Inline bus coupler for EtherNet/IP™ with eight digital inputs and four digital outputs

User manual

PHOENIX CONTACT
INSPIRING INNOVATIONS

User manual

Inline bus coupler for EtherNet/IP™ with eight digital inputs and four digital outputs

UM EN IL EIP BK DI8 DO4-2TX-PAC, Revision 03 12/2017

This user manual is valid for:

Designation	Order No.
IL EIP BK DI8 DO4 2TX-PAC	2897758

7538_en_03

PHOENIX CONTACT GmbH & Co. KG • Flachsmarktstraße 8 • 32825 Blomberg • Germany
phoenixcontact.com

Table of contents

1 For your safety

Read this user manual carefully and keep it for future reference.

1.1 Labeling of warning notes

This symbol indicates hazards that could lead to personal injury.
There are three signal words indicating the severity of a potential injury.

DANGER
Indicates a hazard with a high risk level. If this hazardous situation is not avoided, it will result in death or serious injury.

WARNING
Indicates a hazard with a medium risk level. If this hazardous situation is not avoided, it could result in death or serious injury.

CAUTION
Indicates a hazard with a low risk level. If this hazardous situation is not avoided, it could result in minor or moderate injury.

This symbol together with the **NOTE** signal word alerts the reader to a situation which may cause damage or malfunction to the device, hardware/software, or surrounding property.

Here you will find additional information or detailed sources of information.

1.2 Qualification of users

The use of products described in this user manual is oriented exclusively to:

- Qualified electricians or persons instructed by them. The users must be familiar with the relevant safety concepts of automation technology as well as applicable standards and other regulations.
- Qualified application programmers and software engineers. The users must be familiar with the relevant safety concepts of automation technology as well as applicable standards and other regulations.

2 The Inline EtherNet/IP™ Bus Coupler

EtherNet/IP™ Bus Coupler

The EtherNet/IP™ bus coupler, shown in Figure 2-1, provides an interface between EtherNet/IP™ 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.

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.

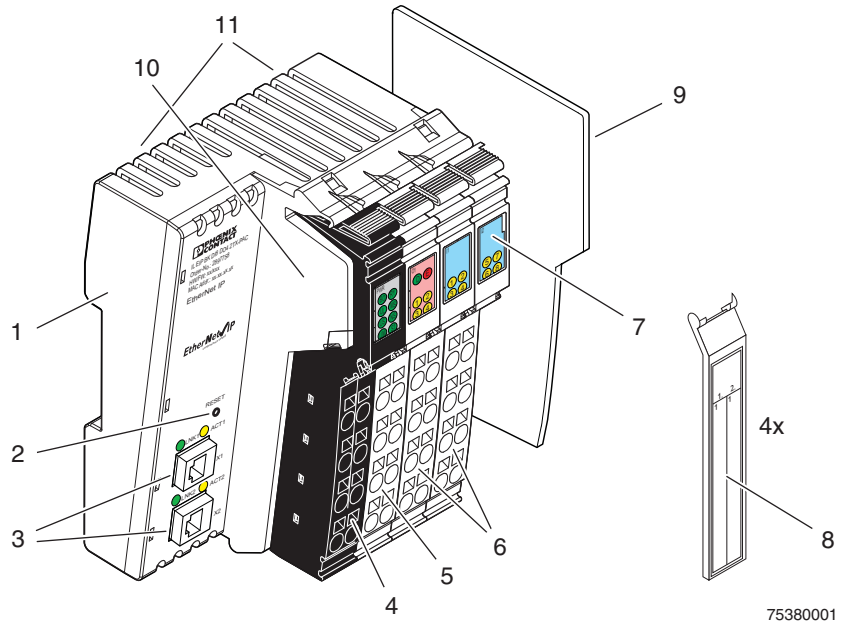


Figure 2-1 Features of the Inline EtherNet/IP™ bus coupler

Key:

1 Electronics base
2 Reset button
3 EtherNet/IP™ connections (twisted pair cables in RJ45 format)
4 Power connector
5 Connector for digital outputs
6 Connector for digital inputs
7 Diagnostic and status indicators
8 Labeling field
9 End plate
10 MAC address in clear text and as a barcode
11 Two FE contacts for grounding the bus coupler using a DIN rail (on the back of the module)

The Electronic Data Sheet (EDS) can be found on the Internet at: phoenixcontact.net/products.

Features

The key features of the IL EIP BK DI8 DO4 2TX-PAC are listed below:

– Module Features

- 2 x Ethernet twisted pair according to 802.3 with auto negotiation and auto crossover connected via an integrated 3-port switch (2 external ports, 1 internal port)
- Ethernet connection via 8-pos. RJ45 female connector
- Ethernet TCP/IP, with 10/100 Base-T (X)
- Industrial Ethernet/IP, Version 1.2
- Process data access via XML
- Web based management (WBM)
- IP parameter configuration:
  BootP, WBM, Static IP (DHCP to be added on later revision)
- Integrated web server
- Eight digital inputs
- Four digital outputs
- Diagnostic and status indicators

– Inline Features

- Up to 61 other Inline terminals can be connected (process data channel)
- Up to eight other PCP modules can be connected
- Can be installed in the field, software for automatic configuration of the station is not required
- Automatic baud rate detection on the local bus (500 kbaud or 2 Mbaud)

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: phoenixcontact.net/products.

– Ethernet and CIP Features

- Electrical isolation of Ethernet interface and logic
- Type of device profile: $0C_{hex}$ communication adapter
- Supported CIP connections in total:
  128 (eight, typical)
- Explicit signaling:
  Max. number of connections 128 (eight, typical)
- I/O signaling:
  Max. number of connections 128 (eight, typical)
- Device configuration possibilities:
  EDS, individual software
- MAC parameter configuration:
  Rate: 10 Mbps, 100 Mbps, automatic
  Duplex: half, full, automatic

Applications

- Connection of sensors/actuators via Ethernet/IP

3 Installation

3.1 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 3-1, shown with the EtherNet/IP™ 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.

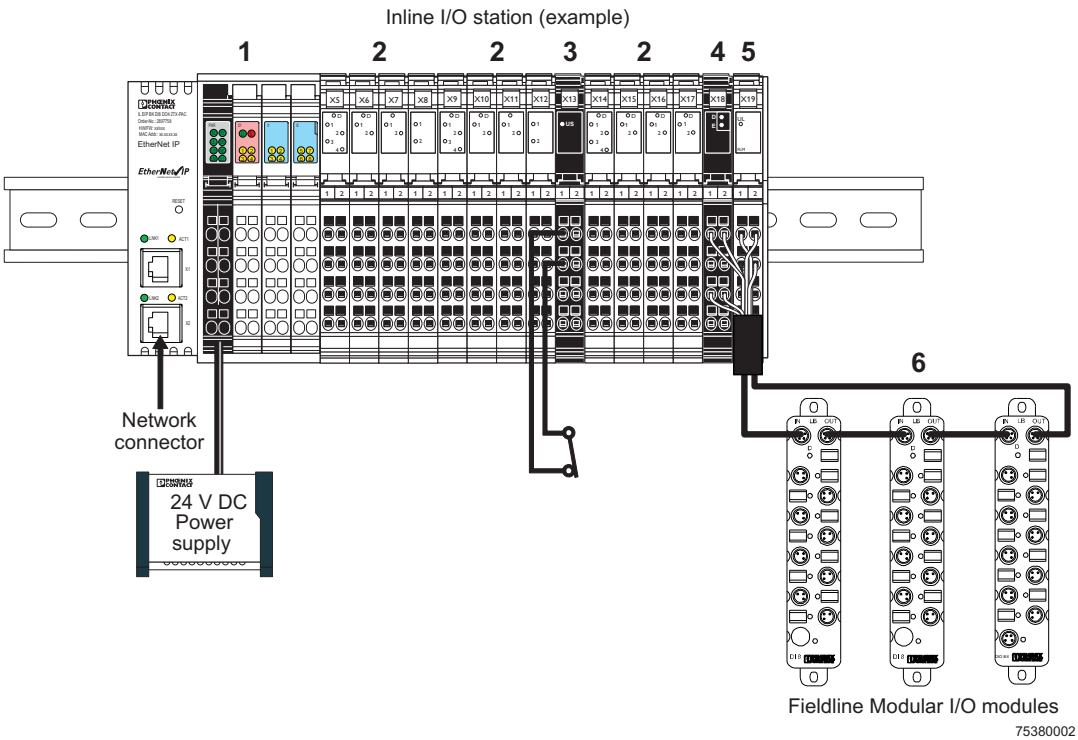


Figure 3-1 Example of a basic Inline station

Key:

1 Bus coupler (here: IL EIP BK DI8 DO4 2TX-PAC)

2 I/O 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

Bus Coupler Module

The first step in setting up a modular I/O station is to connect the bus coupler module to the EtherNet/IP™ 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 EtherNet/IP™ 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 EtherNet/IP™

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 61 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 IL SYS INST UM E user manual.

Observe the current consumption of each device for a given power supply.
Current consumption specifications can be found in the product-specific data sheets.

3.2 Bus Coupler Wiring

3.2.1 Connecting the EtherNet/IP™ System

Connect the EtherNet/IP™ system to the bus coupler (see Figure 3-2) via an RJ45 connector. For the pin assignment, please refer to Table 3-1.

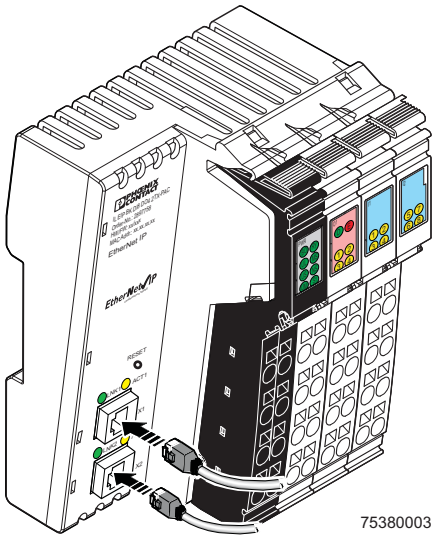
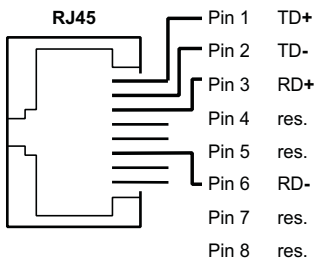


Figure 3-2 Connecting the RJ45 connector

Table 3-1 Pin assignment of the RJ45 connector

Pin	Assignment
1	TxD + (transmit data +)
2	TxD - (transmit data -)
3	RxD + (receive data +)
4	Reserved
5	Reserved
6	RxD - (receive data -)
7	Reserved
8	Reserved



Both Ethernet interfaces have an auto crossover function.

3.2.2 Connection of Supply, Actuators, Sensors

Figure 3-3 and Table 3-2 to Table 3-5 list terminal assignments for the bus coupler connectors. Figure 3-4 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 EtherNet/IP™ 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.

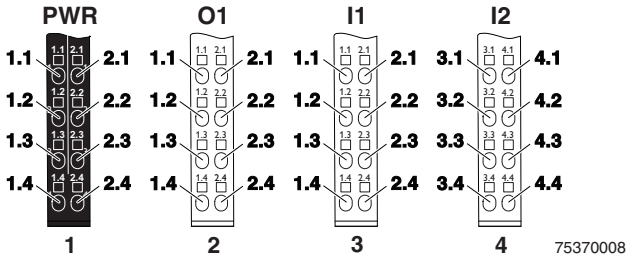


Figure 3-3 Terminal point assignment of the Inline connector

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

Terminal Point	Assignment	Terminal Point	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)

NOTE: Module is damaged in the event of overload

The GND U_M, U_S potential jumper carries the total current of the main and segment circuits. The total current must not exceed the maximum current carrying capacity of the potential jumpers (8 A). If, in the course of configuring, it is found that the 8 A limit is reached at one of the potential jumpers U_S, U_M and GND, a new power terminal must be used.

The functional earth ground must be connected to the 24 V DC supply/functional earth ground connection.

Table 3-3 Terminal point assignment of the output connector (2, O1)

Terminal Point	Assignment	Terminal Point	Assignment
1.1	OUT1	2.1	OUT2
1.2	GND	2.2	GND
1.3	FE	2.3	FE
1.4	OUT3	2.4	OUT4

Table 3-4 Terminal point assignment of the input connector (3, I1)

Terminal Point	Assignment	Terminal Point	Assignment
1.1	IN1	2.1	IN2
1.2	U_S	2.2	U_S
1.3	GND	2.3	GND
1.4	IN3	2.4	IN4

Table 3-5 Terminal point assignment of the input connector (4, I2)

Terminal Point	Assignment	Terminal Point	Assignment
3.1	IN5	4.1	IN6
3.2	U_S	4.2	U_S
3.3	GND	4.3	GND
3.4	IN7	4.4	IN8

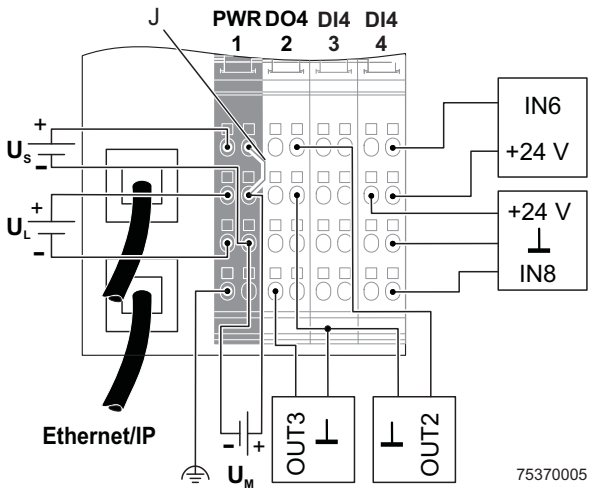


Figure 3-4 Connection example

J = internal jumper (in the module)

4 Startup/Operation

4.1 Default Upon Delivery/Default Settings

Upon delivery, the following functions and features are available:

- IP Configuration
  - IP Address: 0.0.0.0
  - Subnet Mask: 0.0.0.0
  - Default Gateway: 0.0.0.0
  - BootP Requests: Enable
- Software Update
  - Software Update on Next Reboot: Disable
  - TFTP-Server IP Address: 0.0.0.0
  - Downloadable File Name: c2897758.fw
- System Identification
  - Name of Device: IL EIP BK DI8 DO4 2TX-PAC
  - Description: Ethernet/IP Bus Terminal
  - Physical Location: Unknown
  - Contact: Unknown
- Connection Monitoring
- Plug and Play Mode: Enable inputs & outputs

4.2 Starting the Firmware

Upon delivery, the IL EIP BK DI8 DO4 2TX-PAC bus coupler has no valid IP parameters.

Once you have connected the power to the bus coupler, the firmware is started. The following sequence appears on the LEDs:

Table 4-1 LEDs during the starting sequence

Display	Meaning
BO flashing	Starting Boot loader Transmitting BootP requests
BO on	Extracting firmware
BO off	Starting firmware
RY flashing	Firmware ready to operate

4.3 Sending BootP Requests

BootP Enabled

During start-up the device sends BootP requests without interruption until it receives a valid IP address. The requests are transmitted at varying intervals (2 s, 4 s, 8 s, 2 s, 4 s, etc.) so the network is not loaded unnecessarily. If valid IP parameters are received the device will use these parameters until the device is reset or power is cycled.

BootP Disabled

With a valid IP configuration saved to the device and BootP disabled the device will start with the stored IP configuration and will not send BootP requests.

NOTE: Before disabling automatic BootP setting, be sure to record the current IP address. You will need the current IP address if you want to re-enable BootP setting of the IP address. If you forget the IP address, the only way is to delete the whole configuration with the reset button during power up.

4.4 Web-based Management (WBM)

The IL EIP BK DI8 DO4 2TX-PAC has a web server, which generates the required pages for web-based management and, depending on the requirements of the user, sends them to the "Factory Manager" or a standard web browser.

Web-based management can be used to access static information (e.g., technical data, MAC address) or dynamic information (e.g., IP address, status information) or to change the configuration (password-protected).

Calling web-based management

The IL EIP BK DI8 DO4 2TX-PAC web server can be addressed using the IP address if configured correspondingly. The bus coupler homepage is accessed by entering the URL "http://<*IP address*>".

Example: http://172.16.113.38

If you cannot access the WBM pages, check the connection settings in your browser and deactivate the proxy, if set.

Figure 4-1 WBM homepage

4.5 Structure of the Web Pages

The web pages for the Ethernet/IP bus coupler are divided into two sections. The left-hand side has the selection menu with the relevant submenus. The right-hand side displays the information related to the menu item. Static and dynamic information about the bus coupler can be found in the following menus.

