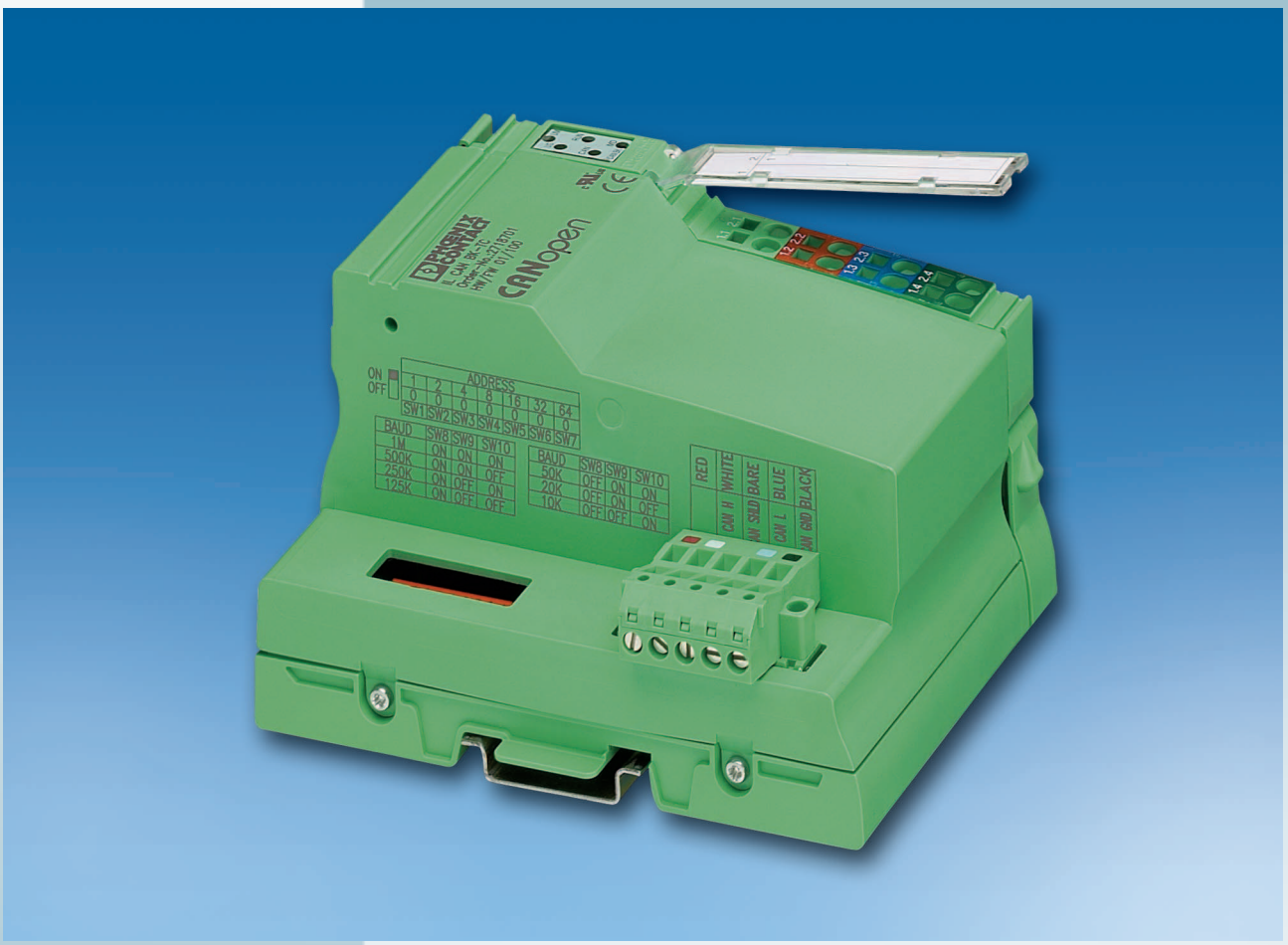
AUTOMATION



User manual

UM EN IL CAN BK-TC-PAC

Order No.: —

Inline bus coupler for CANopen
IL CAN BK-TC-PAC

PHOENIX CONTACT
INSPIRING INNOVATIONS

DOWNLOADED FROM WWW.SCATTS.CO.UK

AUTOMATION

User manual
Inline bus coupler for CANopen

03/2009

Designation: UM EN IL CAN BK-TC-PAC

Revision: 00

Order No.: —

This user manual is valid for:

Designation	Revision	Order No.
IL CAN BK-TC-PAC		2718701

Please observe the following notes

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

User group of this manual

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

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

Explanation of symbols used and signal words

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

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

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

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

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

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

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

General terms and conditions of use for technical documentation

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

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

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

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

Statement of legal authority

How to contact us

Internet

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

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

www.phoenixcontact.net/download

Subsidiaries

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

Published by

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

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

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

Table of contents

1 Documentation landscape for Inline Modular IO on CANopen

The documentation for Inline Modular IO is modular, providing you with the optimum information for your specific bus system or Inline terminal.

This documentation can be downloaded at www.phoenixcontact.net/download.

For a comprehensive list of the documentation, please refer to the ordering data (see Section "Ordering data" on page 6-4).

Terminal-specific documentation can be found in the download area for the corresponding device.

Make sure you always use the latest documentation.

The following documentation is available for the **CANopen** bus system in association with Inline Modular IO:

1.1 CANopen

"Inline bus coupler for CANopen IL CAN BK-TC-PAC" user manual, UM EN IL CAN BK-TC PAC

The manual provides a general introduction to the CANopen system including an overview of the topology.
It also describes the IL CAN BK-TC-PAC bus coupler with its hardware, as well as the supported CANopen and PCP interface objects.

1.2 Inline

"Automation terminals of the Inline product range" user manual, IL SYS INST UM E

This manual is the generic system manual for Inline and describes the use of terminals/modules in the Inline product range for all bus systems.

The following topics are covered:

- The device properties, which are the same for all bus systems
- Notes on the low voltage area
- Overview of the Inline product groups
- Structure and dimensions of Inline terminals
- Electrical potential and data routing
- Mounting and removing
- General technical data and ordering data
- Examples and tips

User manuals (special terminals)

The additional user manuals describe a special Inline terminal (e.g., counter terminal, positioning terminal).

Each manual only describes the relevant terminal-specific special features. As the higher-level manual, the IL SYS INST UM E user manual also applies.

"Configuration of a CANopen system with SyCon" quick start guide, UM QS EN IL CAN BK-TC-PAC

The quick start guide describes step-by-step the system startup with the SyCon software using an example.

Terminal-specific data sheets

The data sheet describes the specific properties of each Inline terminal.
These include at the very least:

- Function description
- Local diagnostic and status indicators
- Pin assignment/terminal point assignment and connection example
- Programming data/configuration data
- Technical data

"Summary of key data for Inline devices" data sheet, DB GB IB IL DEVICE LIST

This data sheet is also referred to as the device list.

