequence that demonstrates how these eight bytes of I/O are updated when reading data using PCP fragmentation. The sequence works on this following basic principle:

Event x	Client issues server 8 bytes of output data
Event x+1	Server responds by updating 8 bytes of input data

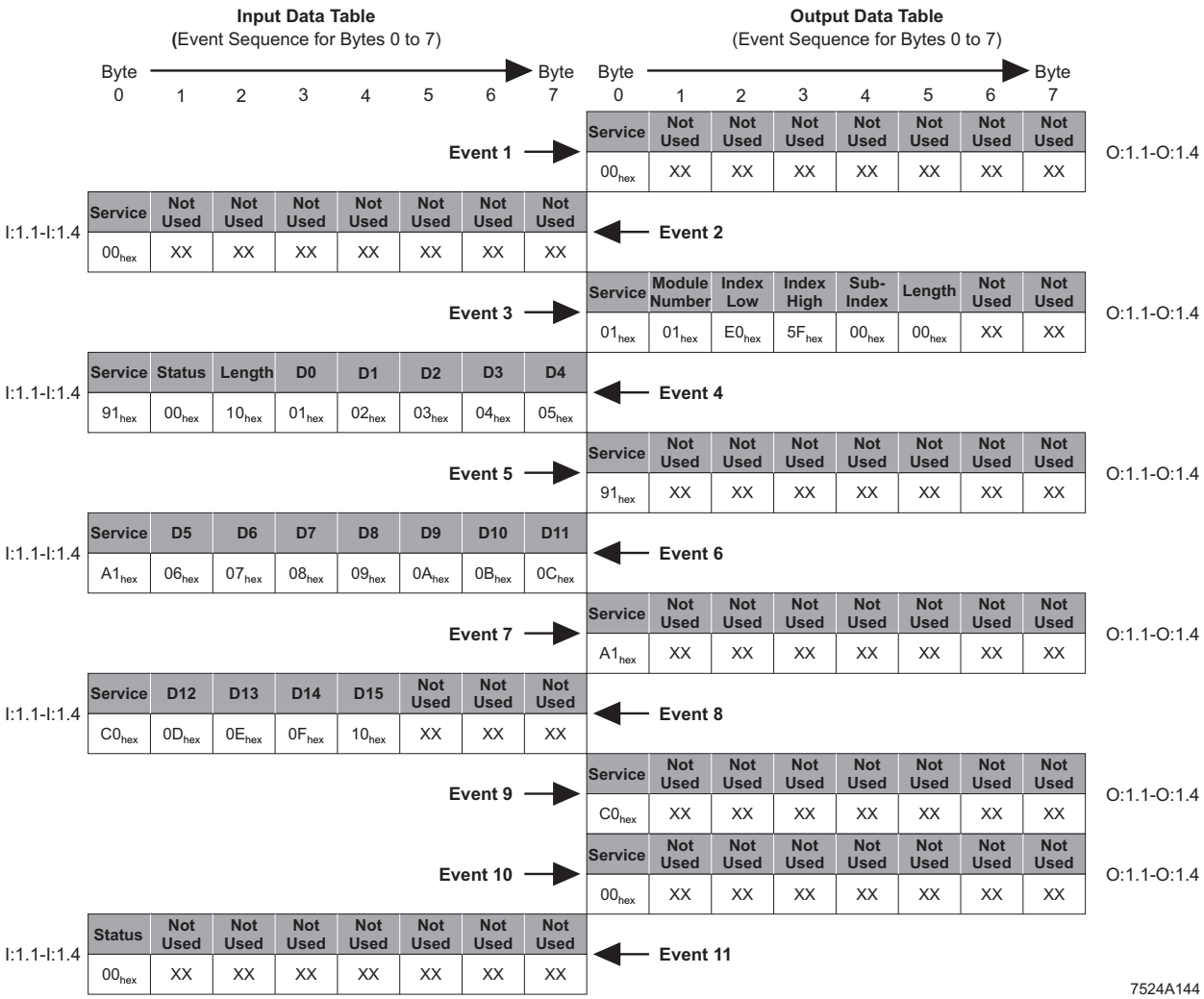


Figure A-4 I/O events for a read sequence using PCP fragmentation

In the following paragraphs the events in Figure A-4 will be explained in detail.

Table A-21 to Table A-31 demonstrate the order of events when issuing a fragmented read service to a PCP device. Each “Event” should be referenced to the I/O data table shown in Figure A-4.

Event 1

Table A-21 shows the transmission from a master to slave when the PCP fragmentation is in “Idle” mode. Note that in byte 0 the service 00_{hex} is being sent.

Table A-21 Master to slave idle request, sending a 00_{hex} no action service

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 2

Table A-22 shows the slaves response to the idle state by replying with a 00_{hex} no action acknowledge.

Table A-22 Slave to master idle response, 00_{hex} no action acknowledgement

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 3

Table A-23 shows how the start fragment request is sent from the master to the slave to initiate a read. This message is built with the format shown below. Note that the target index is $5FE0_{hex}$.

Service, byte 0	01_{hex}	Read
Module Number, byte 1	01_{hex}	First PCP module on the Inline station
Index Low, byte 2	$E0_{hex}$	Low byte of the PCP index to be read
Index high, byte 3	$5F_{hex}$	High byte of the PCP index to be read
Subindex, byte 4	00_{hex}	Subindex is zero
Length, byte 5	00_{hex}	No length requirement
Bytes 6 and 7	XX	Not used

Table A-23 Master to slave read request, sending a 01_{hex} service

Service	Module Number	Index Low	Index High	Sub-Index	Length	Not Used	Not Used
01_{hex}	01_{hex}	$E0_{hex}$	$5F_{hex}$	00_{hex}	00_{hex}	XX	XX

Event 4

Table A-24 shows the response from the slave that includes the first 5 bytes of actual data that was requested from index $5FE0_{hex}$. This reply is explained as follows:

Table A-24 Slave to master, response to the read request

Service	Status	Length	D0	D1	D2	D3	D4
91_{hex}	00_{hex}	10_{hex}	01_{hex}	02_{hex}	03_{hex}	04_{hex}	05_{hex}

Service, byte 0	91_{hex}	
	Bit 7 = 1	This signifies that the fragment is a response
	Bit 6 = 0	This signifies a start fragment
	Bit 5 = 0	This signifies a start fragment
	Bit 4 = 1	This denotes that the response will be sent in fragments
	Bits 3 to 0 = 1	This signifies a read service
Status	00_{hex}	No errors
Length	10_{hex}	Informs the master that there will be 16 data bytes in the message
D0 - D4		First 5 bytes of the message (bytes D5-D15 to be sent in the following fragments)

It's important to realize that event 4 is really the start fragment of the slave's message containing the requested data.

Event 5

Table A-25 shows the master to slave acknowledgment of the response being received. In this acknowledge the service byte reflection is the indication that the 1st response was received.

Table A-25 Master to slave acknowledgement

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
91_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 6

Since the reception of the first 5 bytes of data has been acknowledged the slave is ready to send the next fragment. This second fragment is a middle fragment with a slightly different format. Table A-26 shows the response from the slave that includes the service byte and 7 more bytes of data that was requested from index $5FE0_{hex}$. This reply is explained as follows:

Table A-26 Slave to master, reply with first middle fragment

Service	D5	D6	D7	D8	D9	D10	D11
$A1_{hex}$	06_{hex}	07_{hex}	08_{hex}	09_{hex}	$0A_{hex}$	$0B_{hex}$	$0C_{hex}$

Service, byte 0	$A1_{hex}$	
	Bit 7 = 1	This signifies that the fragment is a response
	Bit 6 = 0	Designates a middle fragment when bit 6 = 0 and bit 5 = 1
	Bit 5 = 1	Designates a middle fragment when bit 6 = 0 and bit 5 = 1
	Bit 4 to 0 = 1	These bit count the fragments. 1 = the 1st middle fragment
Bytes 1 to 7 = 7		Specifies 7 additional bytes of data (5 received in the first response)

Event 7

Table A-27 shows the master to slave acknowledgment of the response being received. In this acknowledge the service byte reflection is the indication that the 1st middle fragment response was received.

Table A-27 Master to slave acknowledge of the first middle fragment

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
$A1_{hex}$	XX	XX	XX	XX	XX	XX	XX

Event 8

Table A-28 shows the slave to master reception of the last fragment that was requested from index $5FE0_{hex}$. This fragment returns the last 4 data bytes as expected by the length byte shown in Figure A-4. This response is explained as follows:

Table A-28 Slave to master acknowledge last fragment response

Service	D12	D13	D14	D15	Not Used	Not Used	Not Used
$C0_{hex}$	$0D_{hex}$	$0E_{hex}$	$0F_{hex}$	10_{hex}	XX	XX	XX

Service, byte 0	$C0_{hex}$	
	Bit 7 = 1	This signifies that the fragment is a response
	Bit 6 = 1	Designates a last fragment when bit 6 = 1 and bit 5 = 0
	Bit 5 = 0	Designates a last fragment when bit 6 = 1 and bit 5 = 0
	Bits 4 to 0	Reserved

Event 9

Table A-29 shows the master to slave acknowledgment of the last fragment being received. In this acknowledge the service byte reflection is the indication that the last fragment response was received.

Table A-29 Master to slave last fragment acknowledgment

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
$C0_{hex}$	XX	XX	XX	XX	XX	XX	XX

Events 10 and 11

Knowing that the last fragment was received the master issues an idle service to the slave as shown in Table A-30 and the slave respond with its reply as shown in Table A-31.

Table A-30 Master to slave idle service

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Table A-31 Slave to master idle service response

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Fragmented Write Example

Figure A-5 shows an example I/O data table that contains only eight bytes of input data and eight bytes of output data. This I/O is shown in an event by event sequence that demonstrates how these eight bytes of I/O are updated when reading data using PCP fragmentation. The sequence works on this following basic principle:

Event x	Client issues server 8 bytes of output data
Event x+1	Server responds by updating 8 bytes of input data

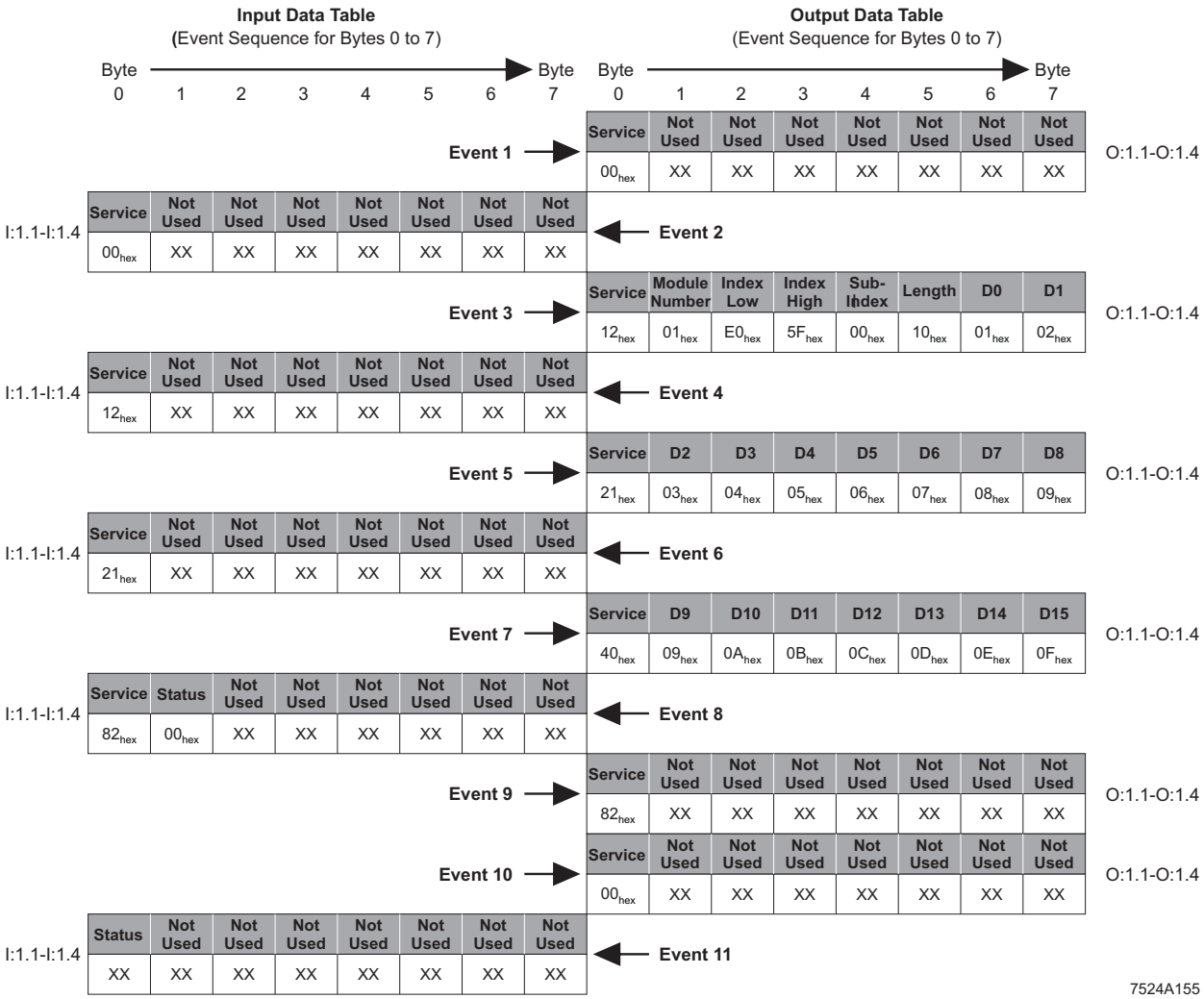


Figure A-5 I/O events for a write sequence using PCP fragmentation

In the following paragraphs the events in Figure A-5 will be explained in detail.