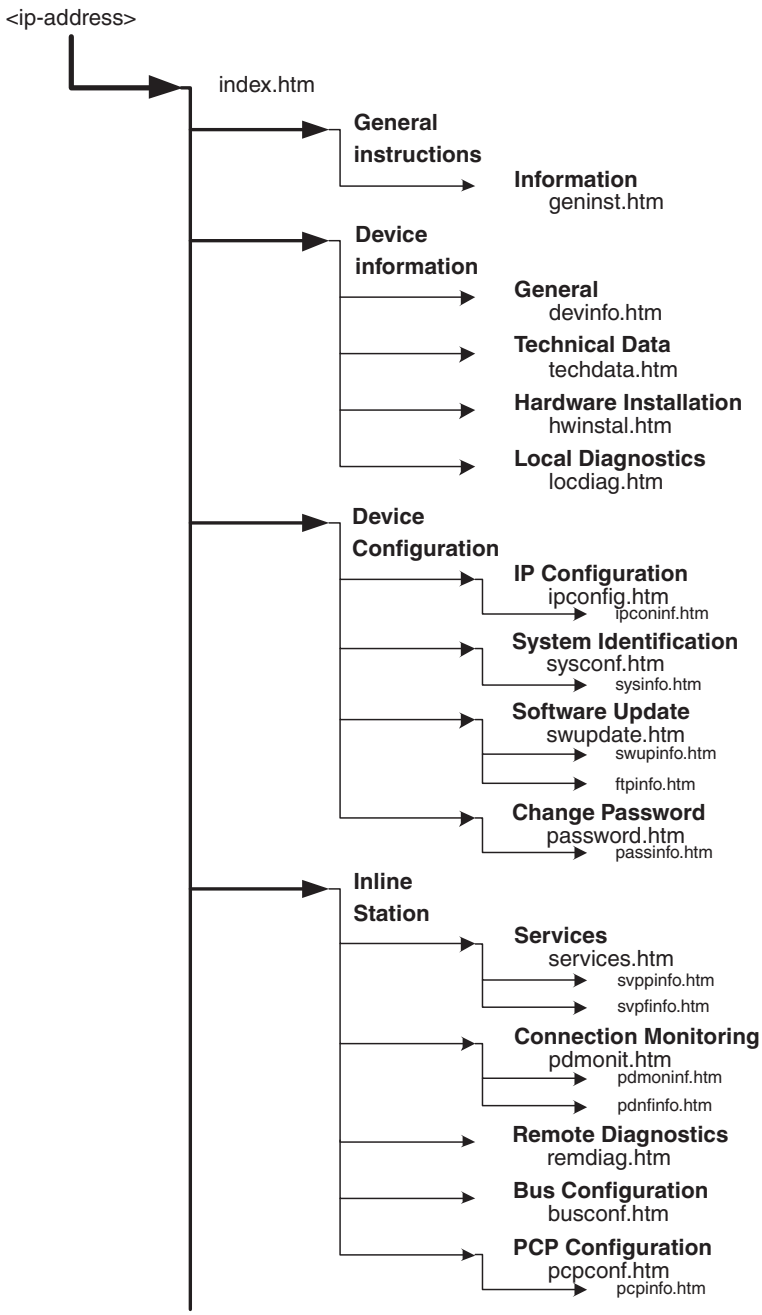


Figure 4-2 Structure of the web pages (example)

4.5.1 Diagnostics

On this website example, you can have a look at the local bus diagnostics. The diagnostic status and diagnostic parameter register are shown here.

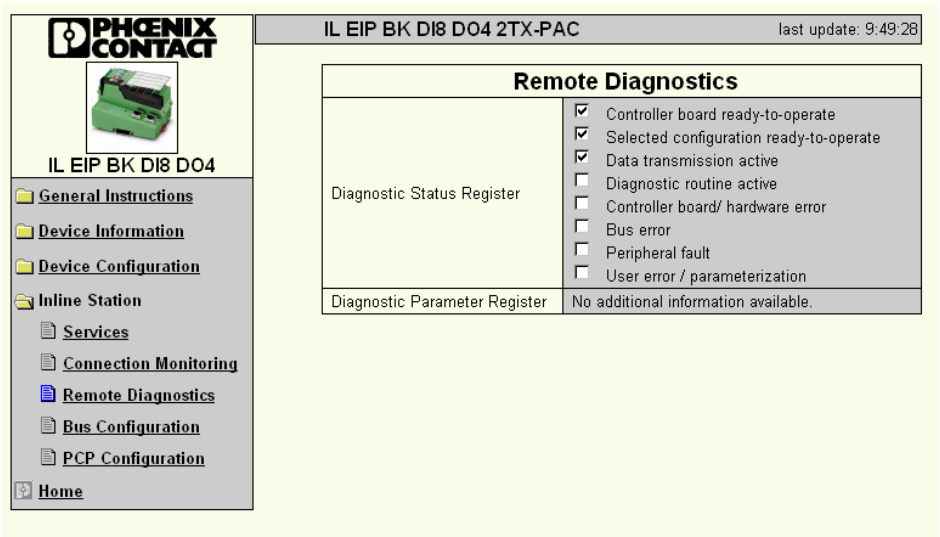


Figure 4-3 Remote diagnostics in WBM

4.5.2 Services

In this example website the user can set or reset the Plug & Play Modes. The user can acquire and store the currently connected I/O configuration. Also the user can confirm any I/O station peripheral faults.

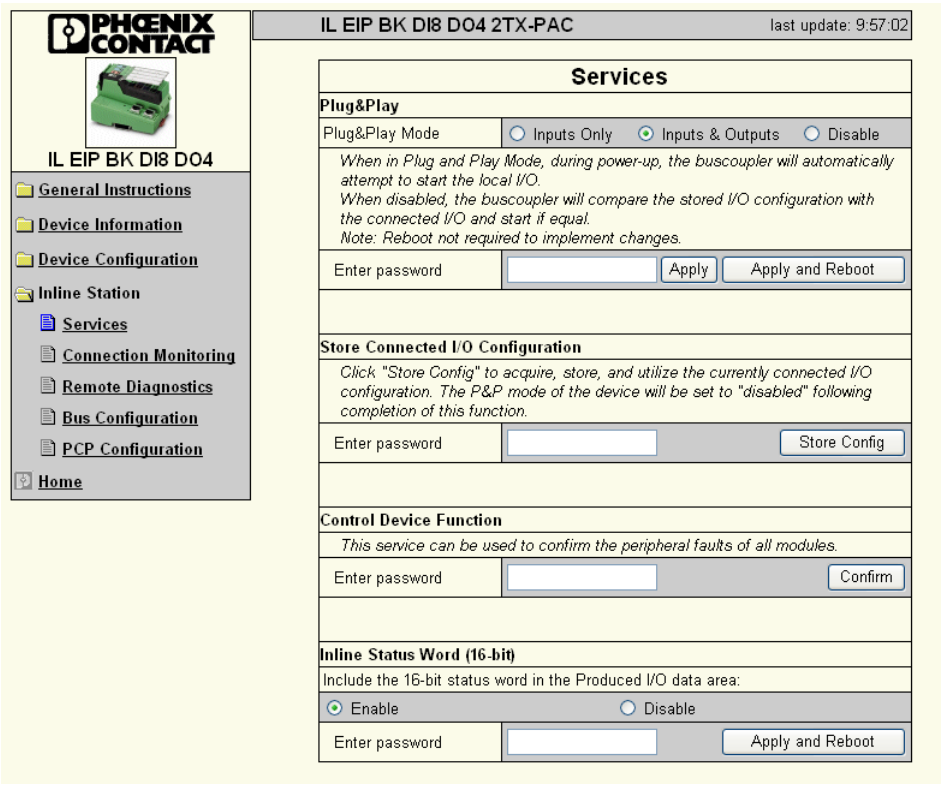


Figure 4-4 Website of the services

4.5.3 Bus Configuration

This example website shows the current configuration of the Inline station including the I/O produced and consumed sizes and local bus baud rate. If an error occurs, the error location is indicated. The number of the affected element is highlighted in red.

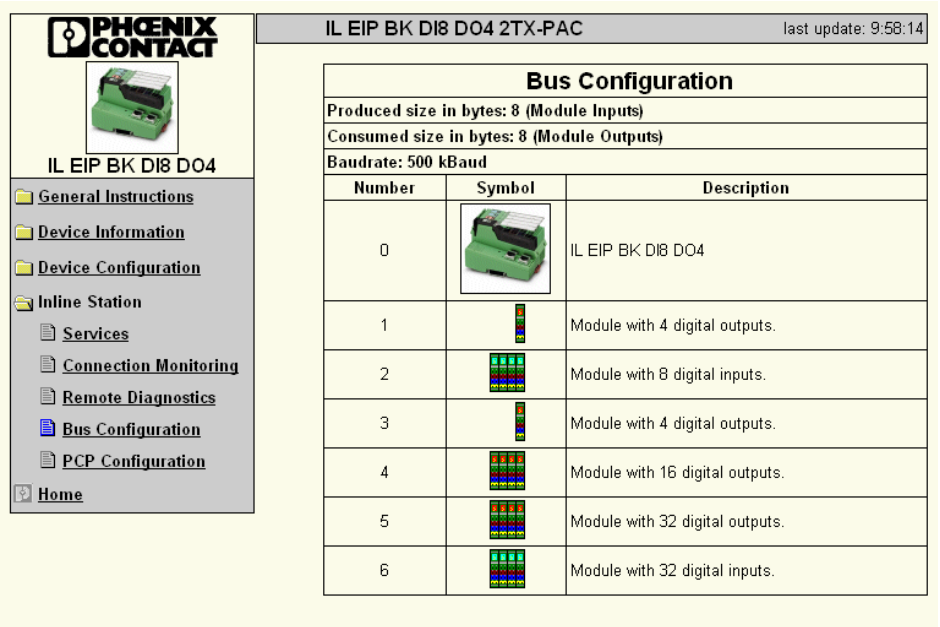


Figure 4-5 Bus configuration of the station

4.5.4 Connection Monitoring (Process Data Monitoring)

Monitoring the Ethernet/IP connections is executed by the Ethernet/IP protocol stack with the help of connection timers. If a connection is aborted, due to a broken cable or a control system without connection reset, the outputs are set back to the pre-defined state. The state is set either in the DOP (digital output point) or in the AOP (analog output point) objects. In order to regain control over the outputs, the error must be removed.

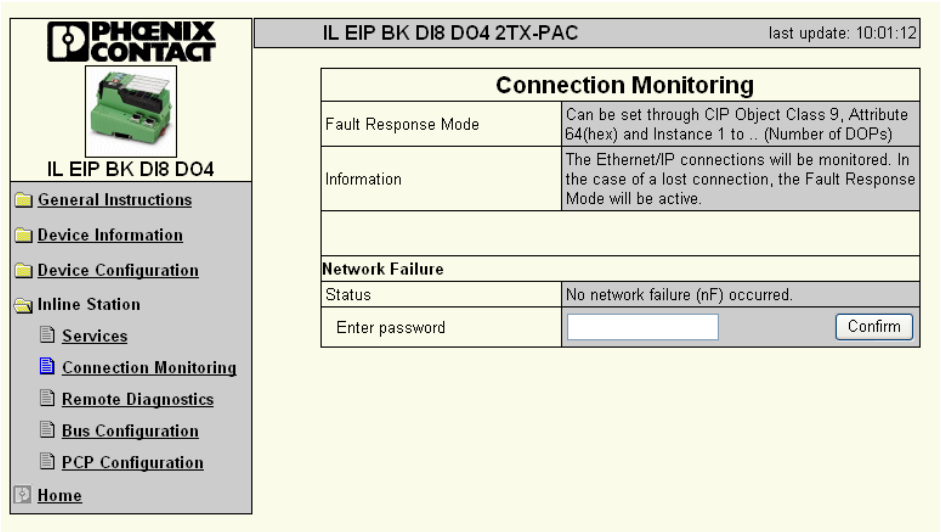


Figure 4-6 Connection monitoring (Process data monitoring)

The point "Inline Station, Connection Monitoring" was called "Inline Station, Process Data Monitoring" in earlier versions.

Please note that attributes are specified as decimal numbers in the object descriptions ($64_{hex} = 100_{dec}$).

4.5.5 Password Protection

All status changes to the bus coupler require the entry of a password. The password can be changed at any time. Your unique password must be between four and twelve characters long (please note that it is case-sensitive). By default upon delivery, the password is "private".

If you forget the password, the only way to access the bus coupler again is to reset the entire configuration using the reset button.

4.5.6 Ethernet Speed and Configuration

The IL EIP BK DI8 DO4 2TX-PAC firmware will include a website setting that allows for the selection of Fixed Manual override or auto-negotiate of transmission speeds. Auto Negotiation plus Manual Override provides support for the widest possible range of network infrastructure devices, such as switches. This will allow the BK to adapt to virtually all possible system applications, including older systems that do not support the higher/auto data rates. By providing the manual over-ride, the system is freed of any auto-negotiate traffic that could possibly impact an older system. However, by default, the module will be in the "auto- negotiate detect" mode of data transmission to allow for easiest connection to current systems.

4.5.7 Firmware Update via WBM and TFTP

The following steps must be carried out when executing a firmware update using WBM:

– In WBM, click on "Device Configuration" and then "Software Update". Enter the IP address of the TFTP server in the "TFTP Server IP Address" field. Then enter the file name of the firmware and the path name, if necessary, in "Downloadable File Name". In the "Software Update on Next Reboot" field, click "Enable".
– Enter your password. To wait until later to apply the update with a restart, click "Apply". To start the update immediately, click "Apply and Reboot".
– Check the execution of the update by checking the firmware version under "Device Information/General". In the event of an error during the download, a restart repeats the download. To abort the update, set "Disable" in the "Software Update on Next Reboot" field.

The firmware version 2.0 may only be used from hardware revision 03.

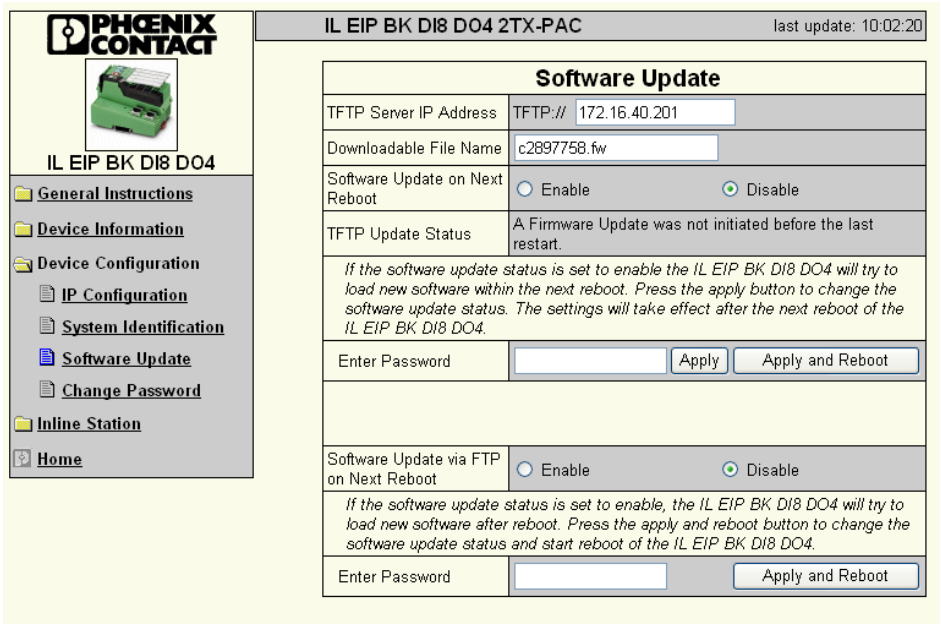


Figure 4-7 "Software Update" menu

If BootP is set to "Enable" and a reply with values for "TFTP Server IP Address" and "Downloadable File Name" is received on a restart, these values overwrite the entries made in WBM. The received values are displayed in WBM after the restart.

In the event of an error during Flash programming (e.g., voltage interrupt), the bus coupler can only be restarted by repeating the update. The bus coupler starts the update automatically after a restart. Access to WBM is no longer possible.

4.5.8 Process Data Access via XML

The integrated web server of the IL EIP BK DI8 DO4 2TX-PAC offers the option of accessing the process data of the connected Inline terminals via a web page in XML format.