In addition to terminal-specific data sheets, this data sheet also includes the key data of every Inline Modular IO device. This includes, for example:

- Programming data: ID code, length code, process data channel, I/O address area
- Error messages
- Power supply/current consumption

Application notes

Application notes provide additional information about special topics.
In conjunction with CANopen, these include, for example:

- Overview of I/O terminals that can be connected to various bus couplers — AH IL BK IO LIST

Device-specific application notes are listed in the device-specific data sheet and are available in the download area for the relevant device.

Package slips

A package slip contains key information for the electrical installation of an Inline terminal or group of Inline terminals. This includes, for example:

- Short description
- Safety notes
- Mounting/removal
- Terminal point assignment
- Local diagnostic and status indicators

2 Inline CANopen bus coupler

CANopen bus coupler

The CANopen bus coupler provides the interface between CANopen and the I/O modules of the Inline product range. It creates the bus signal configuration and the current supply required for the connected I/O modules.

The bus coupler supports all basic digital, analog and function modules of the Inline I/O terminal range.

Please refer to the AH IL BK IO LIST application note for an overview of all supported Inline terminals.

All hardware and software of this design complies with the CANopen specifications of the CiA (CAN in Automation) or the technical specifications of the Inline product family.

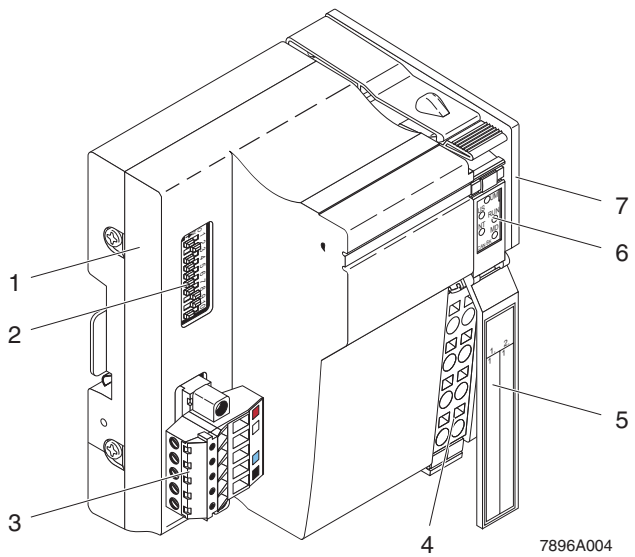


Figure 2-1 Inline CANopen bus coupler components

Key:

1 Electronics base
2 DIP switches to set address and baud rate
3 TWIN-COMBICON connector
4 Power connector
5 Labeling field
6 Diagnostic and status indicators
7 End plate

The electronic device data sheet (EDS) can be found on the Internet at www.phoenixcontact.net/download.

Features

The key features of the IL CAN BK-TC-PAC are listed below:

– Module features

- Extended diagnostics
- Diagnostic and status indicators
- Programmable fault response modes
- I/O configuration can be stored
- Auto-configuration

- Inline features

- Up to 63 Inline terminals can be connected
- Up to 510 digital input and output points
- Up to 254 analog input and output points
- Up to 63 intelligent, special function modules can be connected

- CANopen features

- Direct peer-to-peer communication
- Alarm messages
- Error states and values
- Settable baud rates: 10 kbaud, 20 kbaud, 50 kbaud, 125 kbaud, 250 kbaud, 500 kbaud, 1 Mbaud
- Trigger modes: Event, time, remote request
- Node guarding and heartbeat
- Analog interrupt triggers:
  - Upper limit, lower limit
  - Unsigned delta
- Supports two configuration devices simultaneously

Applications

Connection of sensors/actuators via CANopen.

3 Installation

3.1 The CANopen system

CANopen is a high-level network protocol based on the serial CAN bus. CANopen was originally designed for motion control applications in industrial control systems. Since then, CANopen has expanded into other on-board control applications such as that used for public transportation, off road vehicles, medical equipment, maritime electronics, and building automation.

CANopen has a linear structure. See Figure 3-1 on page 3-2.

There are two methods used to connect devices to the CANopen trunk line (bus line):

1. The first method uses two network connections that directly attach devices to the trunk line.
2. The second method uses T-connectors and branch lines to attach devices to the trunk line.

Only the CAN signal lines and a common CAN transceiver ground are routed in the network cable. CAN hardware is covered in ISO 11898. The CANopen protocol specifications are maintained by an international users and manufacturers group known as “CAN in Automation“ (CiA).

The Producer/Consumer model that is used for transferring Process Data Objects (PDOs) allows broadcasts of information and allows any device on the network to be configured to listen and act on the PDO transmissions of another device. A Client/Server model is used when configuring a device through Service Data Objects (SDOs). This method is necessary, because the client needs confirmation of the requests to the client.

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

3.2 Example topology of an CANopen system

CANopen has a linear bus topology, see Figure 3-1. A termination resistor is required at each end of the trunk line. Branch lines are permitted but should be as short as possible (0.3 m at 1 Mbaud).

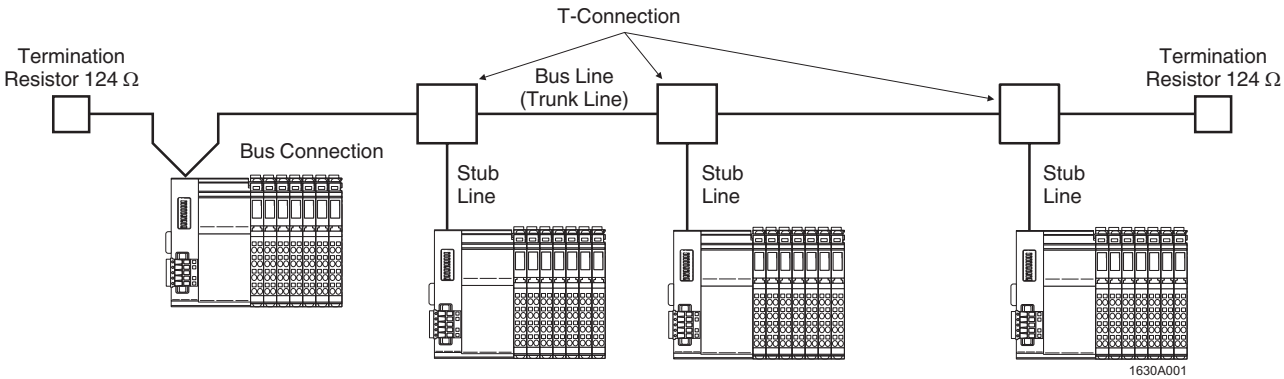


Figure 3-1 CANopen example topology

The total length of trunk and branch lines depends on the data rate and the type of cable. Use Table 3-1 and Table 3-2 as guidelines to determine the type of trunk line, length, termination resistors and number of devices. Refer to the CANopen specification to determine exact trunk and branch cable limits.

Trunk lines may be routed in parallel, twisted, parallel shielded, or twisted shielded depending on the EMC requirements. The wiring topology should be as close as possible to a single line structure in order to minimize reflections.