Table A-32 to Table A-42 demonstrate the order of events when issuing a fragmented write service to a PCP device. Each “Event” should be referenced to the I/O data table shown in Figure A-5.

Event 1

Table A-32 shows the transmission from a master to slave when the PCP fragmentation is in “Idle” mode. Note that in byte 0 the service 00_{hex} is being sent.

Table A-32 Master to slave idle request, sending a 00_{hex} no action service

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 2

Table A-33 shows the slaves response to the idle state by replying with a 00_{hex} no action acknowledge.

Table A-33 Slave to master idle response, 00_{hex} no action acknowledgement

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 3

Table A-34 shows how the start fragment request is sent from the master to the slave to initiate a write. This message is built with the format shown below. Note that the target index is $5FE0_{hex}$.

Table A-34 Master to slave write request, sending a 02_{hex} service

Service	Module Number	Index Low	Index High	Sub-index	Length	D0	D1
12_{hex}	01_{hex}	$E0_{hex}$	$5F_{hex}$	00_{hex}	10_{hex}	01_{hex}	02_{hex}

Service, byte 0	12_{hex}	Fragmented write
	Bit 4	Indicates that the write is fragmented
	Bits 0 to 3	Write service (02_{hex})
Module Number, byte 1	01_{hex}	First PCP module on the Inline station
Index Low, byte 2	$E0_{hex}$	Low byte of the PCP index to be read
Index High, byte 3	$5F_{hex}$	High byte of the PCP index to be read
Subindex, byte 4	00_{hex}	Subindex is zero
Length, byte 5	10_{hex}	16 bytes
Bytes 6 and 7	XX_{hex}	First 2 data bytes

Event 4

Table A-35 shows the acknowledge from the slave that indicates the write request fragment was processed. In this acknowledge the service byte reflection is the indication that the 1st response was received.

Table A-35 Slave to master, acknowledgement of the write service

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
12_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 5

Since the reception of the first 2 bytes of data has been processed and acknowledged the master is ready to send the next fragment. This second fragment is a middle fragment with a slightly different format. Table A-36 shows the service byte and the next 7 bytes of data to be written to index $5FE0_{hex}$. This request is explained as follows:

Table A-36 Master to slave, sending the 1st middle fragment

Service	D2	D3	D4	D5	D6	D7	D8
21_{hex}	03_{hex}	04_{hex}	05_{hex}	06_{hex}	07_{hex}	08_{hex}	09_{hex}

Service, byte 0	21_{hex}	
	Bit 7 = 0	This signifies that the fragment is a request
	Bit 6 = 0	Designates a middle fragment when bit 6 = 0 and bit 5 = 1
	Bit 5 = 1	Designates a middle fragment when bit 6 = 0 and bit 5 = 1
	Bits 4 to 0 = 1	These bit count the fragments. 1 = the 1st middle fragment
Bytes 1 to 7 = 7		More bytes of data (2 sent with first request)

Event 6

Table A-37 shows the slave to master acknowledgment. The service byte reflection indicates that the 1st middle fragment data was received and processed.

Table A-37 Slave to master, acknowledgement of 1st middle fragment

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
21_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 7

Since the reception of the first middle fragment of data has been acknowledged the master is ready to send the next fragment. This next fragment is the last fragment. Table A-38 shows the last fragment that includes the service byte and 7 more bytes of data that is being sent to index $5FE0_{hex}$ (16 bytes total). This last fragment request is explained as follows:

Table A-38 Master to slave last fragment

Service	D2	D3	D4	D5	D6	D7	D8
40_{hex}	09_{hex}	$0A_{hex}$	$0B_{hex}$	$0C_{hex}$	$0D_{hex}$	$0E_{hex}$	$0F_{hex}$

Service, byte 0	40_{hex}	
	Bit 7 = 0	This signifies that the fragment is a request
	Bit 6 = 1	Designates a last fragment when bit 6 = 1 and bit 5 = 0
	Bit 5 = 0	Designates a last fragment when bit 6 = 1 and bit 5 = 0
	Bits 4 to 0 = 0	Reserved
Bytes 1 to 7		Last 7 bytes of data

Event 8

Table A-39 shows the slave to master acknowledgment of the last fragment being received and processed. In this acknowledge the service byte reflection is the indication that the last fragment data was received. It is important to realize that event 8 is really a start fragment from the slave signalling the response and status information for the write request.

Table A-39 Slave to master last fragment acknowledgement

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
82_{hex}	XX	XX	XX	XX	XX	XX	XX

Event 9

Table A-40 shows the master to slave acknowledgment of the last fragment being received. In this acknowledge the service byte reflection is the indication that the last fragment response was received.

Table A-40 Master to slave acknowledgement

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
82_{hex}	XX	XX	XX	XX	XX	XX	XX

Events 10 and 11

Knowing that the last fragment was received the master can issue an idle service to the slave as shown in Table A-41 and the slave respond with its reply as shown inTable A-42.

Table A-41 Master to slave idle

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Table A-42 Slave to master response to idle

Service	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
00_{hex}	XX	XX	XX	XX	XX	XX	XX

Fragmented Error Handling, Loss of Inline Backplane Communications

This PCP fragmentation error handling example, shown in Figure A-6, will show how the fragmentation will react during a break in the communications path on the Inline station's backplane.

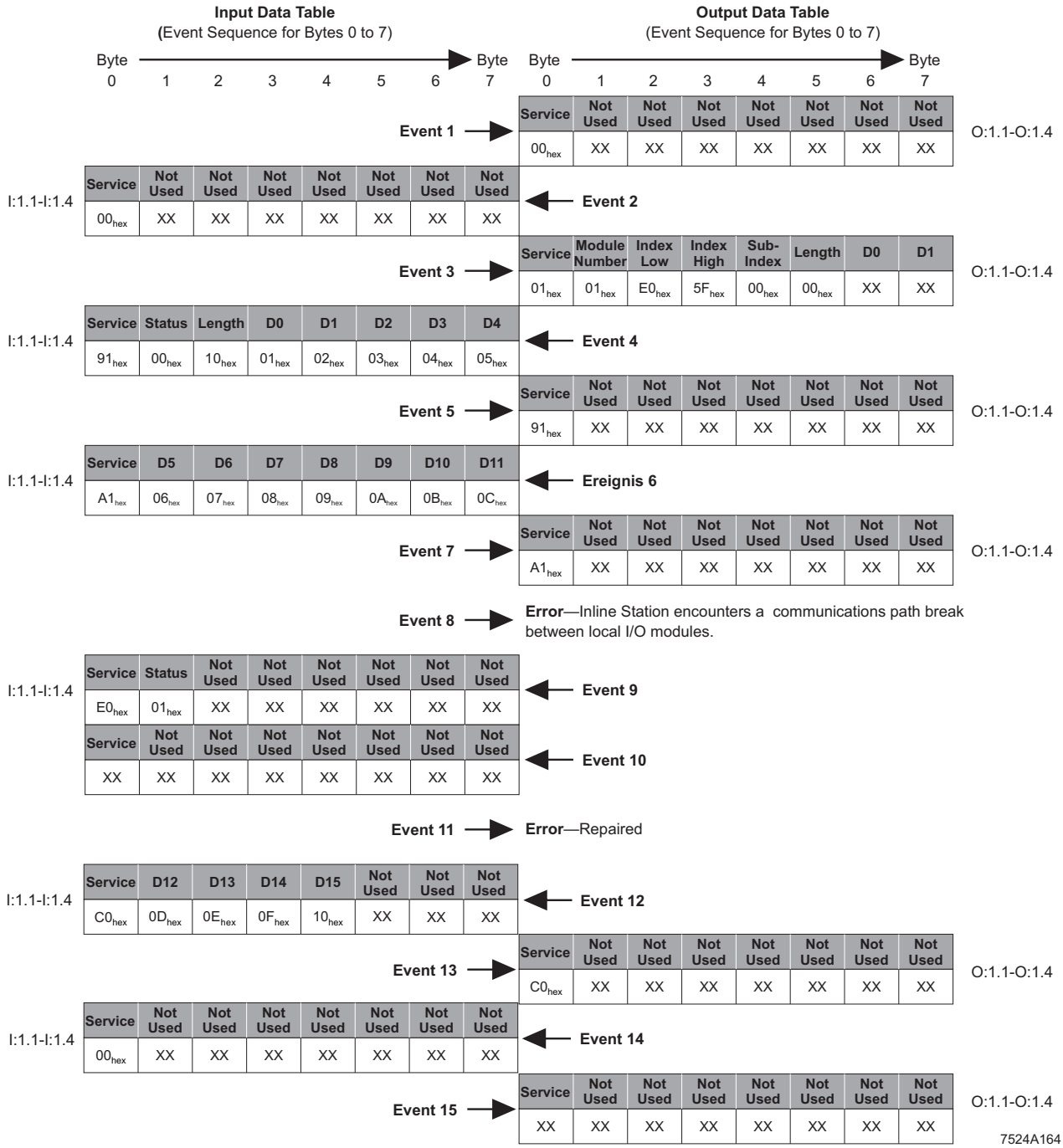


Figure A-6 Local communications error sequence using PCP fragmentation

Events 1 to 7

For an explanation of events 1 to 7, look at the examples in this section for a read sequence.

Event 8

An error has occurred. The communications path between the bus coupler and its I/O has been broken.

Event 9

A PCP Abort/Error fragment has been issued from the slave to the master as shown in Table A-43.

Table A-43 Abort/error fragment

Service	Status	Not Used	Not Used	Not Used	Not Used	Not Used	Not Used
$E0_{hex}$	01_{hex}	XX	XX	XX	XX	XX	XX

Service, byte 0	$E0_{hex}$	
	Bit 7 = 1	This signifies that the fragment is a response
	Bit 6 = 1	Designates an abort/error fragment when bit 6 = 1and bit 5 = 1
	Bit 5 = 1	Designates an abort/error fragment when bit 6 = 1 and bit 5 = 1
	Bits 4 to 0 = 0	Reserved
	Byte 1 = 1	Identifies a module communication error
Bytes 2 to 7	XX	Not used

Event 10

Data is no longer being received in the input table.

Once the connection is reestablished the data will continue to be sent.

Event 11

In this example the communications error is corrected at this point.

Events 12-15

Read service is completed as described previously in the section.

Fragmented Error Handling, Loss of Communications from the Host to the Bus Coupler

If communications is lost between the host controller an the Inline bus coupler during a read or write service the bus coupler will wait for the service to be completed (network cable is repaired) and acknowledge the completion of the service once the transaction has ended.

A 2.4 Method 4: Transfer PCP Data Using Explicit Messages

Explicit messages can be sent as an alternative to using the high-speed data channel and PCP fragmentation to send PCP messages. For method 4 the PCP Special Function Object, Class Code 105_{dec} (69_{hex}) will need to be directly accessed using an explicit message. By default this method does not apply to serial modules. Refer to "Communications Methods" on page A-1 in this section.

The ability to send explicit messages is a function of the DeviceNet™ I/O scanner. Not all scanners have an explicit messaging channel available to the user. This manual will not document the mechanics of the actual sending of an explicit message. Information of this type must be provided in the documentation for the I/O scanner.

Configuration software can be an option to understand the structure of an explicit message before the message is actually integrated into the control program.

The PCP Special Function Object, Class Code 105_{dec} (69_{hex}) is detailed further in Section B, "DeviceNet™ Object Classes, Message Types and Services".

When sending PCP messages using explicit messages I/O scan size can be reduced by removing the eight bytes of fragmentation data that is added to the produced and consumed sizes by default. This can be accomplished by using the PCP Special Function object.

A 2.4.1 Reading PCP Data

When reading PCP data from a PCP module that is not a serial based module, the PCP Special Function Object, Class Code 105_{dec} (69_{hex}) must be used. Within this object, there are two sub-methods that can be used to read data explicitly. Sub-method A involves directly requesting/reading the complete PCP services using attributes 6 and 7. Sub-method B provides greater efficiency. It fixes the module number, index, and subindex so that only the data is received for each request (attributes 8, 12, 13, and 14). The following procedure uses sub-method B.

Three explicit messages will be required to read a specific PCP memory area. They are as follows:

1. Select index
2. Select subindex
3. Read data

These messages will have the format shown below and will need to be sent with the required service.

1. Build Select Index Message: Message is sent using the Set_Attribute_Single service (16_{dec})

Class Code	105_{dec} (69_{hex}), PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	12_{dec}, PCP Read Index (Example: Index number $5FE0_{hex}$)

2. Build Select Subindex Message: Message is sent using the Set_Attribute_Single service (16_{dec})

Class Code	105_{dec} (69_{hex}), PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	13_{dec}, PCP Read Subindex (Example: Subindex number 0)

3. Build Read Data Message: Message is sent using the Get_Attribute_Single service (14_{dec})

Class Code	105_{dec} (69_{hex}),PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	14_{dec}, PCP Read Data

Instance is determined by the physical location on the Inline station. The 1st instance will be assigned to the PCP module located closest to the bus coupler and the last instance (maximum of 8) will be assigned to the right most PCP module. Serial modules will occupy an instance in this object.

The PCP module attribute (attribute 8) defaults to the instance value.

When message 3 is sent, the bus coupler will send the data back to the sender.

The first data byte returned by bus coupler will be the number of bytes to follow. These “following” bytes are the actual data that was sent by the PCP module.

A 2.4.2 Sending PCP Data

When sending PCP data from a PCP module that is not a serial based module, the PCP Special Function Object, Class Code 105_{dec} (69_{hex}) must be used. Within this object, there are two sub-methods that can be used to write data explicitly. Sub-method A involves directly requesting/writing the complete PCP services using attributes 6 and 7. Sub-method B provides greater efficiency. It fixes the module number, index, and subindex so that only the data is sent for each request (attributes 8, 9, 10, and 11). The following procedure uses sub-method B.

Three explicit messages will be required to write a specific PCP memory area. They are:

1. Select index
2. Select subindex
3. Write data

These messages will have the format shown below and will need to be sent with the required service.