You can access the web pages via a standard web browser. To access the XML pages with the process data in the address line of the browser, enter the address in the following format:

http://<IP address>/procdata.xml

4.5.9 XML File Structure

The XML file contains different data areas:

IL_STATION

Frames for the entire XML file. The mandatory elements of this frame are IL_BUS_TERMINAL and IL_BUS.

IL_BUS_TERMINAL

This data area contains information on the entire Inline station (bus coupler and all connected terminals). This data area includes:

TERMINAL_TYPE

This area contains the name of the bus coupler, which is always IL EIP BK DI8 DO4 2TX-PAC.

NAME

Contains the user-specific station name. The station name can be modified via WBM.

IP_ADDRESS

Contains the IP address of the station.

MODULE_NUMBER

Contains the number of connected Inline terminals, including local I/Os. In the event of a bus error, the number of the last known operable configuration is indicated.

DIAGNOSTIC_STATUS_REGISTER

Contains the status of the Inline station, represented by all bits of the diagnostic status register. A detailed description can be found in the diagnostic parameter register. Whenever an error bit is set, the diagnostic parameter register is rewritten.

IL_BUS

Frame for the connected Inline terminals.

IL_MODULE

Frame for the data of an individual Inline terminal. The terminals are numbered consecutively from one to a maximum of 63.

MODULE_TYPE

Contains the terminal type. Possible types are DI, DO, DIO, AI, AO, AIO, and PCP.

PD_CHANNELS

Number of process data channels in an Inline terminal. For digital terminals the number of channels is equal to the number of supported bits. For other terminals, the number of process data words is indicated.
Example: An IB IL AO 2 has two process data channels and an IB IL 24 DO 8 has eight bits and therefore eight process data channels.

PD_WORDS

Number of process data words in an Inline terminal. Please note that analog terminals always have the same number of output and input words. An IB IL AO 2 therefore also has two input words and an IB IL AI 2 also has two output words.

PD_IN

This area is used by all terminals that occupy input data. The number of process data words depends on the terminal type.

PD_OUT

This area is used by all terminals that occupy output data (see also “PD_OUT” on page 29).

Examples:

a) Inline terminal with two active inputs

```
<IL_MODULE number="1">
    <MODULE_TYPE>DI</MODULE_TYPE>
    <PD_CHANNELS>2</PD_CHANNELS>
    <PD_WORDS>1</PD_WORDS>
    <PD_IN word="1">3</PD_IN>
</IL_MODULE>
```

b) Inline terminal with two digital inputs and only the second input is active.

```
<IL_MODULE number="3">
    <MODULE_TYPE>DI</MODULE_TYPE>
    <PD_CHANNELS>2</PD_CHANNELS>
    <PD_WORDS>1</PD_WORDS>
    <PD_IN word="1">2</PD_IN>
</IL_MODULE>
```

c) Inline terminal with 16 digital inputs and the 13th and the 14th input are active.

```
<IL_MODULE number="7">
    <MODULE_TYPE>DI</MODULE_TYPE>
    <PD_CHANNELS>16</PD_CHANNELS>
    <PD_WORDS>1</PD_WORDS>
    <PD_IN word="1">12288</PD_IN>
</IL_MODULE>
```

The input word returns the value 12288 ($2^{12} + 2^{13}$).

d) Inline terminal with two analog inputs, only the first channel being active (14970).

```
<IL_MODULE number="10">
    <MODULE_TYPE>AI</MODULE_TYPE>
    <PD_CHANNELS>2</PD_CHANNELS>
    <PD_WORDS>2</PD_WORDS>
    <PD_IN word="1">14970</PD_IN>
    <PD_IN word="2">8</PD_IN>
    <PD_OUT word="1">0</PD_OUT>
    <PD_OUT word="2">0</PD_OUT>
</IL_MODULE>
```

PD_OUT

This area is used by all terminals with output data. The use of bits is identical to the use of "PD_IN".

In the event of an error in the Inline station, this is indicated in the diagnostic registers. The D LED flashes on the bus coupler. The process data is invalid because only internal values are indicated, not the values on local bus.

In order to make sure that only valid data is displayed, the diagnostic register must also always be requested. The same is valid in the event of a faulty configuration. In this case, local bus does not run and only internal values can be read in the XML file.

In the event of a peripheral fault, all data is valid, except for the data of the faulty terminal.

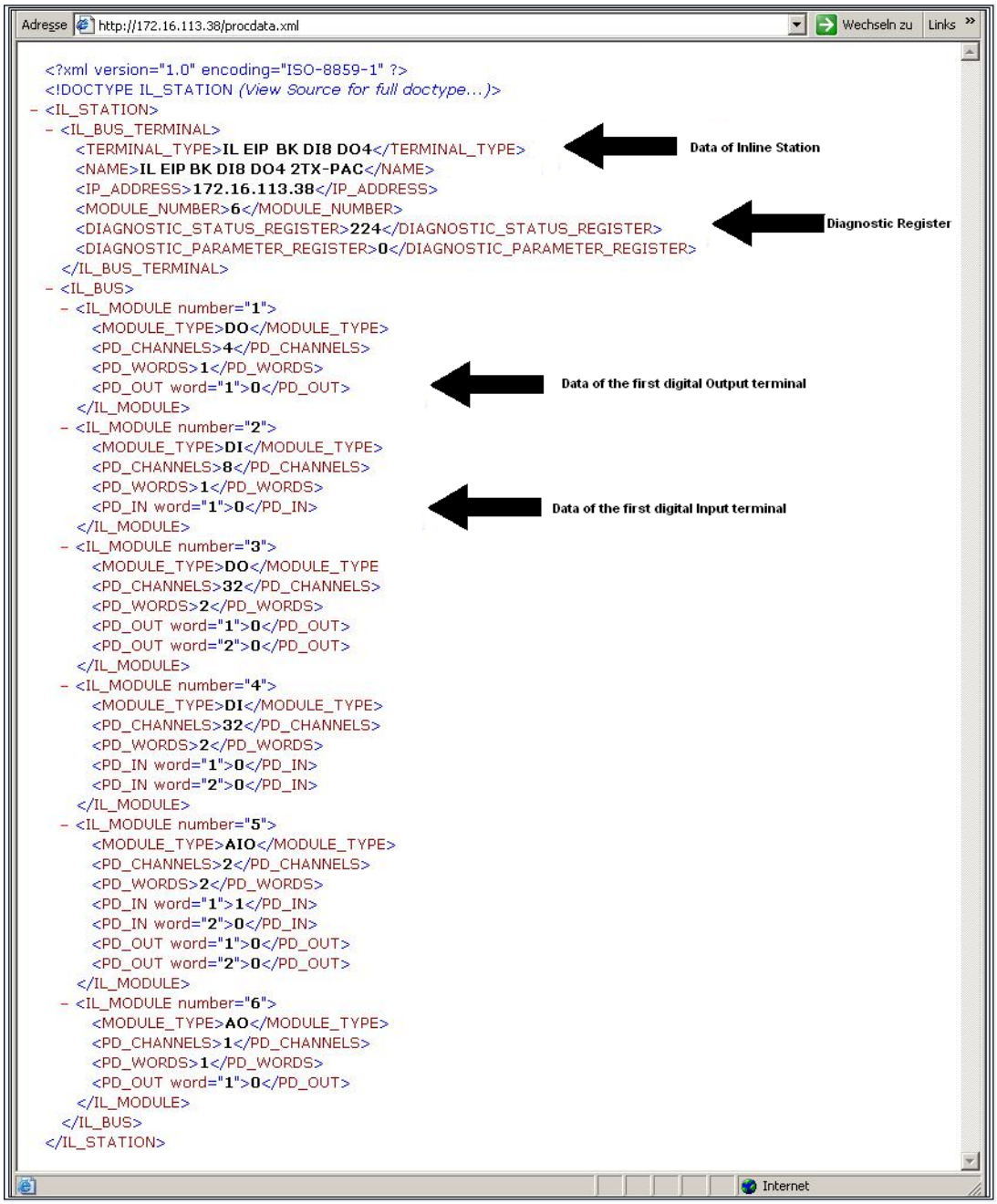


Figure 4-8 Screen for XML data

4.6 Traps

Trap Generation

When important events occur, e.g., a configuration change, the bus coupler sends a trap to a trap manager defined by the user. This enables the network administrator to react quickly to these events and to ensure network availability. Traps are usually only transmitted once.

Supported Traps

– ColdStart is sent twice each time the device is restarted.
– PasswordChange is sent after the password is changed successfully.
– FWHealth is sent after any changes to the firmware operating status.
– Configuration is sent after any changes to the hardware configuration.

SNMP is not supported in the first development step. The integration of SNMP is planned for the second step.

4.7 Autoconfiguration

That means you can now change your local bus configuration and make it active via the website. To use this function you have to stop your I/O connection. That means you cannot use the ADD ALL function when you have a running I/O connection.

4.8 Configuration

Configuration services can be accomplished by using the defined CIP Objects. Some configuration features are also made available to the user through the web server.

4.9 The CIP I/O Module Capacity

The bus coupler is capable of processing the maximum number of instances (points) for 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	61 instances
PCP Special Function Object	8 instances*
Serial Communication Object	8 instances*

*Total instances shared between the PCP Special Function and Serial Communication Objects cannot 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 61.
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 61 device maximum.
   d. Each FLM I/O device does count towards the 61 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.

4.10 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 59.

4.10.1 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, RS232, RS485, and others)

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

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

4.10.1.1 Configuration Methods

When creating, adding to, or changing an Inline Ethernet/IP 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

1. Electronic Data Sheet (EDS) file
2. Auto-configuration (no software required)
3. Sending an explicit message

Electronic Data Sheet (EDS) File Method

The EDS file is the software interface between the bus coupler and a configuration software package. The EDS file contains information about the number of produced and consumed bytes and user-settable parameters.

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 Ethernet/IP 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 (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 the Pad I/O parameter 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 an 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 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 45. 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 Reserve Digital Inputs, parameter Reserve Digital Outputs, parameter Reserve Analog Inputs and parameter 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 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 "MS" 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.

Plug and Play Mode

When in Plug and Play Mode, during power-up, the bus coupler will automatically attempt to start the local I/O. When disabled, the bus coupler will compare the stored I/O configuration with the connected I/O and start if equal.

Reboot not required to implement changes.

Setting the Store Config button will read in the current connected local bus and store it as the I/O configuration.

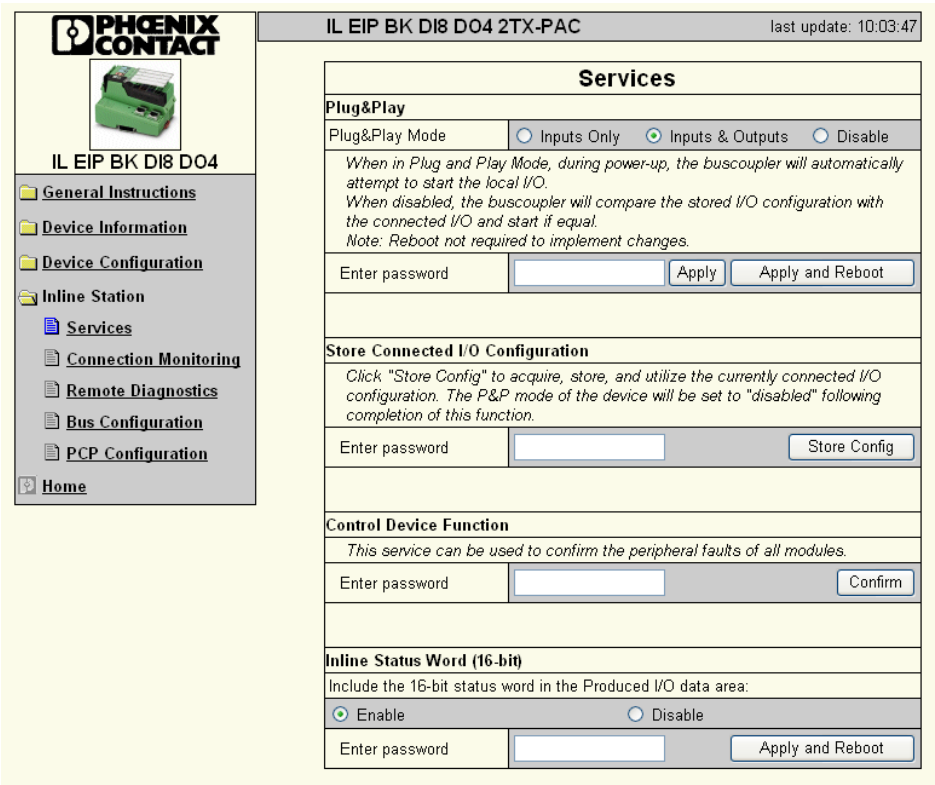


Figure 4-9 Plug and Play mode

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 "**Electronic Data Sheet (EDS) File Method" on page 33**. The parameters listed there 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 Ethernet/IP 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 Ethernet/IP bus coupler.

Service Code	16_{dec}, 10_{hex}
Class Code	100_{dec}, 64_{hex}
Instance	1
Attribute	7 (Add All I/O)
Attribute Data	1

Setting Attribute 7 (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 7 to be set is Service Code 16_{dec}, 10_{hex} (Set_Attribute_Single). Object classes, and services are described in Appendix B "Ethernet/IP Object Classes, Message Types, and Services". The construction software (RSNetworx for EIP or Pyramid EIP scan software) can be used to send explicit messages.

4.10.2 Analog Input (AI) Modules

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 "Class Instance Editor" in the construction software. Additionally, attributes 100 and 101 could be used to write a custom configuration value to the analog input module. Refer to the Thermocouple and RTD module paragraph below 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.

Appendix B “Ethernet/IP 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.

4.10.3 Thermocouple and RTD Modules

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 Appendix B “Ethernet/IP 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 4-10 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.

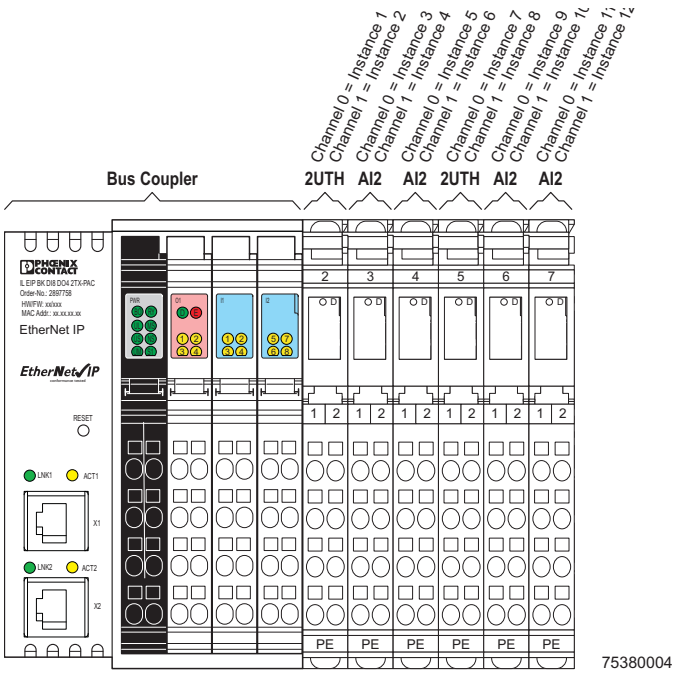


Figure 4-10 I/O station with analog input terminals and thermocouple terminals

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

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 for 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 Appendix B “Ethernet/IP Object Classes, Message Types, and Services” of this manual.

4.10.4 Special Function Modules

General Configuration

Special function modules such as the incremental encoder, absolute encoder and the high-speed counter are configurable through the produced/consumed 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.

4.11 Understanding I/O Memory Mapping

4.11.1 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 configuration software was explained in the previous section.