There is no electrical isolation between CANopen and the CAN interface of the IL CAN BK-TC-PAC. It is recommended that repeaters are used for networks that exceed 200 m.
To isolate the CAN interface from the Inline I/O modules a power terminal, powered by a separate 24 V power supply, must be used.

Table 3-1 Maximum trunk cable length vs. number of devices

Bus cable length (meters)	Cable resistance (mOhms/m)	Cable cross section (mm^2)	Termination resistor (Ohms)	Baud rate (kbits)
0 to 40	70	0.25 to 0.34	124	1000 at 40 m
40 to 300	< 60	0.34 to 0.60	150 to 300	≤ 500 at 100 m
300 to 600	< 40	0.50 to 0.60	150 to 300	≤ 100 at 500 m
600 to 1000	< 26	0.75 to 0.80	150 to 300	≤ 50 at 1 km

Table 3-2 Bus cable DC parameters for networks with less than 64 devices

	Maximum length in meters (0.2 safety margin)			Maximum length in meters (0.1 safety margin)		
Cable cross-section (mm^2)	**32 devices**	**64 devices**	**100 devices**	**32 devices**	**64 devices**	**100 devices**
0.25	200	170	150	230	200	170
0.50	360	310	270	420	360	320
0.75	550	470	410	640	550	480

Bus coupler

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

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

Tasks of the bus coupler:

– Coupling of I/O modules to the remote bus
– Supplying the I/O modules with communications power
– Electrical isolation of the local I/O
– Providing diagnostic information from the connected I/O to CANopen

Trunk cable

The trunk cable is used to connect devices to each other if no branch lines exist in the network. If branch lines are present, T-connectors will be linked using the trunk cable (see Figure 3-1 on page 3-2). The maximum number of CANopen devices in the CANopen network is limited to 127. The maximum length of a trunk cable ranges from 40 m to 1 km, depending on the baud rate. The total cable length can be increased through the use of repeaters.

Branch line

The branch line is used to connect devices to the trunk cable. The connection is made by using a T-connector. Ensure that the branch lines are as short as possible, especially at higher baud rates. At 1 Mbit, the maximum length of a branch line is 0.3 m.

Termination resistors

CANopen requires a termination resistor to be installed at each end of the trunk line. The typical resistor used to terminate a CANopen trunk line has a resistance of 124 Ω and a power dissipation of 0.25 W. Please refer to the CANopen specification DR303-1 to determine exact values and for additional information about termination resistors.

3.3 Inline station

The Inline product family is a modular automation system. Inline automation terminals are joined together to create functional units that meet the requirements of the application (see Figure 3-2).

The control cabinet layout is designed according to function. Both communication and power routing are 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 devices.

Please refer to the IL SYS INST UM E user manual for additional information on the Inline station structure.

Please refer to the AH IL BK IO LIST application note for an overview of Inline terminals that can be connected to the bus coupler.

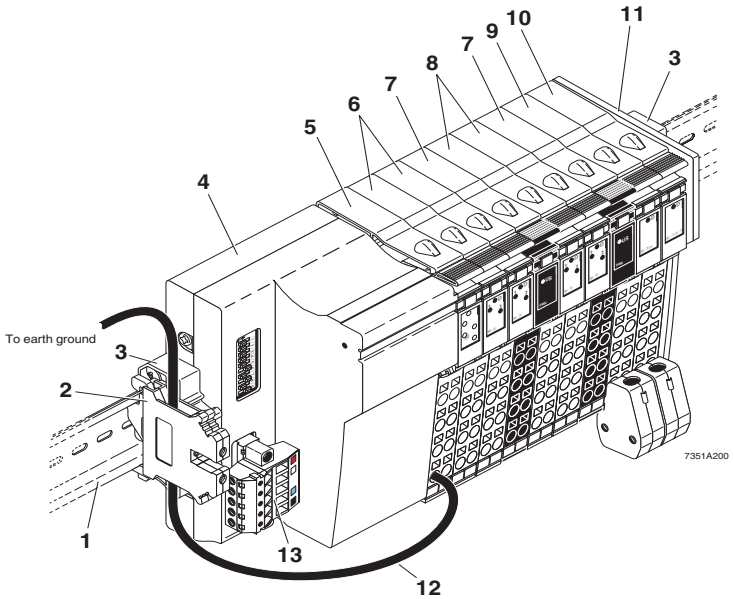


Figure 3-2 Typical Inline station with a CANopen bus coupler

Key:

1 DIN rail
2 Grounding terminal block
3 End clamp
4 Bus coupler
5 Power terminal of the bus coupler
6 Digital input module
7 Power terminal
8 Digital output module
9 Analog input module
10 Analog output module
11 End plate
12 Grounding wire 0.2 mm^2 to 1.5 mm^2
13 TWIN-COMBICON connector

3.4 Wiring the bus coupler

3.4.1 Terminal point assignment of the bus coupler

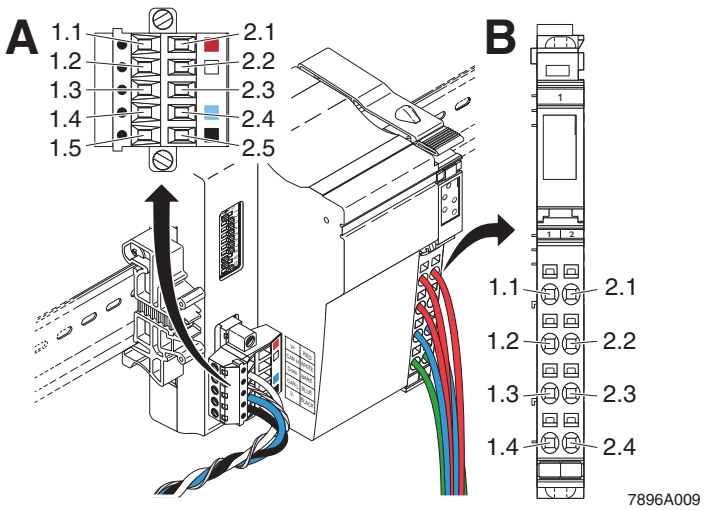


Figure 3-3 Connector numbering of the CANopen bus coupler
A: 5-pos. network connector, B: power connector

Table 3-3 Terminal assignment of the network connector

Terminal point	Labeling		Assignment	Color in the network cable
1.1, 2.1	V+	RED	No connection	
1.2, 2.2	CAN H	WHITE	CAN H	White
1.3, 2.3	Drain	BARE	CAN shield	Bare
1.4, 2.4	CAN L	BLUE	CAN L	Blue
1.5, 2.5	V-	BLACK	CAN ground	Black

When the bus coupler is used in environments with heavy noise, Phoenix Contact recommends the following: Install a grounding terminal block to the left of the bus coupler and connect the network cable shield to this terminal. In this case terminal point 1.3 of the network connector is not used.