1. Build Select Index Message: Message is sent using the Set_Attribute_Single service (16_{dec})

Class Code	105_{dec} (69_{hex}), PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	9_{dec}, PCP Write Index (Example: Index number $5FE0_{hex}$)

2. Build Select Subindex Message: Message is sent using the Set_Attribute_Single service (16_{dec})

Class Code	105_{dec} (69_{hex}), PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	10_{dec}, PCP Write Subindex (Example: Subindex number 0)

3. Build Read Data Message: Message is sent using the Set_Attribute_Single service (16_{dec})

Class Code	105_{dec} (69_{hex}), PCP Special Function Object
Instance	Select the occurrence of the PCP device within the object
Attribute	11_{dec}, PCP Write Data
Data	First byte contains the number of bytes to be sent, then the actual data

Instance is determined by the physical location on the Inline station. The 1st instance will be assigned to the PCP module located closest to the bus coupler and the last instance (maximum of 8) will be assigned to the right most PCP module. Serial modules will occupy an instance in this object.

The PCP module attribute (attribute 8) defaults to the instance value.

When message 3 is sent, the bus coupler will receive the data from the sender.

The first data byte returned by bus coupler will be the number of bytes to follow. These “following” bytes are the actual data that was sent by the PCP module.

A 3 Serial and Generic PCP Modules Produced and Consumed Sizes

Section 3, "Configuration and Operation" provides additional information in regards to how data is mapped into the scanner and other considerations.

A 3.1 Determining Produced and Consumed Size

The bus coupler can auto-configure itself to the Inline I/O connected to it (Refer to section 3, "Configuration and Operation"). Once this is done the total number of produced and consumed data, for the entire Inline station (will include fragmentation data), can be read from the EDS file.

By default any module that uses the PCP protocol (includes the serial modules) will have 8 bytes of produced data and 8 bytes of consumed data added to the network scan. These bytes are used to transfer data back and forth between a DeviceNet™ scanner and, for example, an Inline serial module. This data transfer using 8 bytes is required when using process/cyclic data to Tx/Rx serial data (fragmentation).

In addition to the bytes used for transferring serial or other PCP data there may be additional data produced or consumed by the I/O module. This additional data must be added to the 8 bytes described in the previous paragraph. The number of additional process data bytes can be found in the specific Inline module's data sheet or manual. These process data bytes are will be added to the scan ahead of the 8 bytes of fragmentation data in the scanner's I/O memory. Table A-44 gives an example of calculating the total number of bytes produced and consumed by an individual Inline RS-232 module.

Table A-44 Calculation of a serial module's produced and consumed bytes

	Produced	Consumed
Bytes required by the RS-232 module for status and control (found in data sheet)	2	2
	+	+
Bytes required to Tx/Rx serial data (fragmentation)	8	8
Total used by each RS-232 module	10 bytes	10 bytes

A 3.2 Removing or Adding Fragmentation Data

Fragmentation data must not be removed if using methods 1 or 3 described in section A 2, "Communications Methods".

If serial or other PCP data is going to be transmitted using explicit messages, then the 8 bytes used for the process/cyclic data messaging (fragmentation) that is added to the scan by default, will not be needed. The unused 8 bytes of produced and 8 bytes of consumed should be removed to ensure the best possible network performance.

If the user has a serial module and wants to remove or add the fragmentation data (8 bytes) they must send the following explicit messages. This must be sent to the Serial Communications Object, Class Code 106_{dec} ($6A_{hex}$), using the proper instance and attribute 32, “Fragment Data in DeviceNet™ I/O”.

To add or remove the Serial Data (8 bytes of fragment data) from the DeviceNet™ I/O, set the following:

Class 106

Instance X

Attribute 32 (Serial Communications Object Fragment Data)

0 Removes data

1 Adds data

If the user has any other PCP module and wants to add/remove the fragmentation data (8 bytes) they must send an explicit message. This must be sent to the PCP Special Function Object Class Code 105_{dec} (69_{hex}), using the proper instance and attribute 17, “PCP Fragment Data in DeviceNet™ I/O”.

To add or remove the Other PCP Data from the DeviceNet™ I/O, set the following:

Class 105

Instance X

Attribute 17 (PCP fragment data)

0 Removes data

1 Adds data

A 4 I/O Memory Mapping, Serial and Special Function PCP Modules

Section 3, "Configuration and Operation" provides additional information in regards to how data is mapped into the scanner and other considerations.

I/O Mapping Rules

Section 3, "Configuration and Operation" describes configuration methods (mapping) in greater detail.

The I/O image in the bus coupler flash memory contains all produced-data (input data) and consumed-data (output data) derived from the I/O modules connected to it. I/O image data is added to the poll through the use of parameter 9 (Add All I/O), or by using auto configuration.

An I/O image could contain the Inline Status word (included by default in the produced data), command byte (not included in the consumed data by default), module fault data, reserved I/O space, digital, analog, special function (no PCP), special function PCP modules or serial modules (process data first then fragmentation data).

Mapping priority is determined by the type of module without regard to its location to the BK or other modules of different types. However, it does take into account the order of modules of the same type that exist on the station.

A 5 Configuration Brief for the RS-232 and RS-485/RS-422 Modules

General Configuration

Section B, "DeviceNet™ Object Classes, Message Types and Services" provides details of the Serial Communications Object (Class Code 106_{dec}, $6A_{hex}$) for configuration attributes.

This section describes how to change default settings for the Inline RS-232 and RS-485/ RS-422 modules. Information provided in this section must be used in conjunction with the specific module's data sheet.

In order to change any setting, refer to the module's specific data sheet to determine the appropriate attribute settings for the Serial Communications Object (Class Code 106_{dec}, $6A_{hex}$).

The default settings can only be changed by sending an explicit message. An explicit message can either be sent from a control program or a DeviceNet™ configuration software package. Once the desired parameters have been updated, the settings are stored in flash memory of the bus coupler. If the bus coupler is replaced, the serial configuration will need to be sent again, unless the configuration explicit messages are embedded into the control program.

The explicit message format required to configure the RS-232 or RS-485/RS-422 modules is as follows:

Service	Set_Attribute_Single, 16_{dec} (10_{hex})
Class Code	106_{dec} ($6A_{hex}$) Serial Communications object
Instance	1 (1st serial module)
Attribute	12 This attribute is used to modify the baud rate
Data	08 (Refer to the RS-232 module's data sheet) Code "08" represents a baud rate of 19.2K

Instance is the occurrence of the module within the Serial Communications object. Instances are assigned by the physical order of the serial Inline modules on the station starting with the module closest to the bus coupler being assigned to instance 1. The next module to follow will be assigned to instance 2 and so on up to a maximum of 8 instances. (There is a maximum of 8 PCP modules of any kind allowed to reside on the Inline station.)

Both the RS-232 and RS-485/RS-422 modules occupy instances in the Serial Communications object. If one of each reside on the station the closest to the bus coupler will be assigned to instance 1and the other will be assigned to instance 2.

B DeviceNet™ Object Classes, Message Types and Services

B 1 DeviceNet™ Message Types

The IL DN BK DI8 DO4-PAC supports the Group 2 Offline Connection set.

CAN IDENTIFIER	GROUP 2 Message Type
11111101100	Communication Faulted Response Message
11111101101	Communication Faulted Request Message
11111101110	Communication Ownership Response Message
11111101111	Communication Ownership Request Message

B 2 DeviceNet™ Class Services

The IL DN BK DI8 DO4-PAC supports the following class services and instance services.

Service Code		Service Name
dec	hex	
05	05	Reset
14	0E	Get_Attribute_Single
16	10	Set_Attribute_Single

B 3 DeviceNet™ Object Classes

The IL DN BK DI8 DO4-PAC device supports the following DeviceNet™ object classes.

Class Code		Object Type
dec	hex	
01	01	Identity Object
02	02	Router Object
03	03	DeviceNet Object
04	04	Assembly Object
05	05	Connection Object
08	08	Digital Input Point Object
09	09	Digital Output Point Object
10	0a	Analog Input Point Object
11	0b	Analog Output Point Object
43	2D	Acknowledge Handler Object
48	30	Device Supervisor Object
100	64	Configuration Object
101	65	Inline Interface Object
102	66	Inline Module Object
103	67	Inline Special Function Object
104	68	COS Mask Object
105	69	PCP Object
106	6A	Serial Communications Object (SCO)

B 4 Identity Object (Class Code: 01_{dec}, 01_{hex})

The Identity Object is required on all devices and provides identification of and general information about the device.

B 4.1 Identity Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	10

B 4.2 Identity Object Instance Attributes

Attribute	Access	Name	Type	Value
1	Get	Vendor	UINT	562
2	Get	Product Type	UINT	0 = Generic Device
3	Get	Product Code	UINT	8163
4	Get	Revision	STRUCT OF	
		Major Revision	USINT	1
		Minor Revision	USINT	1
5	Get	Device Status	UINT	
6	Get	Serial Number	UINT	
7	Get	Product Name	STRUCT OF	
		Length	USINT	19
		Name	STRING [6]	IL DN BK DI8 DO4
8	Get	State	USINT	
10	Get/Set	Heartbeat Interval	USINT	Heartbeat interval in seconds

B 4.3 Identity Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
05	05	No	Yes	Reset
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 4.4 Identity Object Attributes

(3): Product Code The product code is fixed at 8163 for the IL DN BK DI8 DO4-PAC. The product code is used within the Electronic Data Sheet format to uniquely identify the product type.

(4): Revision The major revision number will increment as functional enhancements are implemented. The minor firmware revision control number is incremented if minor changes are incorporated.

(5): Device Status

Bit Number	Name	Meaning
0	Owned	= 0, not owned = 1, allocated
1	Reserved	
2	Configured	= 0, not configured – this bit is not supported
3	Reserved	
4-7	User defined	
8	Minor Recoverable fault	= 0, no fault = 1, minor recoverable faults (DOP short circuit)
9	Minor Unrecoverable fault	= 0, no fault = 1, minor unrecoverable faults
8	Major Recoverable fault	= 0, no fault = 1, major recoverable faults (Loss of +24 V DC)
9	Major Unrecoverable fault	= 0, no fault = 1, major unrecoverable faults (Checksum, A/D)
12-15	Reserved	

(6): Serial Number The serial number is encoded in the product during the manufacturing cycle.

(7): Device Name The Device Name provides a character array containing the short string IL DN BK DI8 DO4-PAC.

(8): Device State The Device State reflects whether any errors have occurred and the severity. The following states are supported. The only exit from a Major Unrecoverable fault condition is power cycling the device.

State	Interpretation	Causes
0	Non-existent	
1	Self Test	
2	Standby	
3	Operating	Normal operating mode
4	Major Recoverable fault	Loss of +24 V DC power IBS Uart failure
5	Major Unrecoverable fault	Memory Checksum failure

B 5 Router Object (Class Code: 02_{dec}, 02_{hex})

The Message Router Object provides a messaging connection point through which a client may address a service to any object class or instance residing in the physical device.

B 5.1 Router Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	4

B 5.2 Router Object, Instance 1 Attributes

Router Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	No	Get_Attribute_Single

B 6 DeviceNet Object (Class Code: 03_{dec}, 03_{hex})

The DeviceNet Object defines how the node interfaces to the DeviceNet™ system.

B 6.1 DeviceNet Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	9

B 6.2 DeviceNet Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 6.4)	
1	Get/Set	MAC ID	USINT		(1)
2	Get/Set	Baud Rate	USINT		(2)
3	Get/Set	Bus Off Interrupt	BOOL		(3)
4	Get/Set	Bus Off Counter	USINT		(4)
5	Get/Set	Allocation Information	STRUCT of		(5)
		Choice Byte	BYTE		
		Master Node Addr.	USINT		
6	Get	MAC ID Switch changed	BOOL		
7	Get	Baud Rate Switch Changed	BOOL		
8	Get	MAC ID Switch Value	USINT		
9	Get	Baud Rate Switch Value	USINT		

B 6.3 DeviceNet Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single
75	4B	No	Yes	Allocate Master/Slave
76	4C	No	Yes	Release Master/Slave

B 6.4 Values for DeviceNet Object Attributes

(1): MAC ID

The MAC ID is set using 7 DIP switches. Valid MAC ID addresses are 0 to 63 (0 to $3F_{Hex}$). Setting the switch address to a value greater than 63 will disable the switch and allow software setting of the MAC ID. The software setting defaults to the last hardware setting. The switch is only read during power up.

(2): Baud Rate

The Baud Rate is set using 2 DIP switches located on the module. Setting the switch to a position greater than 2 disables the switch and allows software setting of the Baud Rate. The software setting defaults to the last hardware setting.

(3): Bus Off Interrupt

Bus Off Interrupt (BOI) determines the action if a Bus Off state is encountered.

BOI	Action
0	Hold chip in OFF state (default)
1	If possible reset CAN chip

(4): Bus Off Counter

Bus Off Counter will be forced to 0 whenever set regardless of the data value provided.

(5): Allocation Information

Allocation_byte		
Bit 0	Explicit	Set to 1 to allocate
Bit 1	Polled	Set to 1 to allocate
Bit 2-7	Reserved (always 0)	

B 7 Assembly Object (Class Code: 04_{dec}, 04_{hex})

The Assembly Objects bind attributes of multiple objects to allow data to or from each object to be sent or received over a single connection.

B 7.1 Assembly Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Class ID	UINT	101
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	3

B 7.2 Assembly Object, Instance 100 Attributes

Attribute	Access	Name	Type	Value
3	Get	Data	See Configuration Class	See Configuration Class

B 7.3 Assembly Object, Instance 101 Attributes

Attribute	Access	Name	Type	Value
3	Get/Set	Config Out Data	See Configuration Class	See Configuration Class

B 7.4 Assembly Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 7.5 Assembly Instance 100

Assembly instance 100 is used to generate the I/O Produce Data packet and consists of a variable number of bytes as determined by the configuration object.