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

Consumed Data

Table 4-2 Consumed data

Data	Location			Default Setting	Control Object		
	Class (hex)	Instance	Attribute		Class (hex)	Instance	Attribute
Run/Idle	-	-	-	Enabled	64	1	43
Inline Control Byte	65	1	20	Disabled	64	1	32
Actual DOPs	09	X	3	Enabled	64	1	4
Reserved DOPs	09	X	3	Disabled	64	1	23
Padding	NULL			Enabled if number of DOP bytes uneven	64	1	14
Actual AOPs	0B	X	3	Enabled	64	1	6
Reserved AOPs	0B	X	3	Disabled	64	1	36
AIP Control Data	0A	X	101	Disabled	0A	X	102
Special Function Data	67	X	4	Enabled	67	X	7
PCP Data	–	–	–	Disabled	64	1	38
PCP Module X Process Data	69	X	20	Disabled	69	X	21
PCP Module X Request Fragment	69	X	15	Disabled	69	X	17
SCO Data	–	–	–	Disabled	64	1	39
SCO Module X Control Word	6A	X	6	Disabled	6A	X	31
SCO Module X Transmit Fragment	6A	X	10	Disabled	6A	X	32

Produced Data

Table 4-3 Produced data

Data	Location			Default Setting	Control Object		
	Class (hex)	Instance	Attribute		Class (hex)	Instance	Attribute
Run/Idle	–	–	–	Disabled	64	1	42
Inline Status Byte	65	1	4	Enabled	64	1	11
Inline First Faulted Module	65	1	5	Enabled	64	1	11
DIP Faults	08	X	4	Disabled	64	1	15
DOP Faults	09	X	4	Disabled	64	1	16
AIP Faults	0A	X	4	Disabled	64	1	17
AOP Faults	0B	X	4	Disabled	64	1	18
Special Function Faults	67	X	6	Disabled	64	1	19
Actual DIPs	08	X	3	Enabled	64	1	3
Reserved DIPs	08	X	3	Disabled	64	1	22
Padding	NULL			Enabled if number of DIP bytes uneven	64	1	14
Actual AIPs	0A	X	3	Enabled	64	1	5
Reserved AIPs	0A	X	3	Disabled	64	1	35
AOP Response Data	0B	X	100	Disabled	0B	X	102
Special Function Data	67	X	4	Enabled	67	X	7
PCP Data	–	–	–	Disabled	64	1	38
PCP Module X Process Data	69	X	19	Disabled	69	X	21
PCP Module X Response Fragment	69	X	16	Disabled	69	X	17
SCO Data	–	–	–	Disabled	64	1	39
SCO Module X Status Word	6A	X	5	Disabled	6A	X	31
SCO Module X Receive Fragment	6A	X	9	Disabled	6A	X	32

All produced elements of the same priority will be mapped together regardless of their location. However, their relative location to the bus coupler 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 Pad I/O 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 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 Ethernet/IP 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 4-11 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 4-11 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.

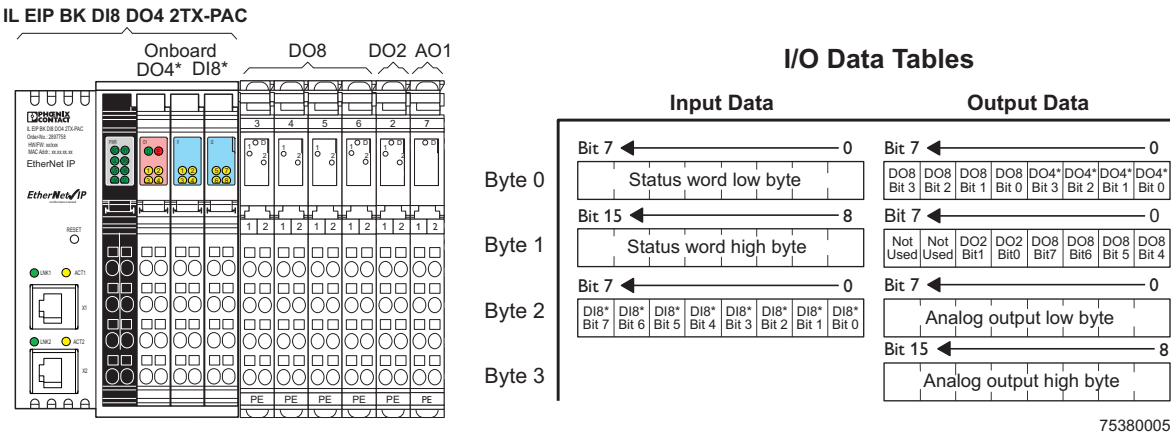


Figure 4-11 Example of an I/O table (memory map) consisting of analog and digital output modules

Figure 4-12 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

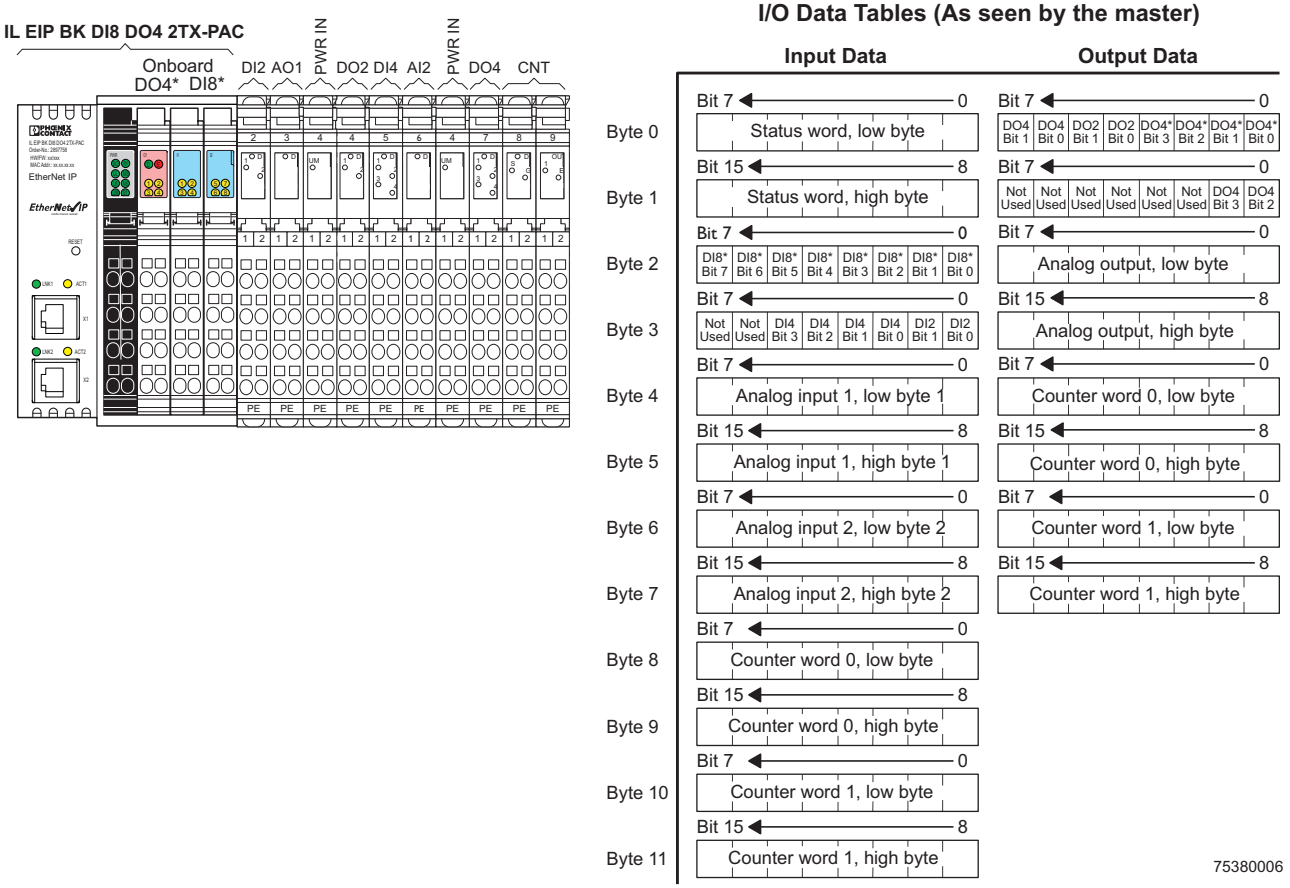


Figure 4-12 Example of an I/O table (memory map) consisting of digital and analog, input and output modules

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 4-12 shows that the total number of input bytes is 12 (byte 0 through byte 11). This includes the Inline Status word. Figure 4-12 also shows that the total number of output bytes is 8 (byte 0 to byte 7).

4.11.2 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 37.

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.

4.12 I/O Data Transfer

A detailed explanation of the following objects and their attributes can be found in Appendix B “Ethernet/IP Object Classes, Message Types, and Services”.

I/O data transfer can be accomplished by establishing an I/O connection (implicit) or by establishing a message connection (explicit). For operational considerations, see Section “Maximum Connection Consideration” on page 50.

Implicit

An implicit connection provides a dedicated path from a producing application to one or more consuming applications and is typically used for real-time data transfer.

Explicit

An explicit connection is a generic connection between two devices where a request is sent and an acknowledge is expected. It is used for configuration and data transfer.

4.12.1 I/O Scan Methods

The Ethernet/IP 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:

– Cyclic
– Change of State (COS)
– Application triggered

4.12.2 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.

4.12.2.1 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 4-13 shows how "actual input" or "latched" data is selected.

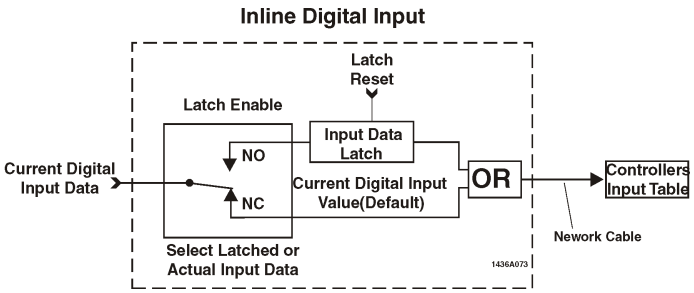


Figure 4-13 Latched or current data selection

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 produced data 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.

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

The DOP object models digital outputs in the Ethernet/IP 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.

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

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

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

The AOP object models analog outputs in the Ethernet/IP 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.

Accessing Analog and Digital Instances

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

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 Ethernet/IP object.

– Incremental encoder
– Absolute encoder
– High Speed counter

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

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

4.13 System Operational Guidelines

4.13.1 Repeat Packet Interval (RPI) Settings

When setting up an Ethernet/IP system, care must be exhibited when setting the RPI value in the control system scanner. Depending on the vendor's implementation, this value may range from 5 ms to 100's of ms in 5 ms increments. The RPI value establishes the rate at which the scanner will send Ethernet/IP messages (packets). It also establishes the maximum rate that the Inline station (in this case) will send messages. Though the value is set in the PC/PLC scanner is also transferred by the scanner to the BK so that the system is working on the same time base. In addition to setting the speed of the network updates, the RPI value is used to set the rate at which the scanner expects to receive back in time, the scanner will assume there is a problem, stop I/O communications and the I/O station will go into its fault response mode.

As is true with most Ethernet/IP devices (Inline included) the CPU in the BK, splits its time servicing the Ethernet/IP network, performing internal functions such as updating internal websites, and of course scanning I/O. In larger I/O systems setting the RPI too low may overload the BK with a level of network traffic it can not accomplish. In these cases the module will stop communicating and go into a fault state. In addition to physically larger I/O stations use of PCP communications (RS-232) modules, etc. requires extra processing therefore potentially higher (larger ms) RPI settings. While actual settings will vary based on the station configuration and application requirements, as a general rule of thumb the following considerations should be followed:

– Configurations requiring RPI rates below 10ms should be tested in advance to confirm operation.
– Configurations requiring PCP modules should use RPI settings at a minimum of 20 ms. Settings below 20 ms should be tested in advance.

4.13.2 Maximum Connection Consideration

The module firmware supports up to 128 connections total (any mix implicit or explicit). Application considerations such as CPU loading, frequency of data updates (RPI parameter), and I/O quantity scanned will impact the actual maximum connections. Fewer connections allow faster data update rates (RPI value). For maximum I/O performance the quantity of connections should be limited to 8 or less.

5 Diagnostics

5.1 Diagnostic and Status Indicators

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

Diagnostic 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 IL SYS INST UM E user manual and to the module-specific data sheet to see which diagnostic and status LED indicators apply to that module.

LEDs on the Ethernet/IP Bus Coupler

Figure 5-1 and Table 5-1 provide the different LED states that can be read from the bus coupler.

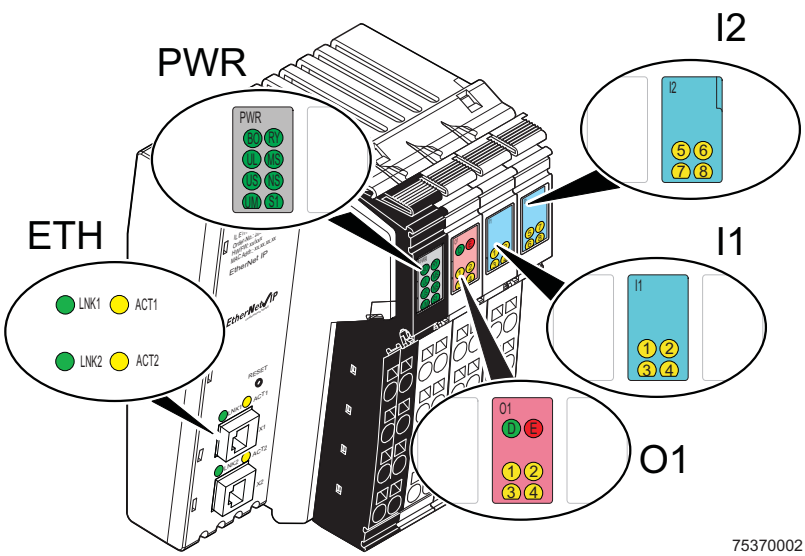


Figure 5-1 Diagnostic and status indicators on the bus coupler

Table 5-1 Local status and diagnostic indicators on the bus coupler

LED	Color	Meaning	State	Description of the LED States
ETH				
LNK1	Green	Link at port 1	ON	Link connection at port 1 present
			OFF	Link connection at port 1 not present
LNK2	Green	Link at port 2	ON	Link connection at port 2 present
			OFF	Link connection at port 2 not present
ACT1	Yellow	Activity on port 1	ON	Data transmission on port 1 active
			OFF	Data transmission on port 1 not active
ACT2	Yellow	Activity on port 2	ON	Data transmission on port 2 active
			OFF	Data transmission on port 2 not active
PWR				
BO	Green	**Bo**ot	ON	Boot loader active, firmware started
			Flashing	Waiting for BootP/DHCP reply
			OFF	Firmware started successfully
UL	Green	$\mathbf{U_L}_{\text{ogic}}$	ON	24 V communications supply / 7.5 communications power present
			OFF	24 V communications supply / 7.5 communications power not present
US	Green	$\mathbf{U_S}_{\text{egment}}$	ON	24 V segment circuit supply present
			OFF	24 V segment circuit supply not present
UM	Green	$\mathbf{U_M}_{\text{ain}}$	ON	24 V I/O supply present
			OFF	24 V I/O supply not present
RY	Green	**Ready**		Ready
			ON	Connection to a process data client established
			Flashing	Firmware ready to operate
			OFF	Firmware not active
MS	Red/ green	**M**odule **s**tatus		Device status
			Green ON	Normal operation
			Red ON	Unrecoverable error
			Flashing green	– Device not configured, or device configuration not complete or faulty – Device in standby mode
			Flashing red	Recoverable error
			Flashing red-green	Selftest
			OFF	No supply voltage

Table 5-1 Local status and diagnostic indicators on the bus coupler

LED	Color	Meaning	State	Description of the LED States
NS	Green/ red	**N**etwork **s**tatus		Network status
			Green ON	Module is online and has established a connection
			Red ON	Error preventing communication with the network (e.g., bus offline or double MAC ID).
			Flashing green	Device online, connections not established Device has finished the "double MAC ID" test but has not established connections to other nodes.
			Flashing red	One or more connections in timeout state
			Flashing red-green	Selftest
			OFF	Device not online Device has not yet finished the "double MAC ID" test. Device has no IP address or is not supplied with voltage.
S1	Green	Boot source **s**tatus	ON	IP parameters received from BootP/DHCP server
			Flashing	BootP request/responses in process
			OFF	Stored IP parameters are used
O1				
D	Green	**D**iagnos-tics	ON	Data transmission within the station active
			Flashing	Data transmission within the station not active
E	Red	**E**rror	ON	Short circuit/overload of outputs
			OFF	No short circuit/overload of outputs
1-4	Yellow	Output **1** to Output **4**	ON	Output active
			OFF	Output not active
I1, I2				
1-8	Yellow	Input **1** to Input **8**	ON	Input active
			OFF	Input not active

5.2 Available Network Diagnostics

A detailed explanation of object classes can be found in Appendix B “Ethernet/IP 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 produced data. 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}).

5.2.1 Inline Status Word

You can make available the data of the Inline Status word in the form of two byte diagnostic data in the produced data. 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 does not add the two bytes to the produced data size by default.
If the Inline Status word is displayed, it updates the status automatically, i.e., if an error is cleared, the status is reset to 0.