Table 3-4 Terminal point assignment of the power connector

Terminal points	Assignment	Terminal points	Assignment
1.1	Segment circuit U_S (+24 V DC)	2.1	Main circuit U_M (+24 V DC)
1.2	Bus coupler supply U_{BK} (+24 V DC)	2.2	Main circuit U_M (+24 V DC)
1.3	GND U_{BK}	2.3	GND U_M, U_S
1.4	FE (Functional earth ground)	2.4	FE (Functional earth ground)

Figure 3-4 shows a wiring schematic for the power connector that includes internal connections. Note that the bus coupler provides I/O power to the main (U_M) and segment (U_S) circuits for the Inline station. Communications power and analog power are also supplied by the bus coupler through the U_{BK} connection.

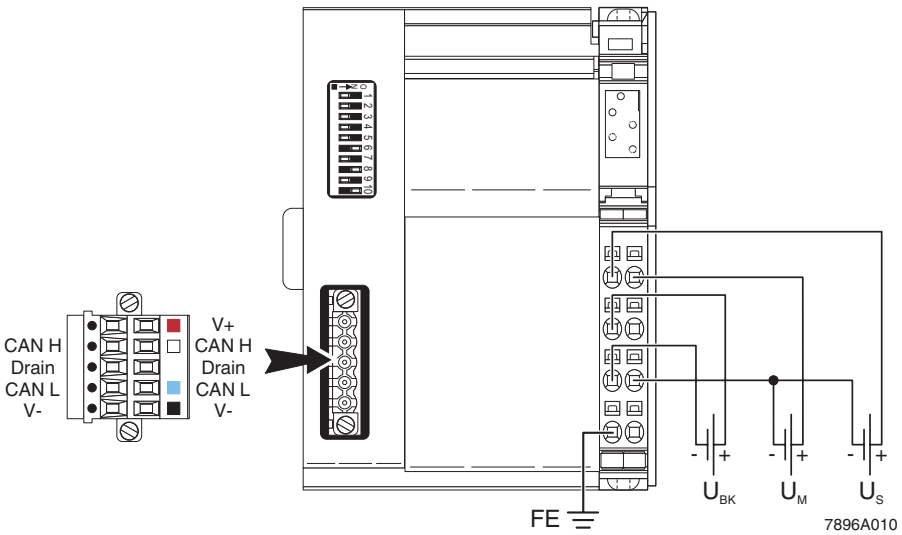


Figure 3-4 Typical bus coupler wiring

Terminal points 2.1 and 2.2 (U_M) are connected internally. Terminal points 1.4. and 2.4 (FE) are connected internally.

3.4.2 Wire type and strip-length requirements

The packing slip or the IL SYS INST UM E user manual explain how to proceed to wire the Inline connector.

For the bus coupler network connector, use a standard 5-wire CANopen cable. See Figure 3-5 for strip-length requirements. Figure 3-6 shows how to install or remove wires from the module.

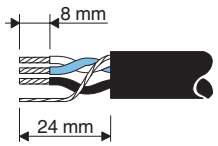


Figure 3-5 Strip-length requirements

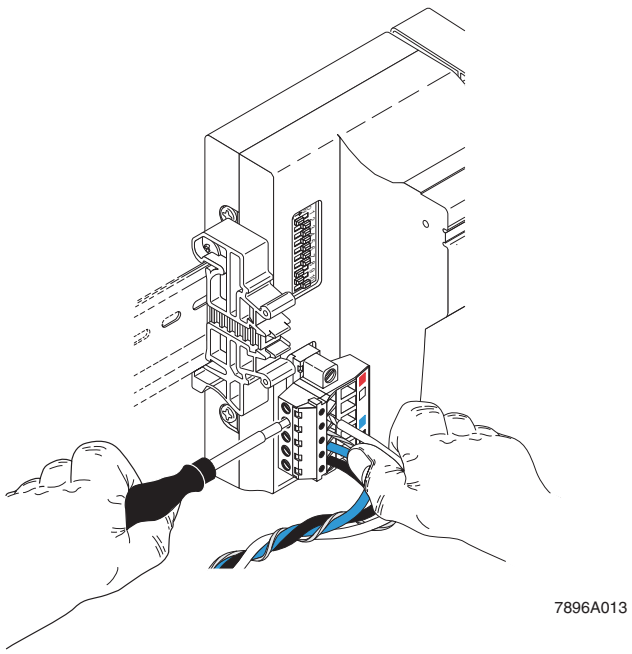


Figure 3-6 Installing and removing cables

3.5 Network considerations

3.5.1 Wiring

Figure 3-7 shows an example of setting up a standard bus cable and branch network using T-connections.

A. Termination resistors

A 124 Ω, 1% metal film, 0.25 W, termination resistor typically must be added at the extreme ends of the CANopen system (refer to CANopen specification DR303-1), see Figure 3-7, detail D.