B 7.6 Assembly Instance 101

Assembly instance 101 is used to consume the I/O request packet and consists of a variable number of digital output states, analog output values, as well as other Inline Data as determined by the configuration object.

B 8 Connection Object (Class Code: 05_{hex}, 05_{dec})

The Connection Objects manage the characteristics of each communication connection. As a UCMM capable device the unit supports up to three UCMM explicit connections, up to three Dynamic I/O connections, as well as one master/slave Poll, Strobe and COS/Cyclic connections. Up to 10 connections can be established at one time.

B 8.1 Connection Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Class ID	UINT	Max Connection Allocated
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	17

B 8.2 Connection Object, Instance 1 Attributes (Explicit Message)

Attribute	Access	Name	Type	Value (see B 8.8)
1	Get	Connection State	USINT	(1)
2	Get	Instance Type	USINT	0 = Explicit Message
3	Get	Transport Class Trigger	USINT	83_{hex}
4	Get	Production Connection	UINT	(4)
5	Get	Consumed Connection	UINT	(5)
6	Get	Initial Comm. Char.	USINT	21_{hex}
7	Get	Production Size	UINT	30
8	Get	Consumed Size	UINT	35
9	Get/Set	Expected Packet Rate	UINT	Default 2500 ms
12	Get/Set	Timeout Activity	USINT	(12)
13	Get	Prod. Path Length	USINT	0
14	Get	Production Path		
15	Get	Cons. Path Length	USINT	0
16	Get	Consumed Path		

B 8.3 Connection Object, Instance 2 Attributes (POLL connection)

Attribute	Access	Name	Type	Value (see B 8.8)
1	Get	Connection State	USINT	(1)
2	Get	Instance Type	USINT	1 = I/O Message
3	Get	Transport Class Trigger	USINT	83_{hex}
4	Get	Production Connection	UINT	(4)
5	Get	Consumed Connection	UINT	(5)
6	Get	Initial Comm. Char.	USINT	1_{hex}
7	Get	Production Size	UINT	See Configuration Class
8	Get	Consumed Size	UINT	See Configuration Class
9	Get/Set	Expected Packet Rate	UINT	Default 2500 ms
12	Get/Set	Timeout Activity	USINT	(12)
13	Get	Prod. Path Length	USINT	6
14	Get	Production Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	100_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}
15	Get	Cons. Path Length	USINT	6
16	Get	Consumed Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	101_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}

B 8.4 Connection Object, Instance 3 Attributes (STROBE connection)

Attribute	Access	Name	Type	Value (see B 8.8)
1	Get	Connection State	USINT	(1)
2	Get	Instance Type	USINT	1 = I/O Message
3	Get	Transport Class Trigger	USINT	83_{hex}
4	Get	Production Connection	UINT	(4)
5	Get	Consumed Connection	UINT	(5)
6	Get	Initial Comm. Char.	USINT	1_{hex}
7	Get	Production Size	UINT	See Configuration Class
8	Get	Consumed Size	UINT	See Configuration Class
9	Get/Set	Expected Packet Rate	UINT	Default 2500 ms
12	Get/Set	Timeout Activity	USINT	(12)
13	Get	Prod. Path Length	USINT	6
14	Get	Production Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	100_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}
15	Get	Cons. Path Length	USINT	6
16	Get	Consumed Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	101_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}

B 8.5 Connection Object, Instance 4 Attributes (COS/CYCLIC connection)

Attribute	Access	Name	Type	Value (see B 8.8)
1	Get	Connection State	USINT	(1)
2	Get	Instance Type	USINT	1 = I/O Message
3	Get	Transport Class Trigger	USINT	83_{hex}
4	Get	Production Connection	UINT	(4)
5	Get	Consumed Connection	UINT	(5)
6	Get	Initial Comm. Char.	USINT	1_{hex}
7	Get	Production Size	UINT	See Configuration Class
8	Get	Consumed Size	UINT	See Configuration Class
9	Get/Set	Expected Packet Rate	UINT	Default 2500 ms
12	Get/Set	Timeout Activity	USINT	(12)
13	Get	Prod. Path Length	USINT	6
14	Get	Production Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	100_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}
15	Get	Cons. Path Length	USINT	6
16	Get	Consumed Path	STRUCT of	
		Log. Seg., Class	USINT	20_{hex}
		Class Number	USINT	04_{hex}
		Log.Seg., Instance	USINT	24_{hex}
		Instance Number	USINT	101_{hex}
		Log.Seg., Attribute	USINT	30_{hex}
		Attribute Number	USINT	03_{hex}

B 8.6 Connection Object, Instances 5-10 Attributes (UCMM or Dynamic I/O Connections)

See ODVA Specifications

B 8.7 Connection Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
05	05	Yes	Yes	Reset
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 8.8 Connection Object Attributes

(1): Connection State

Connection State	Interpretation
0	Non-existent
1	Configuring
3	Established
4	Timed Out

(4) and (5): Connection ID

Connection 1 Produced Connection ID: 10xxxxxx011

Connection 1 Consumed Connection ID: 10xxxxxx100

Connection 2 Produced Connection ID: 01111xxxxxx

Connection 2 Consumed Connection ID: 10xxxxxx101

xxxxxx = Node Address.

(12): Timeout Action

Timeout Action	
0	Timeout (I/O Messaging default)
1	Auto Delete (Explicit Messaging, fixed value)
2	Auto Reset

B 9 Digital Input Point (DIP) Object (Class Code: 08_{dec}, 08_{hex})

The Digital Input Point (DIP) Object models digital inputs in a product. You can use this object in applications as simple as a toggle switch or as complex as a digital I/O control module. There is a separate instance for each digital input available on the device.

B 9.1 DIP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	(NUMBER OF DIPS)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	101

B 9.2 DIP Object, Instance 1.. (NUMBER OF DIPS) Attributes

Attribute	Access	Name	Type	Value (see B 9.4)	Default
3	Get	Input State	BOOL	0 == OFF, 1 == ON (3)	
4	Get	Input Status	BOOL	0 == okay, 1 == fault (4)	
100	Get/Set	Latch Enable	BOOL	0 = Do not latch, 1 = Latch (100)	0
101	Get/Set	Latch State	BOOL	0 = Latch Low, 1 = Latch High (101)	1

B 9.3 DIP Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 9.4 DIP Object Attributes

(3): Input State

Attribute 3 provides the state of the specific digital input. A value of 0 indicates an OFF state and a value of 1 indicates an ON state. The digital inputs provide feedback of the digital output states. If the corresponding output state is set to 0 these points may be used as inputs.

(4): Input Status

The input status bit indicates if an error has occurred associated with a physical input. If the +24 V DC power is not present the circuitry cannot accurately determine the state of the inputs and will set the Input Status bits of inputs 1 to 24. The status bits are cleared when the +24 V DC power is restored.

(100): Latch Enable

Activates the latching function for the input instance

(101): Latch State

Controls whether the latch is set based on a high (1) or low (0) input value. The latch can be clear by explicit message, or all latches can be cleared simultaneously using the command byte.

B 10 Digital Output Point (DOP) Object (Class Code: 09_{dec}, 09_{hex})

The Digital Output Point (DOP) Object models digital outputs in a product. You can use this object in applications as simple as a actuator or as complex as a digital I/O control module. There is a separate instance for each digital output available on the device.

B 10.1 DOP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	(NUMBER OF DOPS)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	8

B 10.2 DOP Object, Instance 1.. (NUMBER OF DOPS) Attributes

Attribute	Access	Name	Type	Value (see B 10.4)	Default
3	Get/Set	Output State	BOOL	State of Output (3)	0
4	Get/Set	Output Status	BOOL	Status of Output (4)	0
5	Get/Set	Fault State	BOOL	0 = fault value, 1 = no chg (5)	1
6	Get/Set	Fault Value	BOOL	0 = Off, 1 = On (6)	0
7	Get/Set	Idle State	BOOL	0 = Idle value, 1 = no chg (7)	1
8	Get/Set	Idle Value	BOOL	0 = Off, 1 = On (8)	0

B 10.3 DOP Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

(3): Output State

Attribute 3 provides the state of the specific digital output.

(4): Output Status

The output status bit indicates a fault condition.

(5): Fault State

The Fault State determines what action is taken if a software fault condition is detected due to a connection timeout.

Fault State	Action Taken
0	Set the output to the stated determined by the Fault Value
1	Leave the output in the current state

(6): Fault Value

The Fault Value determines the state of the DOP output if the Fault State bit is clear and a fault condition occurs.

(7): Idle State

The Idle State determines what action is taken if an idle condition is detected. Idle conditions occur if a Poll request packet is received with less than the calculated number of bytes. Refer to the Configuration object to determine the size of the Poll Request packets. A poll request of 0 bytes is typically used to force an idle condition.

Idle State	Action Taken
0	Set the output to the stated determined by the Idle Value
1	Leave the output in the current state

(8): Idle Value

The Idle Value is used to set the output if the Idle State bit is clear and an idle condition occurs.

B 11 Analog Input Point (AIP) Object (Class Code: 10_{dec}, $0A_{hex}$)

The IL DN BK DI8 DO4-PAC supports variable analog inputs. There is a separate instance for each digital input available on the device.

B 11.1 AIP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	(NUMBER OF AIPS)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	103

B 11.2 AIP Object, Instance 1.. (NUMBER OF AIPS) Attributes

Attribute	Access	Name	Type	Value (see B 11.4)	Default
3	Get	Value	UINT	0_{hex} ... $FFFF_{hex}$ (3)	
4	Get	Status	BOOLEAN	0 = ok (4)	
7	Get/Set	Range	USINT	(7)	
8	Get	Type	USINT	(8)	6
100	Get/Set	Override Range	BOOLEAN	0 = Use Attr 7 for range, 1 = Use Attr 101 for range	
101	Get/Set	Override Value	UINT	0_{hex} ... $FFFF_{hex}$	0
102	Get/Set	Override Data in I/O	BOOLEAN	1 = Use Override Data in I/O message	0
103	Get/Set	AIP Delta	UINT	Value is used to determine when a COS message is generated.	$00FF_{hex}$

B 11.3 Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0e	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 11.4 AIP Object Attributes

(3): Value

Value of analog input.

(4): Status

If the analog input status bit is set it indicates that a hardware fault has occurred during the previous analog read. The value is left at the last valid value read. A fault during the analog input function results in a Major Unrecoverable Fault condition (see Identity object).

(7): Range

The AIP Range value is used when performing Explicit Message reads to the AIP or during POLLING. The AIP Range values are retained in E2 memory.

Range Value	Description
0	-10 to +10 V
2	0 to +10 V
3	+4 to +20 mA
6	0 to +20mA (non-standard DeviceNet™ value)
7	-20mA to +20mA (non-standard DeviceNet™ value)

(8): Type

The AIP Type value is fixed as type 6 (UINT).

B 12 Analog Output Point (AOP) Object (Class Code: 11_{dec}, $0B_{hex}$)

The IL DN BK DI8 DO4-PAC supports Analog Output Points (AOP). There is a separate instance for each digital output available on the device.

B 12.1 AOP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	(NUMBER OF AOPS)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	100

B 12.2 AOP Object, Instance 1.. (NUMBER OF AOPS) Attributes

Attribute	Access	Name	Type	Value (see B 12.4)		Default
3	Get	Value	UINT	0_{hex} ... $FFFF_{hex}$	(3)	0
7	Get/Set	Output Range	BYTE	1 = (0 to +10)	(7)	1
8	Get	Output Data Type	USINT	6 = UINT	(8)	6
9	Get/Set	Fault State	BYTE	0 ... 3	(9)	0
10	Get/Set	Idle State	BYTE	0 ... 3	(10)	0
11	Get/Set	Fault Value	INT	0_{hex} ... $FFFF_{hex}$	(11)	0
12	Get/Set	Idle Value	INT	0_{hex} ... $FFFF_{hex}$	(12)	0
100	Get	AOP Response Data	UINT	0_{hex} ... $FFFF_{hex}$		

B 12.3 AOP Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 12.4 Values for AOP Object Attributes

(3): Value

The analog output value is given in offset binary format. The value provided must be in the range 0 ... 65535 (0_{hex} ... $FFFF_{hex}$).

Value		Output voltage
dec	hex	
0	0	-10 V
2048	800	0 V
4095	FFF	+10 V

(7): Output Range

The analog output range is defaulted to 1. As the output range is controlled on most modules by which pins are wired to, the value does not actually control the range, it is a method for the network to store its wired configuration.

(8): Output Data Type

The analog output data type is fixed as 6 (UINT).

(9): Fault State

The Fault State determines what action is taken if a fault condition is detected. Fault conditions include software conditions (connection timeout).

Fault State	Action Taken
0	Hold the last value
1	Set to low limit (-10 V DC)
2	Set to high limit (+10 V DC)
3	Set to value determined by Fault Value.

(10): Idle State

The Idle State determines what action is taken if an idle condition is detected. Idle conditions occur if a Poll request packet is received with less than the calculated number of bytes. Refer to the Configuration object to determine the size of the Poll Request packets. A poll request of 0 bytes is typically used to force an idle condition.

Idle State	Action Taken
0	Hold the last value
1	Set to low limit (-10 V DC)
2	Set to high limit (+10 V DC)
3	Set to value determined by Idle Value.

(11): Fault Value

The Fault Value determines the output if the Fault State bit is set to 3 and a fault condition occurs. The value must be in the range 0 ... 65535 (0_{hex} ... $FFFF_{hex}$).

(12): Idle Value

The Idle Value is used to set the output if the Idle State bit is set to 3 and an idle condition occurs. The value must be in the range 0 ... 65535 (0_{hex} ... $FFFF_{hex}$).

B 13 Acknowledge Handler Object (Class Code: 43_{dec}, $2D_{hex}$)

The Acknowledge Handler Object is used to manage the reception of message acknowledgments.