To add the data of the Inline Status word to the produced data, activate the status word:

- Set attribute 11 “Use Inline Status” in the Configuration Object (class code 100_{dec}, 64_{hex}) to 1.
- Then set attribute 7 “Add All I/O” in the Configuration Object (class code 100_{dec}, 64_{hex}) to 1.

To remove the data of the Inline Status word from the produced data, disable the status word:

- Set attribute 11 “Use Inline Status” in the Configuration Object (class code 100_{dec}, 64_{hex}) to 0.
- Then set attribute 7 “Add All I/O” in the Configuration Object (class code 100_{dec}, 64_{hex}) to 1.

5.2.2 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 MS 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 MS 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 settings are:

0	None
1	Minor Recoverable Fault
2	Major Recoverable Fault

5.2.3 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. The EDS parameter number 23, "Max Retry", will allow the module to retransmit the data cycle up to the number of times that the "Max Retry" parameter is set to. If the transmission does not pass the CRC (Cyclic Redundancy Check) after the "Max Retry" has expired then the CRC error bit is set.

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,) is 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 EtherNet/IP™ Fault State

This bit can only be set in the Fault Response mode 2. It is made available to let the application know that the local outputs have gone to their preprogrammed EtherNet/IP™ fault state and will no longer respond to the controller.

Bit 7

Reserved for future use.

5.2.4 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 (including two devices on the bus coupler) + 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 MS LED on the bus coupler will blink red.

5.2.5 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 Ethernet/IP implicit or 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.

5.2.6 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 46.

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.

5.2.7 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 Appendix B “Ethernet/IP 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 instances 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.

5.2.8 Fault/Idle State and Value

The bus coupler supports the standard EtherNet/IP™ 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

Appendix B “Ethernet/IP 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.

5.2.9 Inline 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 Appendix B “Ethernet/IP Object Classes, Message Types, and Services”. A list of these codes, in Inline format "IL", is shown in Table 5-2.

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 inTable 5-2. If the format has been changed, the user must refer to the module specific data sheet to determine what error code has been received.

Table 5-2 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

5.2.10 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.

A Serial and Other PCP Inline Modules

A 1 General

A 1.1 Exchanging Device Parameters

Communication Relationships

Before data can be exchanged between two PCP devices, a logical connection must exist between the two devices. Such logical connections are called communication relationships. Communication relationships are established between application processes.

Communication Relationship List (CRL)

The host controller board creates a list for every PCP device. Within this list, all permitted communication relationships are specified, independent of their time of use. The connection type and the context requirements (the connection parameters) by which the communication relationship can be established, are stored in this communication relationship list (CRL). The CRL can be manually changed as required.

The logical connections configured in the communications relationship list guarantee a smooth data exchange between two communication devices. The connection parameters (context conditions) of both communication partners are checked for agreement before the information is exchanged - during the connection establishment.

Communication Reference (CR)

The communication relationship list is set up line by line. Every permissible communication relationship contains a number, the communication reference (CR). The communication relationship is thereby unambiguously coded. An unambiguous code is necessary in order to distinguish between the individual devices. The Inline bus coupler numbers the devices automatically during local bus initialization. It allocates the numbers beginning with 2 in the order of the physical bus configuration.

A 1.2 Communication Services

The client and the server use communication services for the service order and execution. If, in our example, the parameter value for the speed acceleration is to be transmitted to the frequency inverter, then this is done by means of a write command, which implies using the write service.

A service is divided into individual basic service operations (primitives). First, a service request is transmitted, in our example a Write Request: the parameter value is transmitted. The frequency inverter receives the parameter value as a service input, in our example a Write Indication. After execution of the command, when the parameter value has been entered, the frequency inverter transmits a service confirmation, a Write Response.

This is sent back as a service confirmation, in our example as a Write Confirmation. The bus transmits the content of the information in the form of a PDU message (Protocol Data Unit).

A service consists of the following four basic operations:

Service Request The client transmits its service request with this basic operation.

Service Indication The service input is reported to the server with this basic operation.

Service Response The server transmits the service confirmation with this basic operation.

Service Confirmation This basic operation confirms execution of the service, it is the service acknowledgement.

Figure A-1 gives an overview of the interaction of the basic operations..

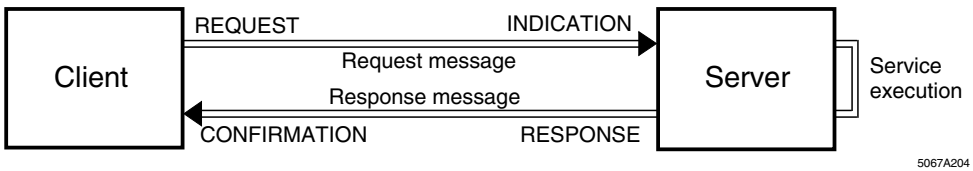


Figure A-1 PCP basic service operations (confirmed services)

A 1.3 Communication Phases

The Inline bus coupler supports connection-oriented (one to one) communication relationships. The communication-oriented communication is divided into three phases:

– Connection establishment
– Data transfer
– Connection abort

Connection Establishment Phase

In the connection establishment phase, a PCP device acting as a client tries to establish a communication link to another PCP device acting as a server. In the process, the context conditions (the connection parameters) that are determined in the communication relationship list of both devices, are checked. If the context requirements agree then the data transfer phase is initiated. Otherwise, the connection establishment is cancelled with an error message.

Data Transfer Phase

In the data transfer phase, the PCP devices exchange data under the context conditions. The connection remains until it is intentionally aborted or a communication error occurs.

Connection Abort Phase

After the data exchange has been completed the connection can be cancelled by a connection abort. If a communication error occurs then the connection is automatically aborted. Data exchange can then only be carried out after a renewed connection establishment.

A 1.4 Description of the Communication Services

Each PCP service is described in this section together with all of its basic operations. The parameters that apply in all basic operations are shown from the beginning and are not always repeated for the individual services

Command Code / Message Code

Preset number that is set for each command and each message. Command Code and Message Code are each given in the description of the basic operations.

Parameter Counter

Indicates the number of the subsequent data words, each of 2 bytes. If there is only a 1 byte parameter in a line then this is also counted.

Invoke ID

Invoke ID does not apply to this PCP version 00. The value is therefore always 00.

Communication Reference (CR)

CR indicates the communication reference that is set for the communication relationship between host controller board and the remote device. The standard setting for the first PCP module is CR=2, for the second PCP module CR=3, etc.

Basic Service Request

Service
Modules
Index (LSB)
Index (MSB)
Subindex
Data [0]
Data [x]

Service Request with Invoke ID

Service
Invoke ID
Modules
Index (LSB)
Index (MSB)
Subindex
Length
Data [0]
Data [x]

Basic Service Response Success

Service
Status
Length
Data [0]
Data [x]

Basic Service Response Success with Invoke ID

Service
Invoke ID
Modules
Index (LSB)
Index (MSB)
Subindex
Length
Data [0]
Data [x]

Basic Service Error Response

Service
Status
Error Class
Error Code
Additional Error (LSB)
Additional Error (MSB)

Service w/Invoke ID Error Response

Service
Invoke ID
Status
Error Class
Error Code
Additional Error (LSB)
Additional Error (MSB)

Basic Service Abort Response

Service
Status
Abort ID
Abort Code
Abort Reason [0]
Abort Reason [1]

Service with Invoke ID Abort Response

Service
Invoke ID
Status
Abort ID
Abort Code
Abort Reason [0]
Abort Reason [1]

Basic Service Reject Response

Service
Status
PDU Type
Reject Code
Additional Error (LSB)
Additional Error (MSB)

Service with Invoke ID Reject Response

Service
Invoke ID
Status
PDU Type
Reject Code
Additional Error (LSB)
Additional Error (MSB)

Supported Services

Code (hex)	Service
00	NOP
01	PCP Read
02	PCP Write
03	Read PDU Size
09	PCP Read with Invoke ID
10	PCP Write with Invoke ID

A 2 Communications Methods

Communications to an Inline serial module and to a generic PCP Inline module can be accomplished in two main ways each. The type of module being used, either serial or generic PCP, determines what types of communication methods are available to the user.

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 Ethernet/IPTM scanner and/or personal preference.

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

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-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 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

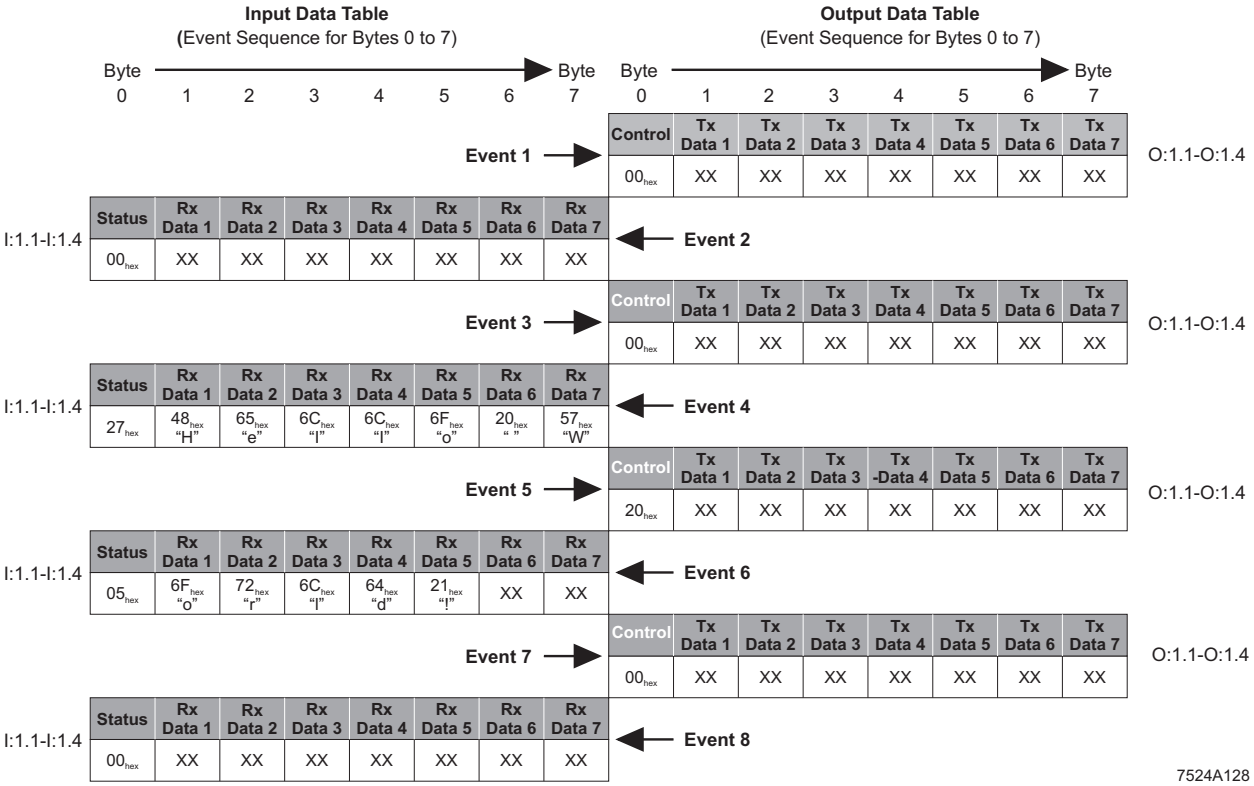


Figure A-2 Serial fragmented read example

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

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-2.

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-3 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 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

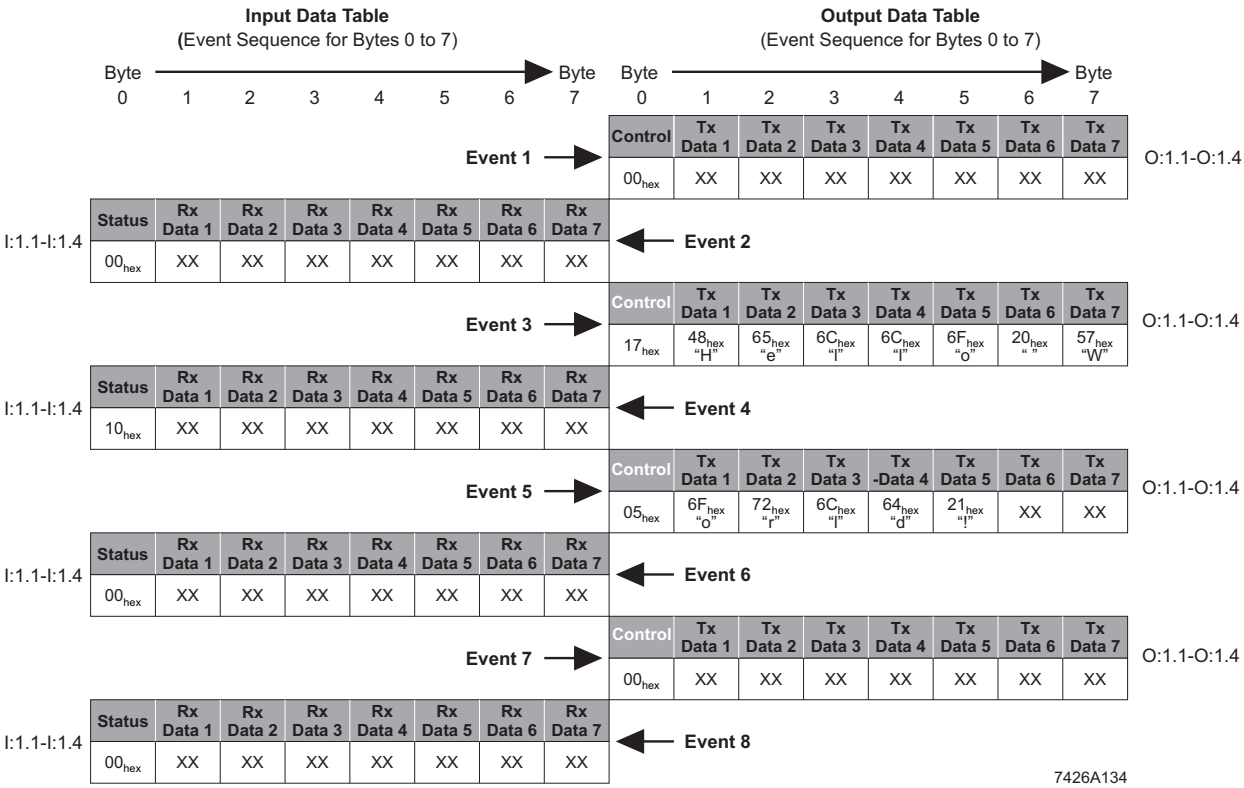


Figure A-3 Serial fragmentation write example

The following paragraphs explain in detail those events listed in Figure A-3.

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-3.

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-4, 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.