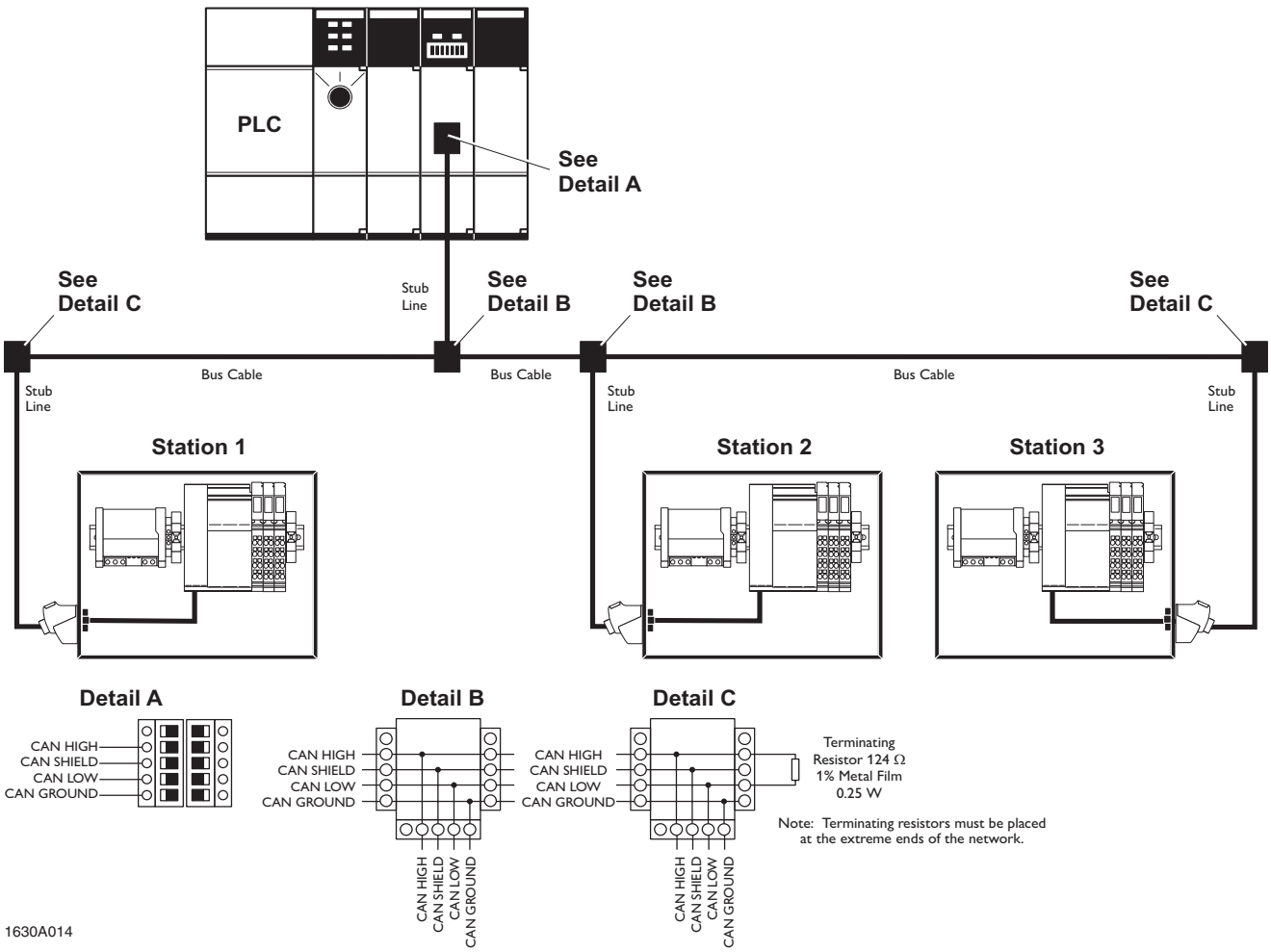


Figure 3-7 Example of setting up network communications and network power in an Inline CANopen system.

B. Daisy chaining

NOTE: Malfunction with incorrect connection sequence

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

The 5-pos. network connector contains two rows of contacts with 5 contacts in each row, see Figure 3-4 on page 3-6. These two rows of contacts provide a means for daisy chaining additional Inline stations, see Figure 3-8.

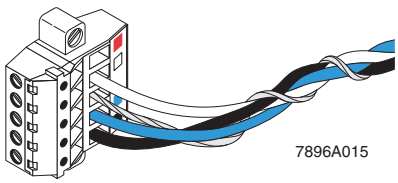


Figure 3-8 Termination of the bus cable

3.5.2 Wire and cable central grounding specifications

WARNING:
Grounding protects human beings and machines from dangerous voltages. To avoid dangers, it is essential that you always follow the grounding procedures outlined in the product-specific data sheets, this user manual and local codes.

CANopen devices must share a common ground with all CAN transceivers. It is important to locate the CANopen network power supply for all devices as close to the physical center of the network as possible. This will reduce the current in the ground line and decrease the overall ground voltage potential. Care must be taken that the ground potentials in the network do not exceed 2 V DC. If this cannot be achieved, electrical isolation will be required.

Figure 3-9 on page 3-10 shows an example of a system grounding scheme using a single power supply. Take note that the power supply's chassis is directly connected to earth ground.

NOTE: Malfunction through data telegrams
All CANopen devices must be grounded to avoid possible signal interference.

The Inline bus coupler is both an isolated physical layer and an I/O device. The only strictly CANopen grounding consideration is CAN Shield. CAN Shield can be terminated in position 1.3 or 2.3 of the network connector, see Figure 3-4 on page 3-6.

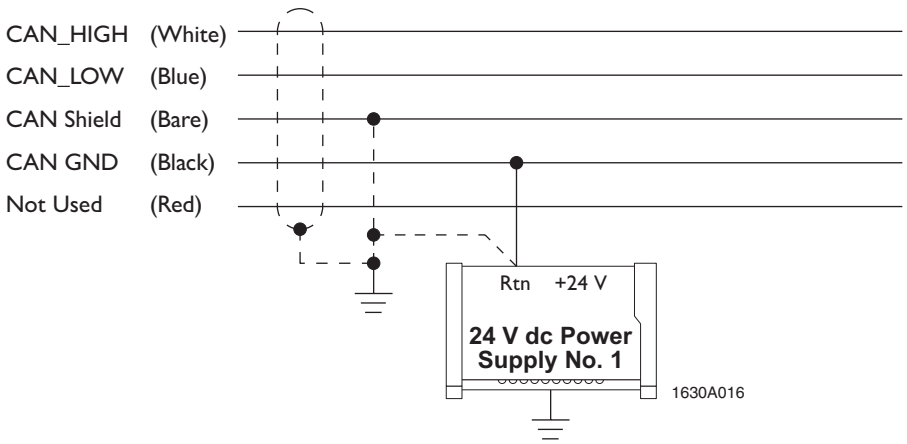


Figure 3-9 CANopen system grounding scheme

4 Operation and configuration

4.1 Introduction

This section provides the quickest path to start up the Inline CANopen bus coupler.

Each procedure in this section deals with information on how to install, operate and troubleshoot the bus coupler.
Information in this section is presented with the assumption that the user has a working knowledge of CANopen and the CANopen Configuration Studio™ by IXXAT (or an equivalent software configuration package).
By default, the bus coupler will be ready to exchange data as soon as it receives a Start Remote Node command.

When the CANopen bus coupler is first received it will be set up to address 1 and a baud rate of 20 K. The bus coupler is delivered to operate in fault response mode 1 (Auto Restart mode, Inline Fault Mode Object Index 3002_{hex}. In the Auto Restart mode all outputs will be turned off in the event of any local I/O failure, except for a peripheral fault (PF). The PF bit will be set when any output is shorted or during a loss of power to an intelligent segment module. The Auto Restart mode is described in more detail later in this section.

Once the failure has been repaired, the outputs will go to their previously programmed fault states/value (default is a "0").

By default, Inline station diagnostics includes the emergency object that is produced by the device during any fault. The PF bit is included in the emergency object and will need to be evaluated by the application to determine the required action for the local station outputs.

4.2 DIP switch settings for address and baud rate

Each bus coupler has 10 DIP switches. The DIP switches are located on the left side of the CANopen bus coupler, see Figure 4-1. DIP switches 1 through 7 are used to set the device address and DIP switches 8, 9 and 10 are used to set the baud rate.

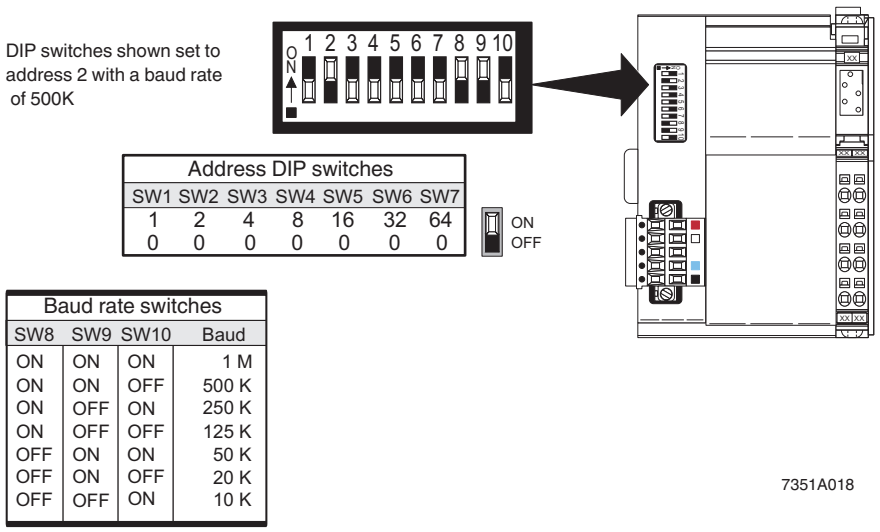


Figure 4-1 CANopen bus coupler DIP switches

4.2.1 Setting the device address

The device address is set using DIP switches 1 through 7. DIP switch 1 is the least significant digit of the device address and DIP switch 7 the most significant digit. Valid device address settings range from 1 to 127. Note that U_{BK} power will need to be cycled (turned OFF and ON) in order to implement any changes to the device address. Device address 0 is reserved and is used to auto-configure the I/O connected to the module. The module will not be switched on at address 0.