B 13.1 Acknowledge Handler Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	7

B 13.2 Acknowledge Handler Object Instance Attributes

Attribute	Access	Name	Type	Value
1	Get/Set	Acknowledge Timer	UINT	See ODVA specs.
2	Get/Set	Retry Limit	USINT	See ODVA specs.
3	Get/Set	COS Producing Connection Instance	UINT	See ODVA specs.
4	Get	Ack List Size	BYTE	See ODVA specs.
5	Get	Ack List	BYTE, Array of UINT	See ODVA specs.
6	Get	Data with Ack Path List Size	BYTE	See ODVA specs.
7	Get	Data with Ack Path List	BYTE, Array of USINT,UINT	See ODVA specs.

B 13.3 Acknowledge Handler Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 14 Configuration Object (Class Code: 100_{dec}, 64_{hex})

The IL DN BK DI8 DO4-PAC poll request/response packets are large. In some applications it may be desired to reduce the packet size if not all the I/O channels are in use. The configuration object will adjust the poll request/response packet sizes. In addition, the configuration object gives access to several operational parameters such as status info.

B 14.1 Configuration Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	255

B 14.2 Configuration Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 14.4)	Default
3	Get/Set*	Number of Digital Inputs	UINT	(3)	
4	Get/Set*	Number of Digital Outputs	UINT	(4)	
5	Get/Set*	Number of Analog Inputs	UINT	(5)	
6	Get/Set*	Number of Analog Outputs	UINT	(6)	
7	Get/Set*	Add All I/O	BOOL	(7)	0
8	Get/Set*	Accept New Configuration	BOOL	(8)	0
9	Get	Module Change Status	BYTE	(9)	
10	Get/Set	Add All Mode	UINT	(10)	
11	Get/Set*	Use Inline Status	BOOL	(11)	
12	Get/Set *	Include DSUP	BOOL	(12)	
13	Get/Set*	Special Function Modules	BOOL	(13)	
14	Get/Set*	Pad Analog	BOOL	(14)	1
15	Get/Set*	Number of DIP Faults	UINT	(15)	
16	Get/Set*	Number of DOP Faults	UINT	(16)	
17	Get/Set*	Number of AIP Faults	UINT	(17)	
18	Get/Set*	Number of AOP Faults	UINT	(18)	
19	Get/Set *	Number of Special Faults	UINT	(19)	
20	Get	Produce Size	UINT	(20)	
21	Get	Consume Size	UINT	(21)	
22	Get/Set*	Bytes Reserved for DIPS	UINT	(22)	
23	Get/Set*	Bytes Reserved for DOPS	UINT	(23)	
24	Get/Set	Fault Mode	USINT	(24)	

Attribute	Access	Name	Type	Value (see B 14.4)	Default
25	Get/Set	CRC Fault Mode	USINT	(25)	
26	Get/Set	PF Fault Mode	USINT	(26)	
27	Get/Set	Power Fault mode	USINT	(27)	
28	Get/Set	Module Change Fault Mode	USINT	(28)	
29	Get/Set	Configuring Fault Mode	USINT	(29)	
30	Get/Set	Connection Fault Mode	USINT	(30)	
31	Get/Set	Fault Cycles Mode	USINT	(31)	
32	Get/Set*	Inline Control Byte in Poll	BOOL	(32)	
33	Get/Set	I/O Bank Select	USINT	(33)	
34	Get/Set	DNET Power Fault	USINT	(34)	
35	Get/Set*	Bytes reserved for AIPs		(35)	
36	Get/Set*	Bytes reserved for AOPs		(36)	
37	Get	I/O Mapping Revision	UINT	(37)	2
38	Get/Set*	Number of PCP Modules	UINT	(38)	
39	Get/Set*	Number of SCO Modules	UINT	(39)	
40	Get/Set*	Number of PCP Faults	UINT	(40)	
41	Get/Set*	Number of SCO Faults	UINT	(41)	
99	Get/Set	Backward Compatible Mode	BOOL	(99)	0
100	Get	Config In Data	See Below	(100)	
101	Get/Set	Config Out Data	See Below	(101)	
102	Get	Config Strobe Data	See Below	(102)	
200	**Get/ **Set	Digital Only Mode	BOOL	(200)	0
201	**Get/ **Set	Vendor Identifier	USINT	(201)	
254	Get	Actual Revision	UINT	(254)	
255	Get	Main Scans/Second	UINT	(255)	

B 14.3 Configuration Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

*: Changing the configuration object will cause the CONSUMED and PRODUCED size of the POLL connection to be changed. These values are retained in E2 memory and may not be set when any I/O connections is in the Operational state.

**: Attributes are accessible only when password has been written.

B 14.4 Values for Configuration Object Attributes

(3): Number Digital Inputs

The Num Digital Input attribute determines the number of input channels to be returned in the POLL RESPONSE packet. The maximum number 32 bits. The number of poll response bytes can be calculated as:

Number of bytes = ((number of channels) + 7) / 8

(4): Number Digital Outputs

The Num Digital Output attribute determines the number of output bytes to be processed in the POLL REQUEST packet. The number of poll response bytes can be calculated as:

Number of bytes = ((number of channels) + 7) / 8

(5): Number Analog Inputs

The Num Analog Input attribute determines the number of analog input channels returned in the POLL RESPONSE packet. Each analog input produces 2 bytes of data in the poll response packet. The number of bytes may be calculated as:

Number of bytes = ((number of channels) * 2)

(6): Number Analog Outputs

The Num Analog Output attribute determines the number of analog output channels. Each analog output consumes two bytes of data in the POLL REQUEST packet. The number of bytes may be calculated as:

Number of bytes = ((number of channels) * 2)

(7): Add All I/O

The Add All I/O attribute will add all Inline I/O modules to the polled connection.

(8): Accept New Configuration

The Accept New Configuration Attribute will keep the current polled I/O setup even though modules may have been added or deleted. This will clear the I/O module change flag in the status attribute.

(9): Module Change Status

Inline Modules have been changed since last configuration.

(10): Add All Mode

Faults

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
	SCO	PCP	SF	AOPS	AIPS	DOPS	DIPS

I/Os

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
Control	SCO	PCP	SF	AOPS	AIPS	DOPS	DIPS

The Add All Mode Attribute allows the user to select which type of I/Os and Faults will be added when the add all attribute (Attribute 7) is set. Default is $000F_{hex}$ meaning all DIPS, DOPS, AIPS, and AOPS will be added, no faults or special function modules are added.

(11): Use Inline Status

When set the first byte of the poll response contains the Inline Status and the second byte contains the number of the first module in the stack that is faulted. (Adds 2 bytes to produce size.)

(12): Include DSUP When set the first byte of the Poll response contains the device supervisor exception status byte. (Adds 1 byte to produce size.)

(13): Special Function Modules When set the IL DN BK DI8 DO4-PAC will put the Process Data for the special function modules with the data in Poll Attribute set in the Poll Command and Poll Response.

(14): Pad Analog When set this attribute will add an extra byte if necessary to align the analog inputs and outputs to word boundaries. (Will add 0 or 1 byte to consume and/or produce size.)

(15): Number of DIP Faults Selects the number of DIP Faults added to the Poll Response.
(Number of bytes = ((number of channels) + 7) / 8)

(16): Number of DOP Faults Selects the number of DOP Faults added to the Poll Response.
(Number of bytes = ((number of channels) + 7) / 8)

(17): Number of AIP Faults Selects the number of AIP Faults added to the Poll Response.
(Number of bytes = ((number of channels) + 7) / 8)

(18): Number of AOP Faults Selects the number of AOP Faults added to the Poll Response.
(Number of bytes = ((number of channels) + 7) / 8)

(19): Number of Special Function Faults Selects the number of Special Function Faults added to the Poll Response.
(Number of bytes = ((number of channels) + 7) / 8)

(20): Produce Size This attribute allows the user to determine the Produce Size of the device.

(21): Consume Size This attribute allows the user to determine the Consume Size of the device.

(22): Bytes Reserved for DIPS This attribute allows the user to reserve bytes in the polled I/O for future expansion of actual Digital Input Points. The polled I/O contains the number of bytes defined by the greater of either the actual number of DIPS or the number of bytes reserved by this attribute.

(23): Bytes Reserved for DOPS This attribute allows the user to reserve bytes in the polled I/O for future expansion of actual Digital Output Points. The polled I/O contains the number of bytes defined by the greater of either the actual number of DOPS or the number of bytes reserved by this attribute.

(24): Fault Mode This attribute allows the user to select which action is taken during a major failure such as

(25): CRC Fault Mode Determines the affect on the Identity Object State and Status Attributes during a CRC Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(26): PF Fault Mode

Determines the affect on the Identity Object State and Status Attributes during a PF Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(27): Power Fault Mode

Determines the affect on the Identity Object State and Status Attributes during a Power Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(28): Module Change Fault Mode

Determines the affect on the Identity Object State and Status Attributes during a Module Change Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(29): Configuring Fault Mode

Determines the affect on the Identity Object State and Status Attributes during a Configuring Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(30): Connection Fault Mode

Determines the affect on the Identity Object State and Status Attributes during a Connection Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(31): Fault Cycles Mode

Determines the affect on the Identity Object State and Status Attributes during a Fault Cycles Fault.

Value	Effect on Identity Status
0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

(32): Inline Control Byte in Poll

When set to a 1 the first byte of the Poll Command is the Inline Control Byte. (See Inline Object.)

(33): I/O Bank Select

Used when the number of DIPS/DOPS is over 255 per class, not total. 0 = read first 255 instances, 1 = read 256-510 instances. When I/O is read implicitly, this is changed automatically.

(34): DNET Power Fault

Enables the BK to monitor DeviceNet™ power and turn off local outputs if the DeviceNet™ power is failing. This feature is not supported by current hardware.

(35): Bytes reserved for AIPs

This attribute allows the user to reserve bytes in the polled I/O for future expansion of actual Analog Input Points. The polled I/O contains the number of bytes defined by the greater of either the actual number of AIPS or the number of bytes reserved by this attribute.

(36): Bytes reserved for AOPs

This attribute allows the user to reserve bytes in the polled I/O for future expansion of actual Analog Output Points. The polled I/O contains the number of bytes defined by the greater of either the actual number of AOPS or the number of bytes reserved by this attribute.

(37): I/O Mapping Revision

Determines what class Inline modules will be mapped to based on ID code (refer to section "I/O Mapping Revision" on page 3-22).

(38): Number of PCP Modules

Indicates how many PCP capable modules are present on the Inline configuration.

(39): Number of SCO Modules

Indicates how many PCP capable modules, capable of serial communications, are present on the Inline configuration.

(40): Number of PCP Faults

Selects number of PCP fault data bits to add to the DNET I/O PCP Fault Data area. If the number entered is less than the number of PCP modules, the first modules, in order of instance, will receive bits until the bits are filled.

(41): Number of SCO Faults

Selects number of SCO fault data bits to add to the DNET I/O PCP Fault Data area. If the number entered is less than the number of SCO modules, the first modules, in order of instance, will receive bits until the bits are filled.

(99): Backward Compatible Mode

When set to a 1, Objects 100-104 are also accessible through class number 64-68 respectively.

(100): Config In Data	Poll Response: [DSUP Exception Status]* [Inline Status]* [DIP Faults]* [DOP Faults]* [AIP Faults]* [AOP Faults]* [Special Function Faults]* [DIP States]* [Pad byte]* [AIP Values]* [Special Function InData]* [PCP Process Data In][PCP Fragmentation Data In] [SCO Process Data In][SCO Fragmentation Data In] *if enabled or set to >0
(101): Config Out Data	Poll Command: [Inline Control Byte]* [DOP States]* [Pad byte]* [AOP Values]* [Special Function OutData] * [PCP Process Data][PCP Fragmentation Data]* [SCO Process Data][SCO Fragmentation Data]* *if enabled or set to >0
(102): Config Strobe Data	Determines produced data used in strobe connection. Reports the first 8 bytes of the Config In Data (attribute 100).
(200): Digital Only Mode	Allows the BK to only recognize digital IO devices to create a reduced-feature BK if desired.
(201): Vendor Identifier	This value controls what header information is reported back to the network. This attribute becomes necessary when brand labeling is desired. This parameter also controls whether the BK reports as a digital only BK or a full BK.
(254): Actual Revision	This value indicates the internal revision which is incremented each time the project is compiled.
(255): Main Scans/Second	This value indicates the number of times per second the main loop is running. This may be useful for determining processor load under heavy loading conditions.

B 15 Inline Interface Object (Class Code: 101_{dec}, 65_{hex})

The Inline Interface Object allows the user to control and monitor the Inline interface on the IL DN BK DI8 DO4-PAC.

B 15.1 Inline Interface Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	201

B 15.2 Inline Interface Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 15.4)	Default
3	Get/Set	Inline Baud Rate	USINT	(3)	0
4	Get	Inline Status	BYTE	(4)	
5	Get	First Faulted Module	USINT	(5)	
6	Get/Set	Max Retry	USINT	(6)	32
7	Get	Number of Modules	UINT	(7)	
8	Get	Number of Bits	UINT	(8)	
9	Get	Number of Bytes	UINT	(9)	
10	Get/Set	UART Baud Rate	UDINT	(10)	250000
11	Get	Scans Per Second	UINT	(11)	
12	Get	Loop Diagnostic Count	UINT	(12)	
13	Get	Connection Failure Endpoint #1	USINT	(13)	
14	Get	Connection Failure Endpoint #2	USINT	(14)	
15	Get/Set	Latched Inline Status	BYTE	(15)	
16	Get	Latched Faulted Module	USINT	(16)	
17	Get	Latched Connection Failure Endpoint #1	USINT	(17)	
18	Get	Latched Connection Failure Endpoint #2	USINT	(18)	
19	Get	Power Supply Fault	BYTE	(19)	
20	Get/Set	Inline Control Byte	BYTE	(20)	
99	Get	Data Handler Revision	UINT	(99)	

B 15.3 Inline Interface Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
5	05	No	Yes	Reset
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 15.4 Values for Inline Interface Object Attributes

(3): Inline Baud Rate

0 = 500 kbps
1 = 2 Mbps

(4): Inline Status

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Stand Alone Mode	Fault Cycles	Connection	Configuring	Module Change	Power Fault	Peripheral Fault	CRC

(5): First Faulted Module — Contains the number of the first module that is faulted 1 = IL DN BK DI8 DO4-PAC

(6): Max Retry — Sets the number of bad responses the IL DN BK DI8 DO4-PAC will accept before flagging an error. Default = 32

(7): Number of Modules — Displays the number of Inline modules that the IL DN BK DI8 DO4-PAC detected.