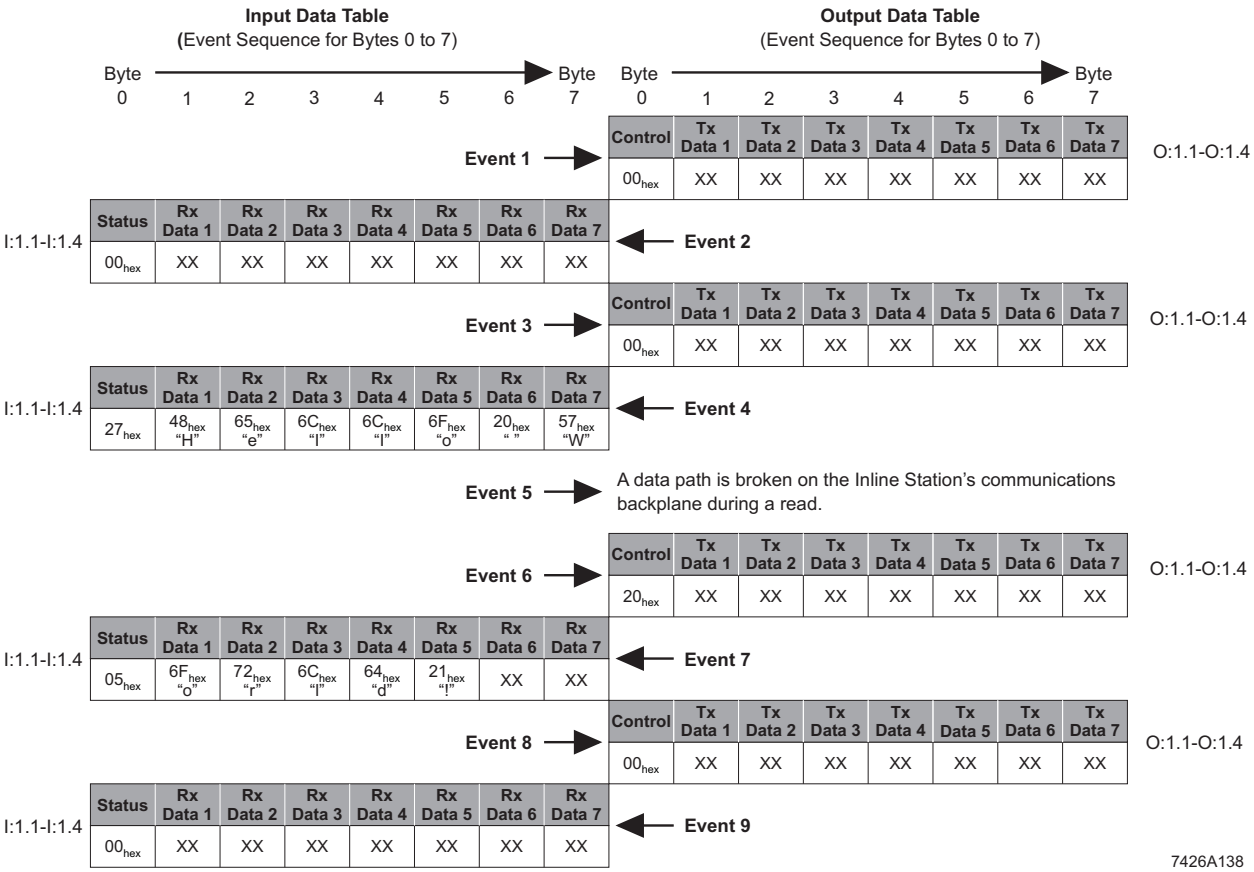


Figure A-4 Serial fragmentation error example 1

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 Ethernet/IP 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 Ethernet/IP 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 actual 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 Peripherals Communications 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 Ethernet/IP 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

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

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 additional 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-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

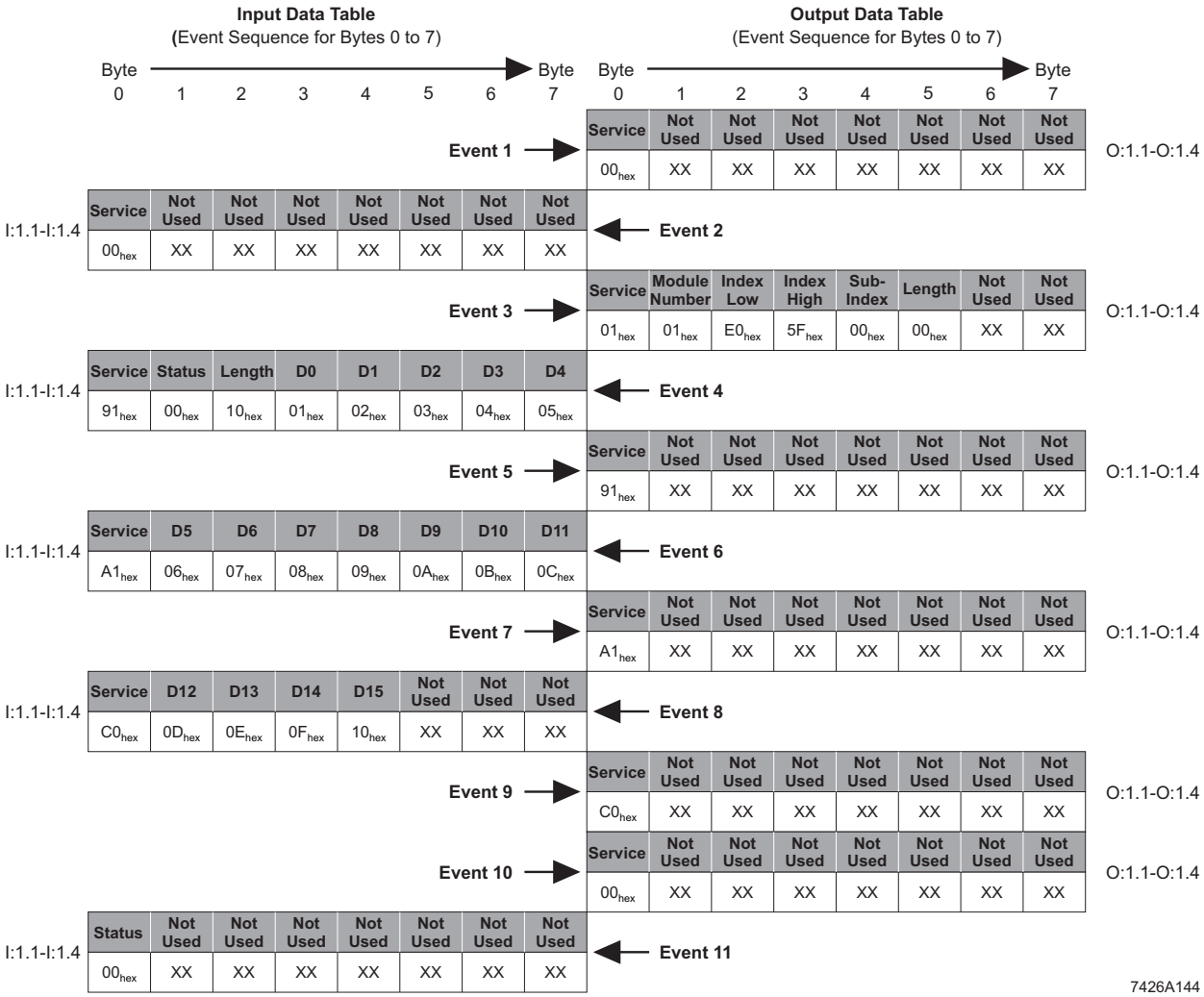


Figure A-5 I/O events for a read sequence using PCP fragmentation

In the following paragraphs the events in Figure A-5 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-5.

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 is 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-5. 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 acknowledgement

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-6 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

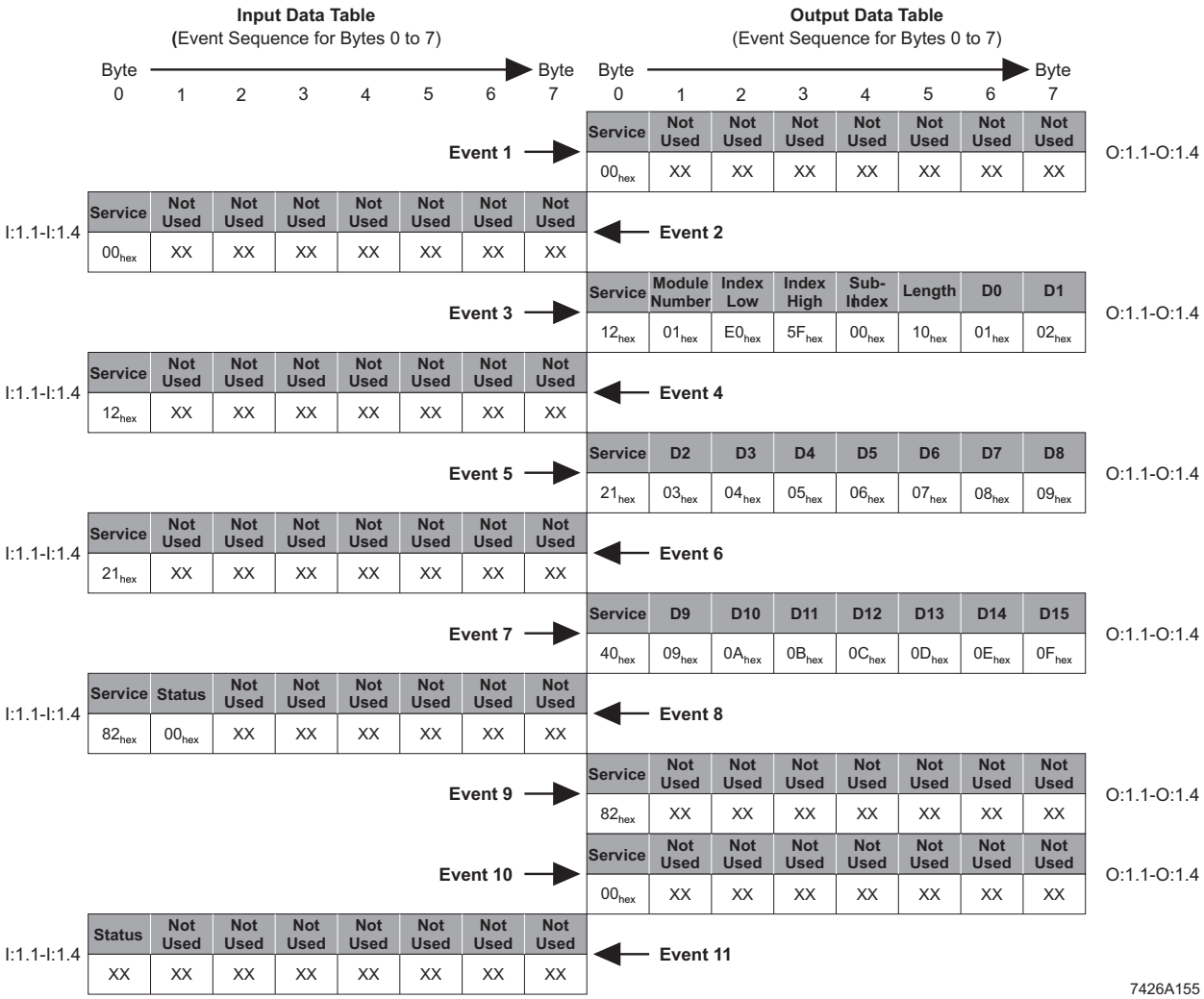


Figure A-6 I/O events for a write sequence using PCP fragmentation

In the following paragraphs the events in Figure A-6 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-6.

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	Subindex	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 signaling 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-7, will show how the fragmentation will react during a break in the communications path on the Inline station's backplane.

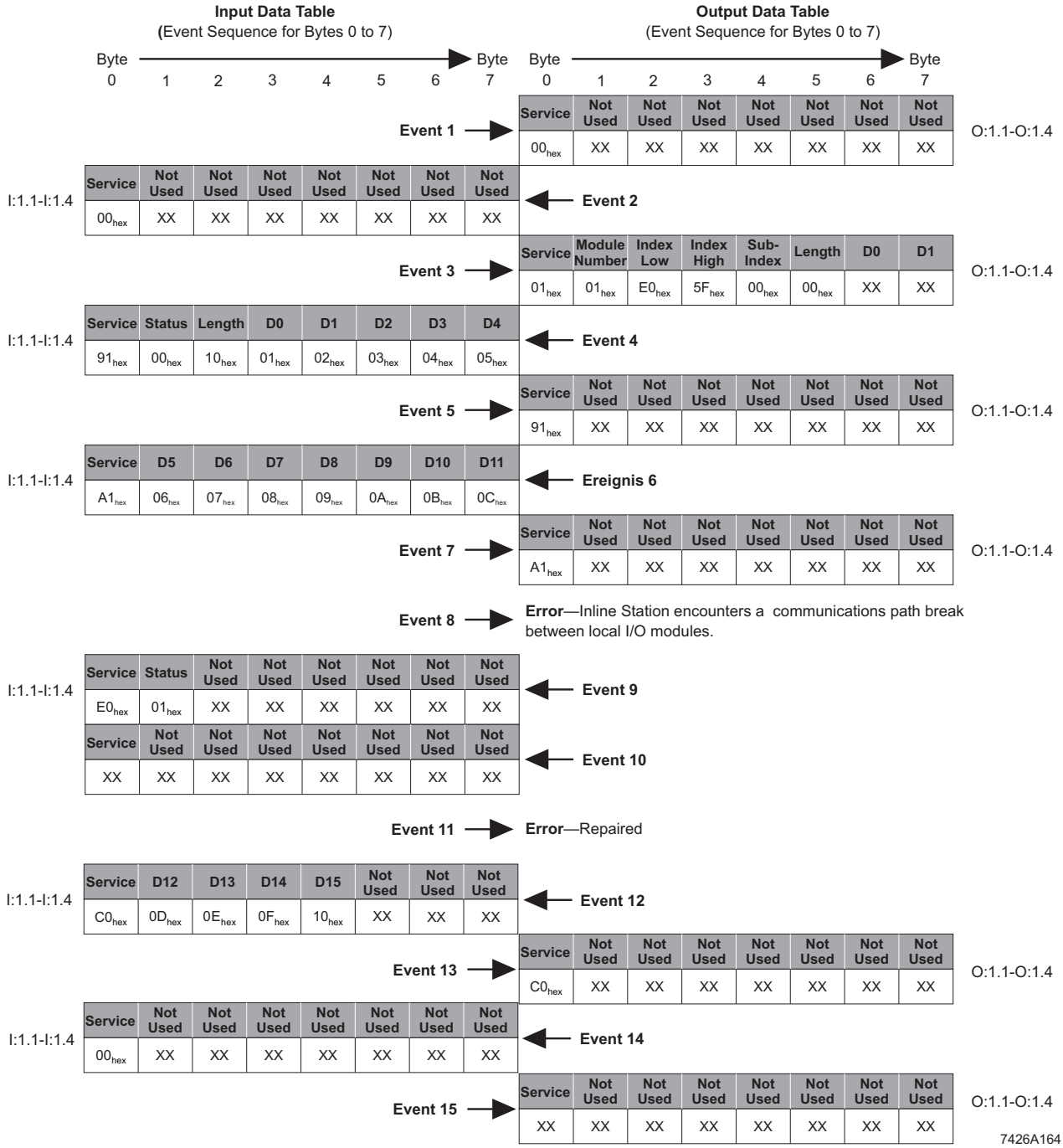


Figure A-7 Local communications error sequence using PCP fragmentation

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 65.

The ability to send explicit messages is a function of the Ethernet/IP 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 Appendix B “Ethernet/IP 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_{de}c)

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, 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 4, "Startup/Operation" provides additional information in regard 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 4, “Startup/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 an Ethernet/IP 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 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 and 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 Appendix 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 Ethernet/IP I/O".

To add or remove the Serial Data (8 bytes of fragment data) from the Ethernet/IP 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 Ethernet/IP I/O".

To add or remove the Other PCP Data from the Ethernet/IP 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 4, "Startup/Operation" provides additional information in regard to how data is mapped into the scanner and other considerations.

I/O Mapping Rules

Section 4, "Startup/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

Appendix B “Ethernet/IP 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 an Ethernet/IP 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 Ethernet/IP Object Classes, Message Types, and Services

B 1 General

The IL EIP BK DI8 DO4 2TX-PAC is an Ethernet/IP capable adapter that functions as a generic device type. The BK maps the I/O connected via the backplane to standard or user defined CIP objects.

The bus coupler supports CIP 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 Communication Objects.

B 2 CIP Class Services

The IL EIP BK DI8 DO4 2TX-PAC supports the following class services and instance services.

Service Code		Service Name
dec	hex	
01	01	Get_Attribute_All
02	02	Set_Attribute_All
05	05	Reset
09	09	Delete
14	0E	Get_Attribute_Single
16	10	Set_Attribute_Single

B 3 CIP Object Classes

The IL EIP BK DI8 DO4 2TX-PAC supports the following CIP object classes:

Class Code		Object Type	On Page
dec	hex		
01	01	Identity Object	105
02	02	Router Object	107
04	04	Assembly Object	108
08	08	Digital Input Point (DIP) Object	109
09	09	Digital Output Point (DOP) Object	111
10	0A	Analog Input Point (AIP) Object	113
11	0B	Analog Output Point (AOP) Object	115
100	64	Configuration Object	117
101	65	Inline Interface Object	122
102	66	Inline Module Object	125
103	67	Inline Special Function Object	127
104	68	COS Mask Object	129
105	69	PCP Object	132
106	6A	Serial Communications Object	136
244	F4	Port Object Class Definition	141
245	F5	TCP/IP Interface Object	142
246	F6	Ethernet Link Object	144

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	6
7	Get/Set	Max Instance Attribute	UINT	7

B 4.2 Identity Object Instance Attributes

Attribute	Access	Name	Type	Value (see B 4.4)
1	Get	Vendor	UINT	562
2	Get	Product Type	UINT	0 = Generic Device
3	Get	Product Code	UINT	8167 (3)
4	Get	Revision	STRUCT OF	(4)
		Major Revision	BYTE	1
		Minor Revision	BYTE	1
5	Get	Device Status	UINT	(5)
6	Get	Serial Number	UINT	(6)
7	Get	Device Name	STRUCT OF	(7)
		Length	USINT	26
		Name	STRING [6]	IL EIP BK DI8 DO4 2TX-PAC

B 4.3 Identity Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
01	01	Yes	Yes	Get_Attribute_All
05	05	No	Yes	Reset
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 4.4 Values for Identity Object Class Attributes

(3): Product Code The product code is fixed at 8167 for the IL EIP BK DI8 DO4 2TX-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
10	Major Recoverable Fault	= 0, no fault = 1, major recoverable faults (Loss of +24 V DC)
11	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 and is guaranteed to be unique across all product lines produced by Phoenix Contact.