4.2.2 Setting the baud rate using DIP switches 8, 9, and 10

The baud rate is set using DIP switches 8, 9, and 10. DIP switch settings for various baud rates are shown in Figure 4-1.

4.3 Determining I/O module capacity

Without regard to the 32 input and 32 output transmit and receive process data objects (PDOs), the bus coupler is internally capable of the following number of subindexes for the specific point or channel types.

- Digital input points 510
- Digital output points 510
- Analog input points 254
- Analog output points 254
- PCP modules 8
- Serial modules 8
- Special function modules 63

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 2 A of communications/logic power (U_L, generated from U_{BK}).

If U_L requires more than 2 A, a IB IL 24 PWR IN/R-PAC terminal can be inserted to reinject U_L power.

2 The maximum number of I/O devices connected to the bus coupler cannot exceed 63.

3 Analog modules cannot draw more than 0.5 A from the analog supply (U_{ANA}). Note that analog modules also require current from the 2 A logic supply (U_L).

There are 2 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 AH EN IL DEVICE LIST application note to determine the amount of current draw required for each I/O module or device to be connected to the bus coupler. The total amount of current draw for all modules/devices cannot exceed the 2 A and 0.5 A ratings stated above.

4.4 Configuring an Inline station

4.4.1 Bus coupler

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

- 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 (subindexes) does each module contain
- Types of I/O modules:
  - Digital input
  - Digital output
  - Analog input
  - Analog output
  - PCP capable (AS-i, RS-232, RS-485, other)
  - Serial (RS-232 or RS-485)
  - Special function (incremental encoder, absolute encoder, high speed counter, other)

Any module that is not recognized as a digital, analog or PCP module will be placed into the special function category.

Configuration methods

An Inline station can only be configured automatically by the bus coupler (auto-configuration).

When creating, adding to, or changing an Inline CANopen 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

- Auto-configuration by setting the DIP switches (no software required)
- Auto-configuration by sending a Service Data Object (SDO) message

A. Auto-configuration by setting the DIP switches

This method allows for in-the-field configuration of the bus coupler without any software. The following default settings are used:

– Receive Process Data Object (PDO) 1 will contain the digital outputs from 1 to 64.
– Transmit PDO 1 will contain digital inputs from 1 to 64.
– Receive PDO 2 will contain analog outputs from 1 to 4.
– Transmit PDO 2 will contain analog inputs from 1 to 4.
– Receive PDO 3 will contain analog outputs from 5 to 8.
– Transmit PDO 3 will contain analog inputs from 5 to 8.
– Receive PDO 4 will contain analog outputs from 9 to 12.
– Transmit PDO 4 will contain analog inputs from 9 to 12.
– All other I/O such as special function modules must be manually mapped.

For auto-configuration, proceed as follows:

- Set DIP switches 1 to 7 of the device address to 0.
- Power up the bus coupler with U_{BK}, U_S, and U_M supplies and the connected I/O modules. If the RUN LED on the bus coupler is permanently on (green), the I/O configuration is stored in the station flash memory.
- Switch off the bus coupler.

The module will not be switched on with the address set to 0.

- Set the device address and baud rate switches to the desired settings.
- Switch on the bus coupler.

B. Auto-configuration by sending a Service Data Object (SDO)

Configuration in this manner can be done by sending an SDO to the Reconfigure I/O Object, Index 3000_{hex}. The bus coupler will scan its local bus and reset all temporary variables that relate to the current configuration with the Reconfigure I/O Object, Index 3000_{hex} set to a "1". This configuration can be transferred into non-volatile flash memory by a save command to the Store Parameters Object, Index 1010_{hex}. This configuration will remain valid until the next Reconfigure I/O command is sent or the bus coupler is reconfigured in another way.

The CANopen Configuration Studio™ or an equivalent software package can be used to send SDO messages.

4.4.2 Analog input (AI) modules

4.4.2.1 General configuration

To change the settings of an analog input channel, the corresponding process data output word must be accessed through the Analog Input Range Object, Index 2400_{hex}. This object index not only changes the range, it also allows for configuration of any of the configurable parameters of an analog module.

A. Configuration example

See Figure 4-2. This example will be used to explain how to reconfigure the range of the first AI 2 module (channel 2) to the bipolar 10 V range and to change the format to "IB ST" (12 bits).

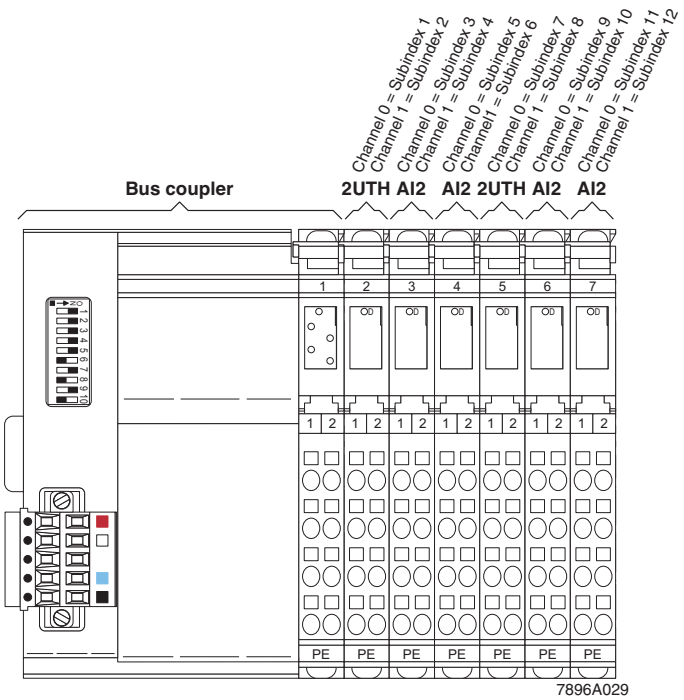


Figure 4-2 I/O station with analog input modules and thermocouple modules

The AI 2, UTH and RTD modules are all 2-channel devices consisting of 16 bits for each channel. Channels for these modules will occupy subindexes in the Analog Input Object(s) indexes.
Each channel has a corresponding output words that will be made available to the user for configuration of the channel in the Analog Input Range Object Index 2400_{hex}.