(8): Number of Bits — Displays the Process Data size of the Inline modules in Bits.

(9): Number of Bytes — Displays the Process Data size of the Inline modules in Bytes.

(10): UART Baud Rate — Baud rate of the UART interface in bits per second.

(11): Scans Per Second — Displays the number of local I/O scans per second.

(12): Loop Diagnostic Count — Displays the loop diagnostic count during a connection failure.

(13): Connection Failure Endpoint #1 — Displays the number of the module at the first end of a connection failure.

(14): Connection Failure Endpoint #2 — Displays the number of the module at the second end of a connection failure.

(15): Latched Inline Status — Displays the latched value of the Inline Status during the last failure. (See Inline Status Attribute.)

(16): Latched Faulted Module

Contains the number of the first module that was faulted during the last fault
1 = IL DN BK DI8 DO4-PAC

(17): Latched Connection Failure Endpoint #1

Displays the number of the module at the first end of a connection failure latched during the last connection failure.

(18): Latched Connection Failure Endpoint #2

Displays the number of the module at the second end of a connection failure latched during the last connection failure.

(19): Power Supply Fault

Displays the status of the power supplies connected to the IL DN BK DI8 DO4-PAC. A value of 1 indicates a power supply out of range. A value of 0 indicates power supply is ok.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
				US	UM	UL	DNET

(20): Inline Control Byte

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
						Clear DIP Latches	Acknowledge PF

(99): Data Handler Revision

Number signifying the revision of the internal data handler used to transport data to/from DeviceNet™ to Inline modules.

B 16 Inline Module Object (Class Code: 102_{dec}, 66_{hex})

The Inline Module Object allows the user to monitor the Inline modules attached to the IL DN BK DI8 DO4-PAC.

B 16.1 Inline Module Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	(NUMBER OF INLINE MOD)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	10

B 16.2 Inline Module Object, Instance 1.. (NUMBER OF INLINE MODULES) Attributes

Attribute	Access	Name	Type	Value (see B 16.4)
3	Get	Inline Module Id (Config)	UINT	(3)
4	Get	Inline Module Id (Current)	UINT	(4)
5	Get	CIP Class #1	USINT	(5)
6	Get	CIP Class #1 First Instance	UINT	(6)
7	Get	CIP Class #1 Last Instance	UINT	(7)
8	Get	CIP Class #2	USINT	(8)
9	Get	CIP Class #2 First Instance	UINT	(9)
10	Get	CIP Class #2 Last Instance	UINT	(10)

B 16.3 Inline Module Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 16.4 Values for Inline Module Object Attributes

(3):	**Inline Module Id (Config)**	Displays the Inline ID for the module as the unit was configured. Bits 12-8 = Data Length, 7-0 = ID code.
(4):	**Inline Module Id (Current)**	Displays the Inline ID for the module currently.
(5):	**CIP Class#1**	Reflects the number of the CIP Class that the module is mapped to. (i.e. 8= DIP, 9= DOP, etc.)
(6):	**CIP Class#1 First Instance**	Reflects the first instance of the CIP object that this module is mapped to.
(7):	**CIP Class#1 Last Instance**	Reflects the last instance of the CIP object that this module is mapped to.
(8):	**CIP Class #2**	Some Inline modules have two types of I/O points in the same module, for example the DIO module has both digital inputs and digital outputs. Also the AI4 and AO4 modules are mapped to two classes (PCP and AIP or AOP). This attribute reflects the CIP object this module's second I/O type is mapped to.
(9):	**CIP Class #2 First Instance**	Reflects the first instance of the CIP Class #2 object that this module's second I/O type is mapped to.
(10):	**CIP Class #2 Last Instance**	Reflects the last instance of the CIP Class #2 object that this module's second I/O type is mapped to.

B 17 Inline Special Function Object (Class Code: 103_{dec}, 67_{hex})

The Inline Special Function (SF) Object allows the user to control and monitor the Inline modules attached to the IL DN BK DI8 DO4-PAC that do not map to any standard DeviceNet™ Object.

B 17.1 Inline Special Function Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	(NUMBER OF INLINE SF)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	7

B 17.2 Inline Special Function Object, Instance 1.. (NUMBER OF INLINE SF) Attributes

Attribute	Access	Name	Type	Value (see B 17.4)	Default
3	Get	In Data	ARRAY	(3)	
4	Get/Set	Out Data	ARRAY	(4)	
5	Get	Data Size	USINT	(5)	
6	Get	Status	BOOL	(6)	
7	Get/Set	Data In Poll	BOOL	(7)	0

B 17.3 Inline Special Function Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 17.4 Values for Inline Special Function Object Attributes

(3): In Data

Data returned from the Inline module to the IL DN BK DI8 DO4-PAC. Data size is determined by the module.

(4): Out Data

Data sent from the IL DN BK DI8 DO4-PAC to the Inline module. Data size is determined by the module.

(5): Data Size

Number of bytes of Process Data used by the module.

(6): Status

Reflects the status of the special function module.

0 = ok

1 = Faulted.

(7): Data In Poll

When set the, input data is in the Poll Response and the output data is in the Poll Command. This attribute affects the produce and consume sizes of the IL DN BK DI8 DO4-PAC and is therefore only settable when the Poll connection is not in the established state.

B 18 COS Mask Object (Class Code: 104_{dec}, 68_{hex})

The COS Mask Object allows the user control which bits in the Config In Data Attribute cause a COS message to be generated.

B 18.1 COS Mask Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	18

B 18.2 COS Mask Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 18.4)
3	Get/Set	Mask Value	USINT	(3)
4	Get/Set	Index	USINT	(4)
5	Get/Set	Byte Value	BYTE	(5)
6	Get/Set	Add All Mode	WORD	(6)
7	Get/Set	Add All	BOOL	(7)
8	Get/Set	Enable Device Supervisor Exception Status	BOOL	(8)
9	Get/Set	Enable Inline Status	BOOL	(9)
10	Get/Set	Enable DIP Faults	BOOL	(10)
11	Get/Set	Enable DOP Faults	BOOL	(11)
12	Get/Set	Enable AIP Faults	BOOL	(12)
13	Get/Set	Enable AOP Faults	BOOL	(13)
14	Get/Set	Enable Special Function Faults	BOOL	(14)
15	Get/Set	Enable DIPs	BOOL	(15)
16	Get/Set	Enable AIPs	BOOL	(16)
17	Get/Set	Enable Special Function In Data	BOOL	(17)
18	Get/Set	AIP Mask	UINT	(18)

B 18.3 COS Mask Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 18.4 Values for COS Mask Object Attributes

(3): Mask Value

Contains the actual array of bytes that is anded with the Config In Data to determine whether to generate a COS message or not.

(4): Index

Gets or sets the index into the COS Mask Array that the Byte Value Attribute will get or set.

(5): Byte Value

Gets or sets the byte in the COS Mask Array that is indexed by the Index Attribute.

(6): Add All Mode

Automatically generates a COS Mask based on the following bits, when the add all attribute is set to a 1.

Faults

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
	SCO	PCP	SF	AOPS	AIPS	DOPS	DIPS

I/Os

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Control	SCO	PCP	SF	AOPS	AIPS	DOPS	DIPS

(7): Add All

When set to a 1, generates a COS Mask based on the Add All Mode attribute.

(8): Enable Device Supervisor Exception

When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any bits in the Device Supervisor Exception Status byte changes state.

(9): Enable Inline Status

When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any bits in the Inline Status Word changes state.

(10): Enable DIP Fault

When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any DIP Fault changes state.

(11): Enable DOP Fault

When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any DOP Fault changes state.

(12): Enable AIP Fault

When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any AIP Fault changes state.

(13):	**Enable AOP Fault**	When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any AOP Fault changes state.
(14):	**Enable SF Fault**	When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any Special Function Module Fault changes state.
(15):	**Enable DIPs**	When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any DIP value changes state.
(16):	**Enable AIPs**	When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any AIP value changes state.
(17):	**Enable Special Function In Data**	When set to a 1, generates a bit pattern in the COS Mask Array that cause a COS message to be generated when any bit in the Special Function In Data changes state.
(18):	**AIP Mask**	This value is used to generate a bit pattern used to mask each AIP value when the Add All Attribute is set. Example: AIP Mask=$FF00_{hex}$ any time an AIP's upper 9 bits changes, a COS message will be generated.

B 19 PCP Object (Class Code: 105_{dec}, 69_{hex})

The PCP Object allows the user to access I/O modules that support the PCP protocol.

B 19.1 PCP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	0-8 (Number of attached PCP modules)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	23

B 19.2 PCP Object, Instance 1.. (Number of PCP Modules) Attributes

Attribute	Access	Name	Type	Value (see B 19.4)	Default
3	Get	PDU Size	USINT	(3)	
4	Get	PCP Channel Size	USINT	(4)	
5	Get	PCP Module Status	BOOL	(5)	
6	Get/Set	PCP Request	ARRAY of BYTE	(6)	
7	Get	PCP Response	ARRAY of BYTE	(7)	
8	Get/Set	PCP Module	USINT	(8)	
9	Get/Set	PCP Write Index	UINT	(9)	
10	Get/Set	PCP Write SubIndex	USINT	(10)	
11	Get/Set	PCP Write Data	SHORT_STRING	(11)	
12	Get/Set	PCP Read Index	UINT	(12)	
13	Get/Set	PCP Read SubIndex	USINT	(13)	
14	Get	PCP Read Data	SHORT_STRING	(14)	
15	Get/Set	PCP Request Fragment	STRUCT of	(15)	
		Service	BYTE		
		Data	ARRAY of BYTE[7]		
16	Get	PCP Response Fragment	STRUCT of	(16)	
		Service	BYTE		
		Data	ARRAY of BYTE[7]		
17	Get/Set	PCP Fragment in DNET I/O	BOOL	(17)	
18	Get	Process Data Size	USINT	(18)	
19	Get	Process Data In	ARRAY	(19)	
20	Get/Set	Process Data Out	ARRAY	(20)	

Attribute	Access	Name	Type	Value (see B 19.4)	Default
21	Get/Set	Process Data in DNET I/O	BOOL	(21)	1
22	Get/Set	Write Invoke ID	USINT	(22)	0
23	Get/Set	Read Invoke ID	USINT	(23)	0

B 19.3 PCP Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 19.4 Values for PCP Object Attributes

(3): PDU Size

Attribute contains the value of the PDU size used for the PCP channel of the module. Typically 64 bytes. This is the max number of bytes the PCP channel can transfer per request/response.

(4): PCP Channel Size

Attribute contains the value of the PCP channel size. This indicates the number of bytes that are transferred during each Inline scan.

(5): PCP Module Status

Attribute is set to a 1 to indicate an error with the module, a zero indicates the module is ok.

(6): PCP Request

Attribute allows the user to send a request to the PCP module. The response can be read in attribute 4. The format is as follows:

PCP Services:

00_{hex}	NOP
01_{hex}	PCP Read
02_{hex}	PCP Write
03_{hex}	Get PDU Size
09_{hex}	PCP Read w/Invoke ID
$0A_{hex}$	PCP Write w/Invoke ID

PCP Request Format:

Byte 1	Service
Byte 2	Module-number
Byte 3	Index LSB
Byte 4	Index MSB
Byte 5	Subindex
Byte 6	Length
Byte 7 to N	Data block, if necessary

PCP Request w/InvokeID Format:

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Module-number
Byte 4	Index LSB
Byte 5	Index MSB
Byte 6	Subindex
Byte 7	Length
Byte 8 to N	Data block, if necessary

The data length is determined by the PCP Index and PCP Subindex of the object to be read/ written, the max data length is variable and can be up to the PDU size for the module, typically 64 bytes (See PDU length attribute).

(7): PCP Response

Attribute allows the user to get responses from PCP modules. The format is as follows:

Successful Response:

Byte 1	Service
Byte 2	Status
Byte 3	Length
Byte 4 to N	Data block, if necessary

Error Response:

Byte 1	Service
Byte 2	Status
Byte 3	Error Class
Byte 4	Error Code
Byte 5	Additional Error Code 1 LSB, if necessary
Byte 6	Additional Error Code 1 MSB, if necessary
Byte 7	Additional Error Code 2 LSB, if necessary
Byte 8	Additional Error Code 2 MSB, if necessary

Successful Response w / InvokeID:

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Length
Byte 5 to N	Data block, if necessary

Error Response w / InvokeID:

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Error Class
Byte 5	Error Code
Byte 6	Additional Error Code 1 LSB, if necessary
Byte 7	Additional Error Code 1 MSB, if necessary
Byte 8	Additional Error Code 2 LSB, if necessary
Byte 9	Additional Error Code 2 MSB, if necessary

The data length is determined by the PCP Index and PCP Subindex of the object to be read/ written, the max data length is variable and can be up to the PDU size for the module, typically 64 bytes (See PDU length attribute).

(8): PCP Module

Attribute allows the user to set the PCP module number that the reads and writes (Attributes 11 and 14) will access.