(7): Device Name The Device Name provides a character array containing the short string IL EIP BK DI8 DO4 2TX-PAC.

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

The 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	0
7	Get	Max Instance Attribute	UINT	0

B 5.2 Router Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value
1	Get	Router Class List	ARRAY	07 00 01 00 02 00 04 00 06 00 F4 00 F5 00 F6 00
2	Get	Max Connections	UINT	128

B 5.3 Router Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single

B 6 Assembly Object (Class Code: 04_{dec}, 04_{hex})

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

B 6.1 Assembly Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Class ID	UINT	101

B 6.2 Assembly Object, Instance 100 Attributes

Attribute	Access	Name	Type	Value
3	Get	Consumed (output) Data	Array of USINT	

B 6.3 Assembly Object, Instance 101 Attributes

Attribute	Access	Name	Type	Value
3	Get	Produced (input) Data	Array of USINT	

B 6.4 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 6.5 Assembly Instance 100

Assembly instance 100 is used to consume the I/O data and consists of a variable number of digital output states and a variable number of analog output values as specified by the Configuration Object.

B 6.6 Assembly Instance 101

Assembly instance 101 is used to generate the I/O produced data. Assembly Instance 101 consists of a variable number of bytes that is specified by the Configuration Object.

B 7 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 7.1 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 7.2 DIP Object, Instance 1.. (Number of DIPs) Attributes

Attribute	Access	Name	Type	Value (see B 7.4)	
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 = Off, 1 = Latch	(100)
101	Get/Set	Latch State	BOOL	0 = Latch Low, 1 = Latch high	(101)

B 7.3 DIP Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
14	0E	Yes	Yes	Get_Attribute_Single

B 7.4 Values for DIP Object Class Attributes

(3): Input State

This attribute 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 ... 24. The status bits are cleared when the +24 V DC power is restored.

(100): Latch Enable

When set to 1, the corresponding input instance is latched at the state defined in Attribute 101.

(101): Latch State

This attribute defines the state of the inputs that will be latched. 0 = latch low input 1 = latch high input.

B 8 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 an 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 8.1 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/Set	Max Instance Attribute	UINT	8

B 8.2 DOP Object, Instance 1.. (Number of DOPs) Attributes

Attribute	Access	Name	Type	Value (see B 8.4)	
3	Get/Set	Output State	BOOL	State of Output	(3)
4	Get	Output Status	BOOL	Status of Output	(4)
5	Get/Set	Fault State	BOOL	0 = Fault value, 1 = No change	(5)
6	Get/Set	Fault Value	BOOL	0 = OFF, 1 = ON	(6)
7	Get/Set	Idle State	BOOL	0 = Idle value, 1 = No change	(7)
8	Get/Set	Idle Value	BOOL	0 = OFF, 1 = ON	(8)
100	Get/Set	Hold on Available State	BOOL	0 = OFF, 1 = ON	(100)

B 8.3 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

B 8.4 Values for DOP Object Class Attributes

(3): Output State

This attribute contains the desired state of the 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 (default)	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. (Default = 0)

(7): Idle State

The Idle State determines what action is taken if an idle condition is detected. Idle conditions occur if an I/O request is received with less than the calculated number of bytes or the Run/Idle header is set to idle and the Run/Idle header is used. Refer to the Configuration Object to determine the size of the I/O consumed data. An I/O request of 0 bytes is typically used to force an idle condition.

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

(8): Idle Value

The Fault Value is used to set the output if the Idle State bit is clear and an idle condition occurs. (Default = 0)

(100): Hold on Available State

In the event of interruption of the connection, setting the attribute ensures that the value of the set output remains.

B 9 Analog Input Point (AIP) Object (Class Code: 10_{dec}, $0A_{hex}$)

The IL EIP BK DI8 DO4 2TX-PAC supports variable analog inputs. There is a separate instance for each analog channel available on the device.

B 9.1 AIP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	(Number of AIPs)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	102

B 9.2 AIP Object, Instance 1.. (Number of AIPs) Attributes

Attribute	Access	Name	Type	Value (see B 9.4)	
3	Get	Value	UINT	0 ... $FFFF_{hex}$	(3)
4	Get	Status	BOOLEAN	0 = ok	(4)
7	Get/Set	Range	USINT	2	(7)
8	Get	Type	USINT	6	(8)
100	Get/Set	Override Range	BOOL	0 = No 1 = Yes	(100)
101	Get/Set	AIP Control Data	UINT		(101)
102	Get/Set*	AIP Control Data in I/O	BOOL	0 = Not in consumed data 1 = In consumed data	(102)

B 9.3 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 9.4 Values for AIP Object Class Attributes

(3): Value

This attribute contains the desired value to be sent to the AOP.

(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 Section “Identity Object (Class Code: $01_{dec, 01hex}$)” on page 105).

(7): Range

The AIP range value is used to configure the measuring range of the Inline analog input module. The AIP Range values are stored in non-volatile memory.

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

(8): Type

The Type attribute determines the data type to be used by the attribute Value. The type is fixed as UINT (6).

(100): Override Range

When set, the control data is sent to the AIP as the control word instead of the value corresponding to the range.

(101): AIP Control Data

If the Override Range attribute is set, the value in this attribute is sent to AIP as the control word instead of the value corresponding to the range. Note: this is a NV attribute and constant changing will use NV life cycles. Use Control Data in I/O for dynamic changing of the control word.

(102): Control Data in I/O

When set the Control Data is added to the I/O connection, allowing dynamic changing of the AIP control word.

B 10 Analog Output Point (AOP) Object (Class Code: 11_{dec}, $0B_{hex}$)

The IL EIP BK DI8 DO4 2TX-PAC supports Analog Output Points (AOP). There is a separate instance for each analog channel available on the device.

B 10.1 AOP Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	(Number of AOPs)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	100

B 10.2 AOP Object, Instance 1.. (Number of AOPs) Attributes

Attribute	Access	Name	Type	Value (see B 10.4)	
3	Get/Set	Value	UINT	0 ... $FFFF_{hex}$	(3)
7	Get/Set	Output Range	BYTE		(7)
8	Get	Output Data Type	USINT	6 = UINT	(8)
9	Get/Set	Fault State	BYTE	0 ... 3	(9)
10	Get/Set	Idle State	BYTE	0 ... 3	(10)
11	Get/Set	Fault Value	INT	0 ... $FFFF_{hex}$	(11)
12	Get/Set	Idle Value	INT	0 ... $FFFF_{hex}$	(12)
100	Get	AOP Response Data	UINT		(100)

B 10.3 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 10.4 Values for AOP Object Class Attributes

(3): Value

The analog output value is given in offset binary format. The value provided must be in the range 0 ... 65535 (0 ... $FFFF_{hex}$). Reference Inline analog output module specific data sheet for actual values.

(7): Output Range

The analog output range defaults to a fixed value of 1 (0 V DC to +10 V DC).

If the AO module reports that it the range is different, it will change to 3 (-10 V DC to +10 V DC).

(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 (0 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 an I/O request packet is received with less than the calculated number of bytes or the run/idle header is set to idle and the run/idle header is enabled. Refer to the Configuration Object to determine the size of the I/O request data. An I/O request of 0 bytes is typically used to force an idle condition if the run/idle header is not enabled.

Idle State	Action Taken
0	Hold the last value
1	Set to low limit (0 V DC)
2	Set to high limit (+10 V DC)
3	Set to value determined by Fault 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 ... $FFFF_{hex}$).

(12): Idle Value

The Fault 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 ... $FFFF_{hex}$).

(100) AOP Response Data

This attribute allows the user to see the response coming back from the AOP module.

B 11 Configuration Object (Class Code: 100_{dec}, 64_{hex})

The Configuration Object allows the user to configure what data they want in each I/O connection. In addition, the configuration object gives access to several operational parameters such as status info.

B 11.1 Configuration Object Class Attributes

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

B 11.2 Configuration Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 11.4)
3	Get/Set	Number of Digital Inputs	USINT	(3)
4	Get/Set	Number of Digital Outputs	USINT	(4)
5	Get/Set	Number of Analog Inputs	USINT	(5)
6	Get/Set	Number of Analog Outputs	USINT	(6)
7	Get/Set	Add All I/O	BOOL	(7)
8	Get/Set	Accept New Configuration	BOOL	(8)
9	Get	Module Change Status	BYTE	(9)
10	Get/Set	Add All Mode	BYTE	(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 I/O	BOOL	(14)
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 Function Faults	UINT	(19)
20	Get	Produced Size	UINT	(20)
21	Get	Consumed Size	UINT	(21)
22	Get/Set	Bytes Reserved for DIPs	UINT	(22)
23	Get/Set	Bytes Reserved for DOPs	UINT	(23)
24	Get	Fault Mode	USINT	(24)

Attribute	Access	Name	Type	Value (see B 11.4)
32	Get/Set	Inline Control Byte in Poll	BOOL	(32)
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)
42	Get/Set	Run/Idle in Produced I/O	BOOL	(42)
43	Get/Set	Run/Idle in Consumed I/O	BOOL	(43)
44	Get/Set	Watchdog Timeout Action	USINT	(44)
45	Get/Set	P&P Mode	UINT	(45)
46	Set	Delete All Timed Out Connections	USINT	(46)
47	Set	Confirm Peripheral Faults	USINT	(47)
48	Get/Set	I/O Size Word Align	USINT	(48)
49	Get/Set	BootP Operation	USINT	(49)

B 11.3 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 I/O connection to be changed. These values are retained in non-volatile memory and may only be set when the I/O connection is not in the RUNNING state.
- Values retained in non-volatile storage.

B 11.4 Values for Configuration Object Class Attributes

(3): Number of Digital Inputs

The Number of Digital Inputs attribute determines the number of input channels to be returned in the I/O produced packet. The number of I/O produced bytes can be calculated as:

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

(4): Number of Digital Outputs

The Number of Digital Outputs attribute determines the number of output bytes to be processed in the I/O consumed packet. The number of I/O consumed bytes can be calculated as:

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

(5): Number of Analog Inputs

The Number of Analog Inputs attribute determines the number of analog input channels returned in the I/O produced packet. Each analog input produces 2 bytes of data in the I/O produced data. The number of bytes may be calculated as:

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

(6): Number of Analog Outputs

The Number of Analog Outputs attribute determines the number of analog output channels. Each analog output consumes two bytes of data in the I/O 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 I/O connection.

(8): Accept New Configuration

The Accept New Configuration attribute will keep the current 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 the last configuration.

(10): Add All Mode

Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
	SCO	PCP	SF	AOPs	AIPs	DOPs	DIPs

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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 I/O produced data contains the Inline Status and the second byte contains the number of the first module in the stack with a fault. (Adds 2 bytes to produced size)

(12): Include DSUP

When set the first byte of the I/O produced data contains the Device Supervisor exception status byte. (Adds 1 byte to produced size)

(13):	**Special Function Modules**	When set the Inline EIP BK will put the process data for the Special function modules with the Data In Attribute set in the I/O consumed and I/O produced.
(14):	**Pad I/O**	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 consumed and/or produced size)
(15):	**Number of DIP Faults**	Selects the number of DIP Faults added to the I/O produced data. (Number of bytes = ((number of channels) + 7) / 8)
(16):	**Number of DOP Faults**	Selects the number of DOP Faults added to the I/O produced data. (Number of bytes = ((number of channels) + 7) / 8)
(17):	**Number of AIP Faults**	Selects the number of AIP Faults added to the I/O produced data. (Number of bytes = ((number of channels) + 7) / 8)
(18):	**Number of AOP Faults**	Selects the number of AOP Faults added to the I/O produced data. (Number of bytes = ((number of channels) + 7) / 8)
(19):	**Number of Special Function Faults**	Selects the number of Special Function Faults added to the I/O produced data. (Number of bytes = ((number of channels) + 7) / 8)
(20):	**Produced Size**	This attribute allows the user to determine the produced size of the device.
(21):	**Consumed Size**	This attribute allows the user to determine the consumed size of the device.
(22):	**Bytes Reserved for DIPs**	This attribute allows the user to reserve bytes in the I/O data for future expansion of actual Digital Input Points. The I/O data 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 I/O data for future expansion of actual Digital Output Points. The I/O data 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.
(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.) Default = 0
(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 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 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.

(42): Run/Idle in Produced I/O

This attribute allows the user to select if the 32 bit Run/Idle header is in the produced data 00 = No (default), 01 = Yes.

(43): Run/Idle In Consumed I/O

This attribute allows the user to select if the 32 bit Run/Idle header is in the consumed data 00 = No, 01 = Yes (default).

(44): Watchdog Timeout Action

Connection	State Interpretation
0	Timeout
1	Auto Delete (default)
2	Auto Reset
3	Deferred Delete

(45): P&P Mode

This attribute allows the user to select the Plug and Play mode.

2 = P&P MODE ENABLED with inputs ONLY

1 = P&P MODE ENABLED with inputs and outputs active

0 = P&P MODE DISABLED, the configuration must match last saved configuration

(46): Delete All Timed Out Connections

This attribute allows the user to delete all timed out connections.

(47): Confirm Peripheral Faults

This attribute allows the user to confirm all peripheral faults.

(48): I/O Size Word Align

This attribute will enable byte padding (if required) to keep the produced and consumed sizes to 16-bit boundaries.

1 = Pad byte (default)

0 = No pad byte

(49): BootP Operation

This attribute allows the user to select the BootP operation

0 = If BootP is enabled, the module issues BootP requests without interrupt until a valid IP configuration is received (default).

1 = If BootP is enabled, the module issues 3 BootP requests. If a valid IP configuration is not received it will then use the last stored IP configuration.

B 12 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 EIP BK DI8 DO4 2TX-PAC.

B 12.1 Inline Interface Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	63

B 12.2 Inline Interface Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 12.4)
4	Get	Inline Status	BYTE	(4)
5	Get	First Faulted Module	USINT	(5)
6	Get/Set	Max Retry	USINT	(6)
7	Get	Number of Modules	UINT	(7)
8	Get	Number of Bits	UINT	(8)
9	Get	Number of Bytes	UINT	(9)
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)
21	Get	Error History (most recent)	UINT	(21)
…	…	…	…	
30	Get	Error History (last saved)	UINT	(30)

B 12.3 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 12.4 Values for Inline Interface Object Class Attributes

(4): Inline Status

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EIP Connection Timeout	Fault Cycles	Inline Connection Error	Configuring	Module Change	Power Fault	Peripheral Fault	CRC

(5): First Faulted Module
Contains the number of the first module that is faulted 1 = Inline BK

(6): Max Retry
Sets the number of bad responses the IL EIP BK DI8 DO4 2TX-PAC will accept before flagging an error.
Default = 32

(7): Number of Modules
Displays the number of Inline modules that the IL EIP BK DI8 DO4 2TX-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.

(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 “(4): Inline Status” on page 123)

(16): Latched Faulted Module
Contains the number of the first module that was faulted during the last fault.
1 = Bus coupler

(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 End-point #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 EIP BK DI8 DO4 2TX-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	RES

(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

(21): Error History (most recent)

Contains the most recent logged error information.

(30): Error History (last saved)

Contains the last logged error information. Attributes 22 ... 29 contain the intermediate error log history.

B 13 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 EIP BK DI8 DO4 2TX-PAC.

B 13.1 Inline Module Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	(Number of Inline Modules)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	7

B 13.2 Inline Module Object, Instance 1.. (Number of Inline Modules) Attributes

Attribute	Access	Name	Type	Value (see B 13.4)
3	Get	Inline Module ID (Config)	UINT	(3)
4	Get	Inline Module ID (Current)	UINT	(4)
5	Get	CIP Class	USINT	(5)
6	Get	First CIP Instance	UINT	(6)
7	Get	Last CIP Instance	UINT	(7)

B 13.3 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 13.4 Values for Inline Module Object Class Attributes

(3):	**Inline Module ID (Config)**	Displays the Inline ID for the module as the unit was configured.
(4):	**Inline Module ID (Current)**	Displays the current Inline ID for the module.
(5):	**CIP Class**	Reflects the number of the CIP class that the module is mapped to (i.e. 8 = DIP, 9 = DOP, etc.).
(6):	**First CIP Instance**	Reflects the first instance of the CIP object that this module is mapped to.
(7):	**Last CIP Instance**	Reflects the last instance of the CIP object that this module is mapped to.