1 Determine which subindex channel 2 of the first AI 2 module (second I/O module) resides in. In this case the second channel of the first AI 2 module will occupy subindex 4 of the 16-bit Analog Input Object (6401_{hex}). In Figure 4-2, the UTH module occupies subindexes 1 and 2, and channel 1 if the first AI 2 module occupies subindex 3.
If the user is having difficulty locating the specific index for the object or subindex for the output word, an SDO can be sent to the following object with the following indexes:

(a) **Out Data COP Object, Index 3205_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the object index that the specific channel output data is mapped to.

(b) **Out Data COP First Subindex Object, Index 3206_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the first subindex that a module's output data is mapped to.

(c) **Out Data COP Last Subindex Object, Index 3207_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the last subindex that a module's output data is mapped to.

If the user is having difficulty locating the specific index or subindex for the actual input channel, an SDO can be sent to the following object with the following indexes:

(a) **In Data COP Object, Index 3202_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the object index that the specific channel input data is mapped to.

(b) **In Data COP First Subindex Object, Index 3203_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the first subindex that a module's input data is mapped to.

(c) **In Data COP Last Subindex Object, Index 3204_{hex}** — Sending an SDO read to the subindex identified by the module number. (The position on the local bus is determined by counting from left to right starting with the bus coupler being assigned to a "0"). This index will provide the number of the last subindex that a module's input data is mapped to.

2 Refer to the module-specific data sheet to determine what value needs to be sent to the Analog Input Range Object, Index 2400_{hex}. For this example the values are as follows:

Range	(bits 3 - 0)	Default = 0 V to 10 V	Bit code 0000
Format	(bits 5 - 4)	Default = Inline 15 bits	Bit code 00
Filter	(bits 9 - 8)	Default = 16-fold average	Bit code 00

– Bits 3-0 must be set to 0001 (bipolar 10 V range).
– Bits 5-4 must be set to 01 (IB ST format).
– Bits 7-6 are set to 00 (bits are unused).
– Bits 9-6 are set to 00 (default filter setting).
– Bits 14-10 are set to 00000 (bits are unused).
– Bit 15 must be set to a 1 (signifies that the data is for reconfiguration).
– The value to be sent is 8011_{hex}.

3 Send 8011_{hex} to Analog Input Range Object, Index 2400_{hex}, Subindex 4 using a Service Data Object (SDO) message. This message can be sent using a configuration tool such as the CANopen Configuration Studio™. Once the range is configured for a new value, the bus coupler will retain that sending in non-volatile flash memory.

"Appendix: CANopen objects and indexes" on page B-1 provides details of the Analog Input Range Object, Index 2400_{hex}.

4.4.3 Thermocouple and RTD modules

4.4.3.1 General configuration

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

There is no "Thermocouple" or "RTD" object. To change the default settings, the Analog Input Point Object, Index 2400_{hex} must be used. For more information refer to "Appendix: CANopen objects and indexes" on page B-1 and the Analog Input information in this section.

Default settings for the thermocouple modules are:

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

In order to change default settings, refer to the thermocouple data sheet to determine the appropriate attribute settings for the Analog Input Range Object, Index 2400_{hex}.

Keep in mind that thermocouple or RTD subindexes will appear as analog input subindexes. There are 2 subindexes for every thermocouple or RTD module.

The user must keep track of which subindexes are analog inputs and which subindexes are thermocouple inputs or RTD modules (not shown in Figure 4-2 on page 4-6). Figure 4-2 shows a station where subindexes 1 and 2 of the Analog Input Object, Index 6401_{hex} are used by the 2-channel thermocouple. Subindexes 3, 4, 5 and 6 of the Analog Input Object are used by the next two, 2-channel analog modules. Subindexes 7 and 8 of the Analog Input Object are used by the next 2 channel thermocouple module. The last two, 2-channel analog input modules occupy subindexes 9, 10, 11 and 12.

Default settings are modified by sending a Service Data Object (SDO) message to a specific subindex. This message can be sent using the CANopen Configuration Studio™.

Once the new thermocouple or RTD setting is made, the new configuration will be stored in flash memory of the bus coupler.

Analog Input Range Object, Index 2400_{hex} is described in "Appendix: CANopen objects and indexes" on page B-1.

4.4.4 Special function modules

4.4.4.1 General configuration

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

The special function module's input and output data can be mapped to the Process Data Object (PDO) of the user's choice. Special function PDO data is not mapped by default.

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.

4.4.5 Analog outputs

4.4.5.1 General configuration

For analog output modules that support configurable parameters, such as the two channel IB IL AO 2/U/BP-PAC module, it will be necessary to gain access to the AOP Response Data Object, Index 2410_{hex} to verify each command that needs to be sent to configure the module.

The IB IL AO 2/U/BP-PAC requires a sequence of commands sent to the proper subindex containing the output word for the AO 2. These output words reside in the 16-bit Write Analog Output Object, Index 6411_{hex}.

After each command in the sequence is sent, a positive or negative response to the command needs to be checked in the proper subindex in the AOP Response Data Object. Once the sequence is completed and all responses have been verified positive, the module is ready to accept normal analog output data using the 16-bit Write Analog Output Object.

Only analog output modules that can be configured and have associated input words, such as the IB IL AO 2/U/BP-PAC, will occupy subindexes in the AOP Response Data Object, Index 2410_{hex}. Each channel for the IB IL AO 2/U/BP-PAC module will occupy 1 subindex in the AOP Response Data Object. Subindexes are assigned by location on the local bus with the module(s) located closest to the bus coupler being assigned the first subindex(es).
Analog output modules that do not require configuration will not occupy subindexes in this object. Refer to the module specific data sheet for programming information.

4.4.6 Quick start using the CANopen Configuration Studio™ (CCS)

Use the following quick start procedure to configure the CANopen bus terminal.

1 Start CCS

2 Open a project

a. Click on New Project

b. Select the folder for the project to be stored

c. Enter the project name

d. Confirm the settings

3 Set the network baud rate

a. Select the network in the project window.

b. Right click

c. Select Properties

4 Insert devices

a. Insert a group for the new device.

- Select the network from the project window
- Right click
- Select New Group
- Enter name of group
- Click OK.

b. Select the group that was just added

c. Right click

d. Select New Device

e. Enter a name for the device e.g., CAN bus coupler

f Select Node Address

g Click on Select EDS

h. Use dialog box to locate correct EDS file.

i. Click OK

5 Select the network from the project window.

6 Click on CAN Bus Access

7 Select Upload from Device(s)

8 Minimize CANopen Network Access window

9 Click on devices

10 Click on Object Dict. Browser

11 You can now view and modify all of the objects in the object dictionary that can be found in Appendix A. To change the values in a module, go back to the CANopen Network Access window and select Download to Device(s).

Further details for linking Process Data Objects (PDOs) etc. can be found in the IXXAT CANopen Configuration Studio™ Manual.