(9): PCP Write Index

Attribute sets the Index of the PCP Object Dictionary that will be written when the user writes to the PCP Write Data Attribute (Attribute 11)

- PCP Write SubIndex:
  Attribute sets the SubIndex of the PCP Object Dictionary that will be written when the user writes to the PCP Write Data Attribute (Attribute 11)
- PCP Write Data:
  Attribute allows the user to write data to the PCP Object Dictionary that is referenced by the PCP Module Attribute, PCP Write Index, and PCP Write SubIndex.
- PCP Read Index :
  Attribute sets the Index of the PCP Object Dictionary that will be read when the user reads from the PCP Read Data Attribute (Attribute 14)
- PCP Read SubIndex:
  Attribute sets the SubIndex of the PCP Object Dictionary that will be read when the user reads from the PCP Read Data Attribute (Attribute 14)

(10): PCP Write Subindex

Sets the Subindex of the PCP Object that will be written when the user writes to the PCP Write Data Attribute (Attribute 11).

(11): PCP Write Data

Allows the user to write data to the PCP Object that is referenced by the PCP Module Attribute, PCP Write Index, and PCP Write Subindex. The first byte in the string indicates the number of bytes to be written.

(12): PCP Read Index

Sets the Index of the PCP Object that will be read when the user reads from the PCP Read Data Attribute (Attribute 14).

(13): PCP Read Subindex

Sets the Subindex of the PCP Object that will be read when the user reads from the PCP Read Data Attribute (Attribute 14).

(14): PCP Read Data

Attribute allows the user to read data from the PCP Object Dictionary that is referenced by the PCP Module Attribute, PCP Read Index, and PCP Read SubIndex.

(15): PCP Request Fragment

Attribute allows the user to send requests to the PCP module. This attribute performs the same function as the PCP Request Attribute, except that the request is broken up into packets of 8 bytes each. This allows the user to include the attribute in the Poll without consuming an excess amount of bandwidth or to use DeviceNet™ tools that do not support variable length explicit services.

See Fragmented Service Section above.

(16): PCP Response Fragment

Attribute allows the user to get responses from PCP modules. This attribute performs the same function as the PCP Response Attribute, except that the response is broken up into packets of 8 bytes each. This allows the user to include the attribute in the Poll without consuming an excess amount of bandwidth or to use DeviceNet™ tools that do not support variable length explicit services.

See Fragmented Service Section above.

(17): PCP Fragment in DNET I/O

If set the PCP Response Fragment is added to the Produce data of all the DeviceNet™ I/O connections and the PCP Request Fragment is added to the Poll Consume Data. This attribute affects the produce and the consume sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.

(18): Process Data Size

Number of bytes of Process Data used by the module.

(19): Process Data In

Data returned from the Inline module to the IL DN BK DI8 DO4-PAC. Data size is determined by the I/O module.

(20): Process Data Out

Data sent from the IL DN BK to the Inline module. Data size is determined by the I/O Module.

(21): Process Data in DNET I/O

If set the PCP Process Data In is added to the Produce data of all the DeviceNet™ I/O connections and the PCP Process Data Out is added to the Poll Consume Data. This attribute affects the produce and the consume sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.

(22): Write Invoke ID

Invoke ID used for write service

(23): Read Invoke ID

Invoke ID used for read service

B 20 Serial Communications Object (Class Code: 106_{dec}, $6A_{hex}$)

The Serial Communications Object allows the user to control and transfer Serial Data on RS-232 and RS-485 485/422 modules.

B 20.1 Serial Communications Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	0 ... 8 (Number of attached Serial modules)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	33

B 20.2 Serial Communications Object, Instance 1.. (Number of Serial Modules) Attributes

Attribute	Access	NV	Name	Type	Value (see B 20.4)
3	Get	V	Module Type	USINT	(3)
4	Get	V	Module Status	BOOL	(4)
5	Get	V	Serial Status Word	WORD	(5)
6	Get/Set	V	Serial Control Word	WORD	(6)
7	Get	V	Receive Data	SHORT_STRING	(7)
8	Get/Set	V	Transmit Data	SHORT_STRING	(8)
9	Get	V	Receive Data Fragment	STRUCT of	(9)
			Service	BYTE	
			Data	ARRAY of USINT[7]	
10	Get/Set	V	Transmit Data Fragment	STRUCT of	(10)
				BYTE	
				ARRAY of USINT[7]	
11	Get/Set	NV	Protocol	USINT	(11)
12	Get/Set	NV	Baud Rate	USINT	(12)
13	Get/Set	NV	Data Width	USINT	(13)
16	Get/Set	NV	Error Pattern	USINT	(16)
17	Get/Set	NV	First Delimiter	USINT	(17)
18	Get/Set	NV	Second Delimiter	USINT	(18)
19	Get/Set	NV	3964R Priority	USINT	(19)

Attribute	Access	NV	Name	Type	Value (see B 20.4)
20	Get/Set	NV	Output Type	USINT	(20)
21	Get/Set	NV	DTR Control	USINT	(21)
22	Get/Set	NV	Rotation Switch	USINT	(22)
23	Get/Set	NV	XON Pattern	USINT	(23)
24	Get/Set	NV	XOFF Pattern	USINT	(24)
31	Get/Set	NV	Status/Control in DNET I/O	BOOL	(31)
32	Get/Set	NV	Fragment Data in DNET I/O	BOOL	(32)
33	Get/Set	NV	Serial Enable	BOOL	(33)

B 20.3 Serial Communications Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 20.4 Values for Serial Communications Object Attributes

(3): Module Type This value indicates the module type of the Serial Module.

0 = RS-232

1 = RS-485/RS-422

(4): Module Status This value indicates the module status.

0 = Ok

1 = Faulted

(5): Serial Status Word MSB

15	14	13	12	11	10	9	8
Number of Received Characters (Mode Dependent)							

LSB

7	6	5	4	3	2	1	0
Reserved	Transmit Buffer Not Empty	Transmit Buffer Full	Receive Buffer Full	Re-Init Executed	Send Error	Receive Error	Received Buffer Not Empty

(6) Serial Control Word MSB

15	14	13	12	11	10	9	8
Reserved							

LSB

7	6	5	4	3	2	1	0
DTR	Reserved	Reserved	Reserved	Execute Re-Init	Reset Send Error	Reset Receive Error	Reserved

(7): Receive Data This Attribute allows the user to read the Serial Data in one Get_Attribute_Single. The format is a SHORT_STRING.

(8): Transmit Data This Attribute allows the user to transmit Serial Data in one Set_Attribute_Single. The format is a SHORT_STRING.

(9): Receive Data Fragment This Attribute allows the user to read the Serial Data in pieces. This can be useful for tools that do not support variable length or non-standard length data sizes. It is also used to map the data into the process data. See Fragmented Serial Input Data section above.

(10): Transmit Data Fragment This Attribute allows the user to write the Serial Data to Transmit in pieces. This can be useful for tools that do not support variable length or non-standard length data sizes. It is also used to map the data into the process data. See Fragmented Serial Output Data section above.

(11): Protocol

Code (hex)	Meaning
00	Transparent
01	End-to-end
02	Dual buffer
03	3964R
04	XON/XOFF

(12): Baud Rate

Code (hex)	Value
00	110
01	300
02	600
03	1200
04	1800
05	2400
06	4800
07	9600
08	19200

(13): Data Width

Code (hex)	Meaning		
Data	Bits	Parity	Stop Bits
00	7	Even	1
01	7	Odd	1
02	8	Even	1
03	8	Odd	1
04	8	Without 1	1
05	7	Without 1	1
06	7	Even	2
07	7	Odd	2
08	8	Even	2
09	8	Odd	2
0A	8	Without 2	2
0B	7	Without 2	2

(14) and (15): Reserved

(16): Error Pattern

Code (hex)	Meaning
24	$
xx	Any character

(17): First Delimiter

Code (hex)	Meaning
0D	Carriage return (CR)
xx	Any character

(18): Second Delimiter

Code (hex)	Meaning
0A	Line feed (LF)
xx	Any character

(19): 3964R Priority

Code (hex)	Meaning
00	Low priority
01	High priority

(20): Output Type

Code (hex)	Meaning
00	RS-232 (*Default for RS-232 Module Type)
01	**RS-485** (*Default for RS-485 Module Type)
02	RS-422

(21): DTR Control (Only valid for RS232 Type)

Code (hex)	Meaning
00	Automatic
01	Via process data

(22): Rotation Switch

Code (hex)	Meaning
00	No rotation
01	Rotation

(23): XON Pattern

Code (hex)	Meaning
11	Default
xx	Any character(not the same as XOFF pattern)

(24): XOFF Pattern

Code (hex)	Meaning
13	Default
xx	Any character (not the same as XON pattern)

25 to 30: Reserved

(31):	**Status/Control in DNET I/O**	If set the Serial Status Word is added to the Produce data of all the DNET I/O connections and the Serial Control word is added to the Poll Consume Data. This attribute affects the produce and the consume sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.
(32):	**Fragment Data in DNET I/O**	If set the Receive Data Fragment is added to the Produce data of all the DNET I/O connections and the Transmit Data Fragment is added to the Poll Consume Data. This attribute affects the produce and the consume sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.
(33):	**Serial Enable**	If set the Serial Object is used to send and receive serial Data. Clear this attribute if data transfer is to be accomplished using the PCP object.

C Electronic Data Sheet (EDS) File

This appendix assumes that the user has a working knowledge of DeviceNet™ and RSNetWorx™ configuration software.

The bus coupler supports DeviceNet™ using ODVA standard Digital Input Points (DIP), Digital Output Points (DOP), Analog Input Points (AIP) and Analog Output Points (AOP). Additional objects include user defined Configuration, Inline Interface, Inline Module, Inline Special Function, PCP Special Function and Serial Communications objects.

The EDS (Electronic Data Sheet) file is the software interface between the bus coupler and the RSNetWorx™ configuration software package.

Within the EDS file, parameters are made available to

– configure poll size,
– evaluate the Inline station's status,
– update system configurations, and
– display other informative system characteristics.

Table C-1 lists applicable EDS file parameters, their default values, and a description about their use.

Any configuration changes after the initial power up configuration will require the use of the EDS file, an auto-configuration sequence or an Explicit message to update the configuration. Explicit messages and auto-configuration are covered in section 3, "Configuration and Operation".

Table C-1 EDS file parameters

Parameter No.	Parameter Name	Default Value	Parameterr Description
1	Use Inline Status	1	0: (= False) saved to the coupler disables the 2-byte addition to the poll. 1: (= True) adds 2 bytes of diagnostic data to the polled produced size. Bits are defined in section 4, "Diagnostics".
2	Pad I/O	0	0: (= False) disables this up to 1-byte pad. 1: (= True) saved to the device will cause analog values or special function data to start on an even I/O memory byte. If the pad is not needed it will not be added.
3	Reserve Digital Inputs	0	0: No digital input reservation will be made in the poll. x: Number of digital inputs to be reserved + number of currently used digital inputs.
4	Reserve Digital Outputs	0	0: No digital output reservation will be made in the poll. x: Number of digital outputs to be reserved + number of currently used digital outputs.
5	Reserve Analog Inputs	0	0: No analog input reservation will be made in the poll. x: Number of analog inputs to be reserved + number of currently used analog inputs.
6	Reserve Analog Outputs	0	0: No analog output reservation will be made in the poll. x: Number of analog outputs to be reserved + number of currently used analog outputs.

Table C-1 EDS file parameters (continued)

Parameter No.	Parameter Name	Default Value	Parameterr Description
7	I/O Mapping Revision	1	This parameter controls which DeviceNet™ I/O class Inline modules will be placed into. See section 3, "Configuration and Operation".
8	Add All Mode	127	127_{dec}: allows all digital, analog, special function, PCP, and Serial modules to be added to the poll after an “Add All I/O”. Refer to section B, "DeviceNet™ Object Classes, Message Types and Services", Class Code 100_{dec}, 64_{hex} for additional information.
9	Add All I/O	0	1: (True) saved to the coupler will add all Inline I/O to the polled connection. Automatically returns to 0.
10	Produced Size	0	An upload from the device will display the current Produced Size.
11	Consumed Size	0	An upload from the device will display the current Consumed Size.
12	Number of Digital Inputs	0	An upload from the device will display the current number of digital inputs.
13	Number of Digital Outputs	0	An upload from the device will display the current number of digital outputs.
14	Number of Analog Inputs	0	An upload from the device will display the current number of analog inputs.
15	Number of Analog Outputs	0	An upload from the device will display the current number of analog outputs.
16	Number of DIP Faults	0	X: Number of DIP faults to be added to the poll.
17	Number of DOP Faults	0	X: Number of DOP faults to be added to the poll.
18	Number of AIP Faults	0	X: Number of AIP faults to be added to the poll.
19	Number of AOP Faults	0	X: Number of AOP faults to be added to the poll.
20	Number of Special Faults	0	X: Number of special function faults to be added to the poll.
21	Byte 0, Inline Status Word	0	An upload from the device will display byte 0 of the Inline status word.
22	First Faulted Module	0	An upload from the device will display the first module with a fault. Byte 1 of the Inline status word.
23	Number of Modules	0	An upload displays the number of modules on the Inline backplane.
24	Number of Bits	0	An upload displays the number of bits on the Inline backplane (input bits + output bits).
25	Local Scans per Second	0	An upload will display the number of local scans per second.
26	Fault Mode	0	Controls how the DeviceNet™ interface reacts to a fault. Outputs turn off by default.
27	Loop Diagnostic Count	0	Displays loop diagnostic count during connection failure.
28	Connection Failure Endpoint #1	0	Displays the number of the module at the first end of the failed connection.
29	Connection Failure Endpoint #2	0	Displays the number of the module at the other end of the failed connection.
30	Latched Inline Status	0	Displays the latched error code of the last fault.