B 14 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 EIP BK DI8 DO4 2TX-PAC that do not map to any standard CIP Object.

B 14.1 Inline Special Function Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	(Number of Inline SFs)
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	7

B 14.2 Inline Special Function Object, Instance 1.. (Number of Inline Special Function Modules) Attributes

Attribute	Access	Name	Type	Value (see B 14.4)
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 I/O	BOOL	(7)

B 14.3 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 14.4 Values for Inline Special Function Object Class Attributes

(3): IN Data

Data returned from the Inline module to the IL EIP BK DI8 DO4 2TX-PAC. Data size is determined by the module.

(4): OUT Data

Data sent from the IL EIP BK DI8 DO4 2TX-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 I/O

When set, the IN Data is in the I/O produced and the OUT data is in the I/O consumed. This attribute affects the produced and consumed sizes of the IL EIP BK DI8 DO4 2TX-PAC and is therefore only settable when the I/O connections are not in the established state.

B 15 COS Mask Object (Class Code: 104_{dec}, 68_{hex})

The COS Mask Object allows the user control which bits in the produced (input) data cause a COS message to be generated.

B 15.1 COS Mask Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	2
2	Get	Max Object Instance	UINT	1
6	Get	Max Class Identifier	UINT	7
7	Get	Max Instance Attribute	UINT	18

B 15.2 COS Mask Object, Instance 1 Attributes

Attribute	Access	Name	Type	Value (see B 15.4)
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	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 15.3 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 15.4 Values for COS Mask Object Class Attributes

(3): Mask Value

Contains the actual array of bytes that is ANDed with the produced to determine whether to generate a COS message or not.

(4): Index

Points to specific byte in the COS Mask Value Attribute allowing the Byte Value Attribute to be 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

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	Inline Status	PCP	SF	AOPs	AIPs	DOPs	DIPs

I/Os

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
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 Faults

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 Faults

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 Faults

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 Faults

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 16 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 16.1 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 16.2 PCP Object, Instance 1.. (Number of PCP Modules) Attributes

Attribute	Access	Name	Type	Value (see B 16.4)
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 CIP 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 16.4)
21	Get/Set	Process Data in CIP I/O	BOOL	(21)
22	Get/Set	PCP Write Invoke ID	USINT	(22)
23	Get/Set	PCP Read Invoke ID	USINT	(23)

B 16.3 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 16.4 Values for PCP Object Class 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 maximum 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

This 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:

Byte 1	Service
	1 = Read PCP
	2 = Write PCP
	3 = Read PDU size
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

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 "(3): PDU Size" above).

(7): PCP Response

This 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

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 “(3): PDU Size” on page 133).

(8): PCP Module

This 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

This 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).

(10): PCP Write SubIndex

This 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).

(11): PCP Write Data

This 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.

(12): PCP Read Index

This 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)

(13): PCP Read SubIndex

This 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)

(14): PCP Read Data

This 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

This 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 I/O data without consuming an excess amount of bandwidth or to use CIP tools that do not support variable length explicit services.

(16):	**PCP Response Fragment**	This 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 I/O data without consuming an excess amount of bandwidth or to use CIP tools that do not support variable length explicit services.
(17):	**PCP Fragment in CIP I/O**	If set the PCP Response Fragment is added to the produced data of all the I/O connections and the PCP Request Fragment is added to the consumed data. This attribute affects the produced and the consumed 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 EIP BK DI8 DO4 2TX-PAC. Data size is determined by the I/O module.
(20):	**Process Data OUT**	Data sent from the IL EIP BK DI8 DO4 2TX-PAC to the Inline module. Data size is determined by the I/O module.
(21):	**Process Data IN CIP I/O**	If set the PCP Process Data IN is added to the produced data of all the I/O connections and the PCP Process Data OUT is added to the consumed data. This attribute affects the produced and the consumed sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.
(22):	**PCP Write Invoke ID**	This attribute defines what invoke ID is used for PCP writes. Default is 0.
(23):	**PCP Read Invoke ID**	This attribute defines what invoke ID is used for PCP reads. Default is 0.

B 17 Serial Communications Object (Class Code: 106_{dec}, $6A_{hex}$)

The Serial Communications Object allows the user to control and transfer serial data on RS232 and RS485 485/422 modules.

B 17.1 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 17.2 Serial Communications Object, Instance 1.. (Number of Serial Modules) Attributes

Attribute	Access	NV	Name	Type	Value (see B 17.4)
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)
20	Get/Set	NV	Output Type	USINT	(20)
21	Get/Set	NV	DTR Control	USINT	(21)

Attribute	Access	NV	Name	Type	Value (see B 17.4)
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 CIP I/O	BOOL	(31)
32	Get/Set	NV	Fragment Data IN CIP I/O	BOOL	(32)
33	Get/Set	NV	Serial Object Enable	BOOL	(33)

B 17.3 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 17.4 Values for Serial Communications Object Class 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

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Number of Received Characters (Mode Dependent)							

LSB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 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

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Reserved							

LSB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 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.

(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.

(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	**Bit**	**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

(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 RS-232 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)

(31): Status/Control IN CIP I/O

If set, the serial status word is added to the produced data of all the I/O connections and the serial control word is ADDed to the I/O consumed data. This attribute affects the produced and the consumed 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 CIP I/O

If set the Receive Data Fragment is ADDed to the produced data of all the CIP I/O connections and the Transmit Data Fragment is added to the I/O consumed data. This attribute affects the produced and the consumed sizes of the I/O connections, and is therefore only settable when there are no I/O connections in the established state.

(33): Serial Object Enable

If set, the Serial Object will handle the buffering of the serial data. Set this attribute to 0 to enable direct access of the serial modules using the PCP class instead of the SCO class.

B 18 Port Object Class Definition (Class Code: 244_{dec}, $F4_{hex}$)

B 18.1 Port Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1 (one CIP Port on BK)
3	Get	Number of Object Instances	UINT	1
8	Get	Entry Port	UINT	1 (only one port supported)
9	Get	All Ports	ARRAY OF STRUCT	00 00 00 00 04 00 02 00

B 18.2 Port Object Instance Attributes

Attribute	Access	NV	Name	Type	Value
1	Get	V	Port Type	UINT	04
2	Get	V	Port Number	UINT	02
3	Get	V	Port Object	UINT	02 00 20 f5 24 01
4	Get	V	Port Name	SHORT_STRING	IL EIP BK 10/100 EIP Port

B 18.3 Port Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
1	01	Yes	Yes	Get_Attribute_All
2	02	No	No	Set_Attribute_All
14	0E	Yes	Yes	Get_Attribute_Single

B 19 TCP/IP Interface Object (Class Code: 245_{dec}, $F5_{hex}$)

B 19.1 TCP/IP Interface Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	1
2	Get	Max Object Instance	UINT	1
3	Get	Number of Object Instances	UINT	1

B 19.2 TCP/IP Interface Object Instance Attributes

Attribute	Access	NV	Name	Type	Value (see B 19.4)
1	Get	V	Status	DWORD	(1)
2	Get	V	Configuration Compatibility	DWORD	(2)
3	Get/Set	NV	Configuration Control	DWORD	(3)
4	Get	V	Physical Link Object	Struct of:	00
			Path Size	UINT	
			Path	Padded EPATH	
5	Get	NV	Interface Configuration	Struct of:	Configuration-specific
			IP Address	UDINT	
			Network Mask	UDINT	
			Gateway Address	UDINT	
			Name Server	UDINT	
			Name Server 2	UDINT	
			Domain Name	STRING	
6	Get	NV	Host Name	STRING	Configuration-specific

B 19.3 TCP/IP Interface Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
1	01	Yes	Yes	Get_Attribute_All
14	0E	Yes	Yes	Get_Attribute_Single
16	10	No	Yes	Set_Attribute_Single

B 19.4 Values for TCP/IP Interface Object Class Attributes

(1): Status Indicates the status of the configuration attribute.

Bit	Called	Definition
0 ... 3	Interface Configuration Status	0 = Not Configured 1 = Valid Configuration 2 ...15 = Reserved
4 ... 31	Reserved	Reserved (set to 0)

(2): Configuration Capability The Configuration Capability indicates the devices support for optional network capabilities.

Bit	Called	Definition
0	BootP Client	1 (TRUE) indicates the device can receive its configuration via BootP.
1	DNS Client	1 (TRUE) shall indicate the device is capable of resolving host names by querying a DNS server.
2	DHCP Client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via DHCP.
3	DHCP-DNS Update	1 (TRUE) shall indicate the device is capable of sending its host name in the DHCP request.
4	Configuration Settable	1 (TRUE) shall indicate the Interface Configuration attribute is settable.
5 ... 31	Reserved	Reserved for future use and shall be set to zero.

(3): Configuration Control The configuration control attribute tells the device how to determine its network configuration:

Bit	Called	Definition
0 ... 3	Startup Configuration	0 = Use configuration stored in NV 1 = Use **BootP** 2 = for DHCP reserved 3 ... 15 = Reserved*
4	DNS Enable	If 1 the device will resolve host names by using DHCP.
5 ... 31	Reserved	Reserved for future use (set to 0)

*Currently, 0 and 1 are the only valid values for Attribute 3. Future revisions will include the value of 2.

B 20 Ethernet Link Object (Class Code: 246_{dec}, $F6_{hex}$)

B 20.1 Ethernet Link Object Class Attributes

Attribute	Access	Name	Type	Value
1	Get	Revision	UINT	3
2	Get	Max Object Instance	UINT	2
3	Get	Number Object Instances	UINT	2
6	Get	Max. Class Identifier	UINT	7
7	Get	Max. Instance Attribute	UINT	10

B 20.2 Link Object Instance Attributes

Attribute	Access	Name	Type	Value
1	Get	Interface Speed	UDINT	(1)
2	Get	Interface Flags	DWORD	(2)
3	Get	Physical Address	ARRAY OF 6 USINTS	(3)
7	Get	Interface Type	USINT	(7)
10	Get	Interface Label	SHORT_STRING	(10)

B 20.3 Ethernet Link Object Common Services

Service Code		Class	Instance	Service Name
dec	hex			
1	01	Yes	No	Get_Attribute_All
14	0E	Yes	Yes	Get_Attribute_Single

B 20.4 Values for Ethernet Link Interface Object Class Attributes

(1): Interface Speed

The Interface Speed attribute shall indicate whether the interface is running at 10 Mbps, 100 Mbps, 1 Gbps, etc. The scale of the attribute is in Mbps, so if the interface is running at 100 Mbps then the value of the Interface Speed attribute shall be 100. The Interface Speed is intended to represent the media bandwidth; the attribute shall not be doubled if the interface is running in full-duplex mode.

(2): Interface Flags

Bit	Called	Definition
0	Link Status	Indicates whether or not the Ethernet 802.3 communications interface is connected to an active network. 0 indicates an inactive link; 1 indicates an active link.
1	Half/Full Duplex	0 indicates the interface is running half duplex; 1 indicates full duplex. Note that if the Link Status flag is 0, then the value of the Half/Full Duplex flag is indeterminate.
2 ... 31	Reserved	Shall be set to zero

(3): Physical Address

The Physical Address attribute contains the interface's MAC layer address. The Physical Address is an array of octets. The recommended display format is "XX-XX-XX-XX-XX-XX", starting with the first octet. Note that the Physical Address is not a settable attribute. The Ethernet address shall be assigned by the manufacturer, and shall be unique per IEEE 802.3 requirements.

(7): Interface Type

The Interface Type attribute shall indicate the type of physical interface. Fixed at a value of 2 (Twisted Pair 10/100 base-T).

(10): Interface Label

Text String describing the interface.
Instance 1 = "IL EIP 10/100 Port-X1" Instance 2 = "IL EIP 10/100 Port-X2"

Please observe the following notes

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 user documentation) does not constitute any further duty on the part of Phoenix Contact to furnish information on modifications to products and/or technical documentation. You are responsible to verify the suitability and intended use of the products in your specific application, in particular with regard to observing the applicable standards and regulations. All information made available in the technical data is supplied without any accompanying guarantee, whether expressly mentioned, implied or tacitly assumed.

In general, the provisions of the current standard Terms and Conditions of Phoenix Contact apply exclusively, in particular as concerns any warranty liability.

Phoenix Contact reserves the right to register its own intellectual property rights for the product identifications of Phoenix Contact products that are used here. Registration of such intellectual property rights by third parties is prohibited.

Other product identifications may be afforded legal protection, even where they may not be indicated as such.

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:
phoenixcontact.com

Make sure you always use the latest documentation.
It can be downloaded at:
phoenixcontact.net/products

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 phoenixcontact.com.

Published by

PHOENIX CONTACT GmbH & Co. KG
Flachsmarktstraße 8
32825 Blomberg
GERMANY

PHOENIX CONTACT Development and Manufacturing, Inc.
586 Fulling Mill Road
Middletown, PA 17057
USA

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

SCATTERGOOD & JOHNSON LTD
ELECTRICAL ENGINEERING & FLUID CONTROL DISTRIBUTORS
Est.1899

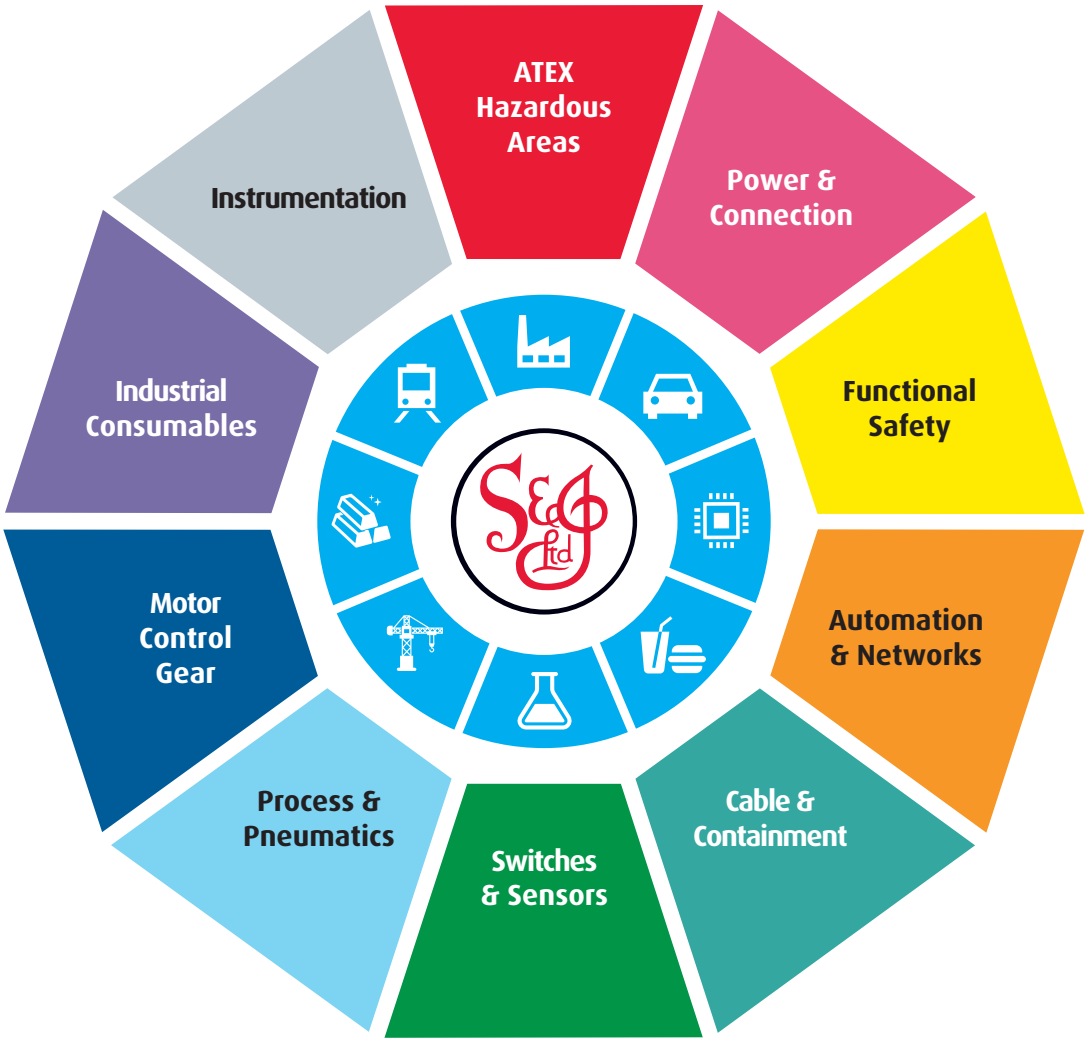
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

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