Table C-1 EDS file parameters (continued)

Parameter No.	Parameter Name	Default Value	Parameterr Description
31	Latched Faulted Module	0	Displays the module number that had a problem during the last fault.
32	Latched Connection Failure Endpoint #1	0	Displays the number of the module at the first connection failure latched during the last connection error.
33	Latched Connection Failure Endpoint #2	0	Displays the number of the module at the second connection failure latched during the last connection error.
34	Power Supply Status	0	Displays the status of the power supplies connected to the module.
35	CRC Fault Mode	Major	Selects error severity for a CRC fault.
36	PF Fault Mode	Minor	Selects error severity for a Peripheral Fault.
37	Power Fault Mode	Minor	Selects error severity for a Power Fault.
38	Module Change Fault Mode	Major	Selects error severity for a Module Change Fault.
39	Configuring Fault Mode	Major	Selects error severity for a Configuring Fault.
40	Connection Fault Mode	Major	Selects error severity for a Connection Fault.
41	Fault Cycles Fault Mode	Minor	Selects error severity for a Fault Cycles Fault.
42	COS Mask Index	0	Index of COS Mask Byte Value. Must be set to the desired byte number to access the COS Mask byte value.
43	COS Byte Value	0	This value is a mask to determine whether to send a COS message or not. Refer to section B, "DeviceNet™ Object Classes, Message Types and Services", Class Code 104_{dec}, 64_{hex}.
44	COS Add All Mode	16385	Determines what type of COS Mask will be generated during a COS Add All I/O. Refer to section B, "DeviceNet™ Object Classes, Message Types and Services", Class Code 104_{dec}, 68_{hex}.
45	COS Add All	0	Add all inputs defined in the COS “Add All Mode“.
46	COS Enable Inline Status Word	1	Causes a COS message to be generated on an Inline Status Word transition.
47	COS DIP fault	0	Causes a COS message to be generated when a DIP fault changes.
48	COS DOP Fault	0	Causes a COS message to be generated when a DOP fault changes.
49	COS AIP Fault	0	Causes a COS message to be generated when a AIP fault changes.
50	COS AOP Fault	0	Causes a COS message to be generated when a AOP fault changes.
51	COS Special Function Fault	0	Causes a COS message to be generated when a special function fault changes.
52	COS Enable DIP	1	Causes a COS message to be generated when any DIP changes state.
53	COS Enable AIP	0	Causes a COS message to be generated when any AIP changes state.
54	COS Enable Special Function	0	Causes a COS message to be generated when any special function changes state.

Table C-1 EDS file parameters (continued)

Parameter No.	Parameter Name	Default Value	Parameterr Description
55	COS AIP Mask Value	65280	Mask value for analog inputs.
56	Inline Error History 1	0	Most Recent Error Entry
57	Inline Error History 2	0	Second Most Recent Error Entry
58	Inline Error History 3	0	Third Most Recent Error Entry
59	Inline Error History 4		Fourth Most Recent Error Entry
60	Inline Error History 5		Fifth Most Recent Error Entry
61	Inline Error History 6		Sixth Most Recent Error Entry
62	Inline Error History 7		Seventh Most Recent Error Entry
63	Inline Error History 8		Eighth Most Recent Error Entry
64	Inline Error History 9		Ninth Most Recent Error Entry
65	Inline Error History 10		Last Saved Error Entry

D Editing the COS Mask

D 1 General

This appendix describes how to select an input trigger for generating a Change-of-State (COS) network scan.

D 2 Background

The bus coupler receives data in a Change-of-State (COS) type scan. By default any transition of any bit for the Inline Status Word or any digital input point transition will generate a COS scan.

This application note will explain how to mask your produced data to set up the COS trigger on a per bit (event) basis. This information will allow the user to customize their COS trigger event(s) as opposed to allowing any Inline Status or digital input bit to generate a COS state.

D 3 Introduction

The COS mask can be viewed or modified by sending explicit messages to the COS Mask Object (Class Code 104_{dec}, 68_{hex}) or by using the EDS file. If the EDS file is used, information in this application note can be adapted to accomplish the same results.

Figure D-1 shows how the mask bit value coupled with an input transition may or may not generate a COS scan on the DeviceNet™ network. The actual mask bit value is the user selectable enable that determines whether or not the transition will generate the COS trigger.

To set the mask bit value Attributes 4 (COS Mask Index) and 5 (COS Mask Byte Value) will be used in this document.

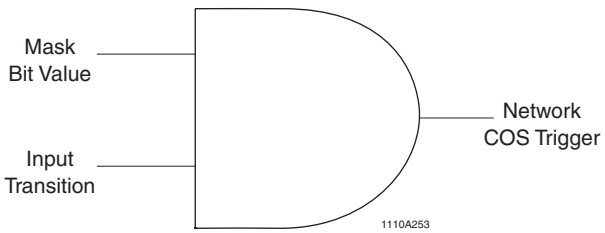


Figure D-1 Logic behind generating a COS event

Figure D-2 shows the example system that will be discussed. This system has 4 bytes of produced data. 2 bytes for the Inline Status word, 1 byte for the onboard I/Os and 1 byte for the (2) DI4's. By default in this example all bits associated to the produced data will generated a COS scan.

This example will show how to set the 3rd bit of the 2nd DI-4 to be the only bit capable of generating a COS scan.

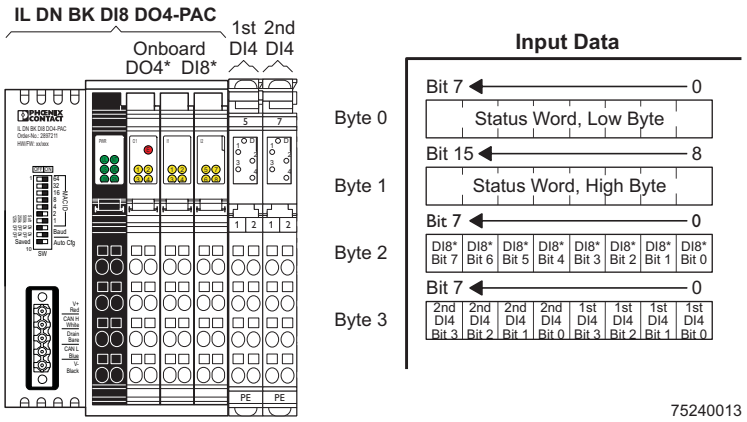


Figure D-2 Example system

D 4 Getting Started

Table D-1 shows the Mask Indexes and Mask Byte Values that would be default for our example system shown in Figure D-2. The Mask Index works like a pointer that can be moved up or down to access the Mask Byte Value data that is located at the specified Mask Index. The following steps need to be taken to program the desired mask values.

Table D-1 COS Mask Index 0, Default COS Mask Byte Value

Mask Index	Mask Byte Value	
	Bin	Hex
0	**1111 1111**	**FF**
1	1111 1111	FF
2	1111 1111	FF
3	1111 1111	FF

Step 1.

By default the Mask Index is pointing towards zero but to ensure this position a Set_ Attribute_Single should be done by send the following explicit message:

Message 1. Set pointer to COS Mask Index 0

Service Code	16 Set_Attribute_Single
Class Code	68_{hex} (COS Mask Object)
Instance	1
Attribute	4 (COS Mask Index)
Data	0 (word)

Message 2. Write 00_{hex} to the COS Mask Byte Value

This message will write 00_{hex} to the first byte of data that is the mask for the low byte of the Inline Status word. 00_{hex} will disable all of these bits from generating a COS scan as shown in Table D-2. This can be done with the following explicit message:

Service Code	16 Set_Attribute_Single
Class Code	68_{hex} (COS Mask Object)
Instance	1
Attribute	5 (COS Mask Byte Value)
Data	0 (word) (This value is FF_{hex} by default)

Table D-2 COS Mask Index 0, New COS Mask Byte Value

Mask Index	Mask Byte Value	
	Bin	Hex
0	**0000 0000**	**00**
1	1111 1111	FF
2	1111 1111	FF
3	1111 1111	FF

Step 2.

Move the pointer to the second index and write 00 as the COS Mask Value by using the following 2 explicit messages. Results are shown in Table D-3.

Message 1. Set pointer to COS Mask Index 1 (being moved from 0).

Service Code	16 Set_Attribute_Single
Class Code	68_{hex} (COS Mask Object)
Instance	1
Attribute	4 (COS Mask Index)
Data	1 (word)

Message 2. Write 00_{hex} to the COS Mask Byte Value (This byte is the mask for the upper byte of the Inline Status word.).

Service Code	16 Set_Attribute_Single
Class Code	68_{hex} (COS Mask Object)
Instance	1
Attribute	5 (COS Mask Byte Value)
Data	0 (word) (This value is FF_{hex} by default)

Table D-3 COS Mask Index 1, New Mask Byte Value

Mask Index	Mask Byte Value	
	Bin	Hex
0	0000 0000	00
1	**0000 0000**	**00**
2	1111 1111	FF
3	1111 1111	FF

Step 3.

Move the pointer to the second index and write 40_{hex} as the COS Mask Value by using the following 2 explicit messages. Results are shown in Table D-4. The 40_{hex} will enable the 3rd bit of the second DI4 to be the only bit enabled to trigger a COS scan on the DeviceNet™ network.

Message 1. Set pointer to COS Mask Index 2 (being moved from 1).

Service Code	16 Set_Attribute_Single
Class Code	68_{hex} (COS Mask Object)
Instance	1
Attribute	4 (COS Mask Index)
Data	2 (word)

Message 2. Write 40_{hex} to the COS Mask Byte Value (This byte is the mask for the 3rd byte of produced data occupied by the 1st and 2nd DI-4 modules.).

Service Code	16 Set_Attribute_Single
Class Code	0x68 (COS Mask Object)
Instance	1
Attribute	5 (COS Mask Byte Value)
Data	40_{hex} (word) (This value is FF_{hex} by default)

Table D-4 COS Mask Index 2, New COS Mask Byte Value

Mask Index	Mask Byte Value	
	Bin	Hex
0	0000 0000	00
1	0000 0000	00
2	**0100 0000**	**40**

SCATTERGOOD & JOHNSON LTD
ELECTRICAL ENGINEERING & FLUID CONTROL DISTRIBUTORS
Est.1899

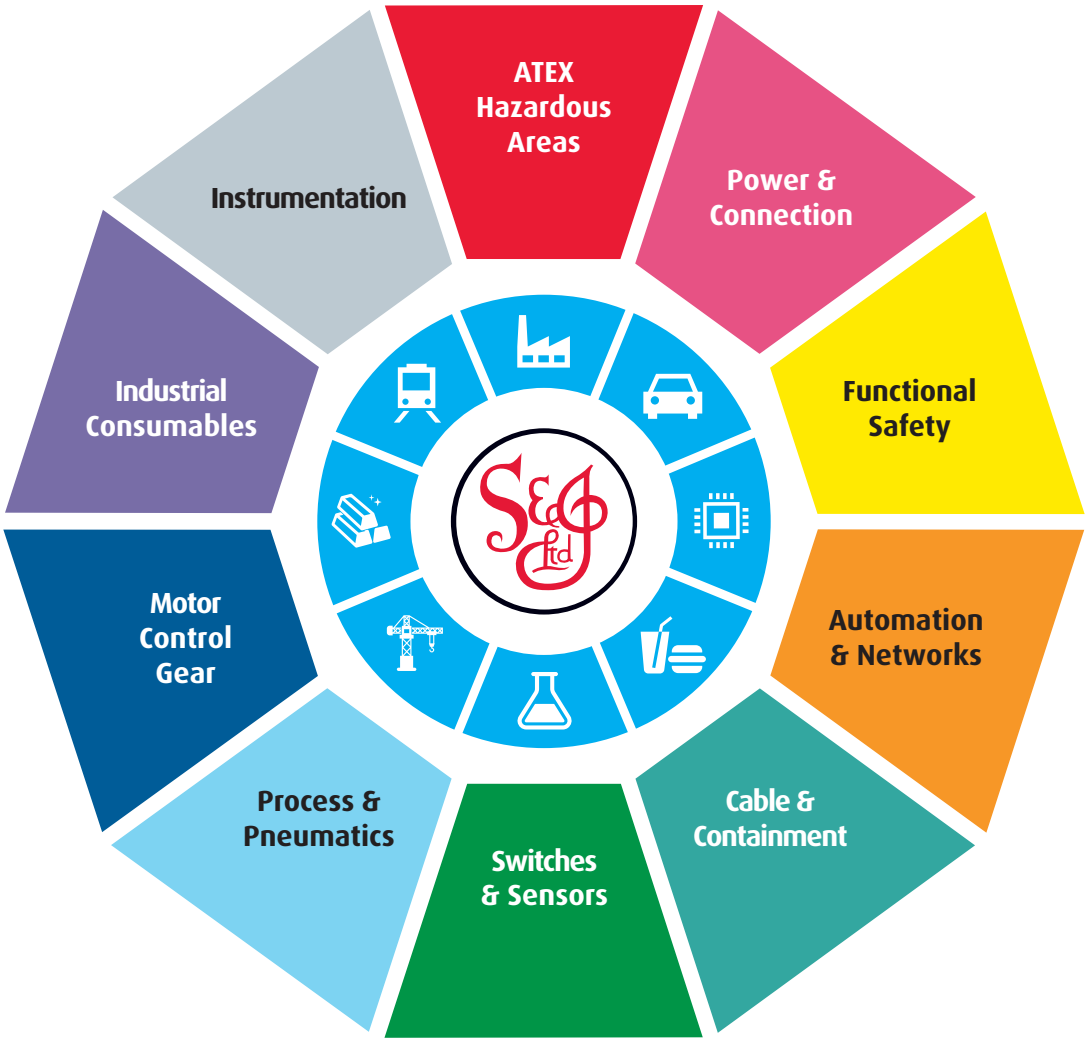
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

Working with a range of quality product suppliers across a number of specialist markets, we are not your average 'box shifter' - we are your technical and supply chain partner.

We fully support every product we sell - for free! Our internal team and external sales engineers can answer any product or application question, no matter the complexity.

Backing up this technical ability is a range of 50,000+ products available from stock for nationwide next day delivery (same day if required!), or you can collect what you need from any of our trade counters around the UK.

Select your specialist interest below to learn more about how we can help.



Online, In Branch and On the Road - Scattergood & Johnson Ltd, there when you need us.

www.scatts.co.uk