4.5 Understanding I/O memory mapping

4.5.1 Bus coupler

The bus coupler automatically maps its I/O to Process Data Objects (PDOs) according to the CANopen specification DS401 V2.0. This procedure is explained in Section "Configuring an Inline station" on page 4-4. The user can map the PDOs to a PLC or to another node by using such tools as the Visual Object Linker, that is part of the IXXAT CANopen Configuration Studio™.

If the configuration software uses the EDS file, ESD is the basis that the memory mapping of the master agrees with the automatic mapping of the bus coupler.

4.5.2 The Electronic Data Sheet (EDS) file method

The EDS file is the software interface between the bus coupler and a CANopen configuration software package such as CANopen Configuration Studio™. The EDS file describes the Object Dictionary entries that the module supports and allows the software package to display the information to the user in a readable form.

The Reconfigure I/O Object, Index 3000_{hex}, can be utilized for newly configured or changed configurations, if the I/O modules are to be included in Process Data Objects.

The bus coupler will scan its local bus and reset all temporary variables that relate to the current configuration with the Reconfigure I/O Object, Index 3000_{hex} set to a "1". This configuration can be transferred into non-volatile flash memory by a save command to the Store Parameters Object, Index 1010_{hex}. This configuration will remain valid until the next Reconfigure I/O command is sent or the bus coupler is reconfigured in another way.

An explanation of each of these indexes can be found in "Appendix: CANopen objects and indexes" on page B-1.

4.6 I/O data transfer

A detailed explanation of object indexes can be found in "Appendix: CANopen objects and indexes" on page B-1.

I/O data transfer can be accomplished by setting the bus coupler to the operational state. This is done by sending a Start_Remote_Node command to the device. Once this is accomplished, the device starts consuming and producing the Process Data Objects (PDOs) that are configured for operation by the user or the bus coupler itself. Keep in mind that data will be sent to and received from the bus coupler in the low-byte/high-byte format.

4.6.1 Communications

The bus coupler can communicate over CANopen through the use of several communication object types. The following types are available to the user:

– Process Data Objects (PDO)
– Service Data Objects (SDO)
– Synchronization Objects
– Emergency Objects
– Network Management Objects (Not described in this user manual)

The bus coupler can transfer I/O data over the following CANopen communication objects:

– 32 Receive PDOs
– 32 Transmit PDOs
– 2 Server SDOs

A. Process Data Objects (PDO)

The process data object is used to transfer I/O data to and from the bus coupler. The message length is limited to 8 bytes of data. Therefore, multiple PDOs have to be used when transferring more than 8 bytes. PDOs can be synchronous or asynchronous. Synchronous PDOs are transmitted within a predefined time period after receiving a SYNC message from a SYNC producer. PDOs can be triggered by three different modes: event driven, timer driven and remotely requested. The default mode for the bus coupler's transmit PDOs is asynchronously triggered by inputs, see "Appendix: CANopen objects and indexes" on page B-1. This means that whenever any of the digital inputs change, PDO 1 is transmitted. By default the Analog Input interrupts are disabled, so the user will either have to enable the interrupt method of choice or set the event timer to allow the default Analog Input TPDOs (2-4) to be transmitted. Analog Input interrupts or the event timer will trigger any of the 1 to 32 TDPOs that have analog inputs mapped.

B. Service Data Objects (SDO)

Service Data Objects are used to configure the CANopen bus coupler. They can also be used to transfer more than 8 bytes of data. The bus coupler provides two separate SDO server objects, allowing multiple masters to simultaneously communicate with the device using SDOs.

C: Synchronization Objects

The bus coupler provides a synchronization object that allows the user to synchronize when the data is sampled and transmitted from each device on the network. Each device can be programmed with a time interval that the device waits for after receiving the SYNC object and before sending the PDO.

D: Emergency Objects

The bus coupler supports an Emergency Object, which allows the device to relay its status change immediately when a fault occurs or is cleared. This saves bandwidth by not having to repeatedly send its status over and over as part of the process data.

4.6.2 Standard digital, analog and special function object indexes

Complete details of the following indexes can be found in "Appendix: CANopen objects and indexes" on page B-1.

A. Digital Input Modules Object, Index 6000_{hex} - $61FF_{hex}$

Object Indexes 6000_{hex} - $61FF_{hex}$ provide access to digital input data and associated parameters. Access is provided in the form of groups of 8, groups of 16, and groups of 32. There is a separate subindex for each group of digital input points available on the device. Example Object Index 6000_{hex} provides access to groups of 8 digital inputs. Reading subindex 0 will tell the user how many groups of 8 are installed on the bus coupler. Reading subindex 1 will be the input states of inputs 1-8, reading subindex 2 will be the states of inputs 9-16, etc. See Appendix A for more information.

(1) Setting inputs to latch

Each digital input point can be independently configured to latch on a desired state. This is accomplished by using Digital Input Latch Enable Object, Index 2000_{hex}, and Digital Latch State Object, Index 2001_{hex}. Figure 4-3 shows how "actual input" or "latched" data is selected. Enabling the latch and the desired latch state must be done by sending an SDO.

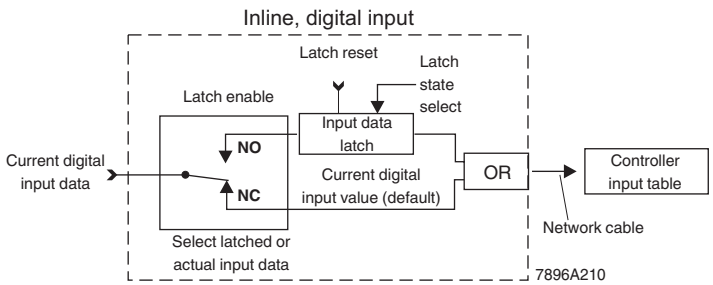


Figure 4-3 Latched or current digital input data

Digital Input Latch Enable Object, Index 2000_{hex} accesses groups of 8 inputs. Each bit in the corresponding subindex will enable the latching functionality when set to a logic "1" and will disable functionality when it is set to a logic "0". Object Index 2000_{hex} is set to "0" by default which disables the latching mode.

If the Digital Latch Enable Object is enabled by being set to a "1", then the user must select the desired latch level. This level can be set in Digital Latch State Object, Index 2001_{hex} of the specific subindex. If the Digital Latch State is set to a "0", then the input subindex will latch on a logic "0". If the Digital Latch State is set to a "1" (default), then the input instance will latch on a logic "1".

(2) Resetting latched inputs

Resetting latched inputs must be done by setting bit 1 in the Inline Control Byte Object, Index 3111_{hex}. However, setting bit 1 will clear all latched inputs. After the latches are reset, bit 1 in the Inline control byte must be set back to zero to allow for the next latched condition to occur when the control byte is mapped in a PDO.

By default the Inline control byte is not included in a PDO. The user can map the control byte to the PDO of choice.

If the user does not want to clear the latches using PDOs, then the SDO message to the Inline Interface Control Byte Object, Index 3111_{hex} can be sent. Setting the Inline Interface Control Byte to a value of "2" will reset all latches enabling the next input latch. It will also automatically reset the object value to a "0".

The actual value of the "latch" is stored in the bus coupler's memory. By default the latch state is 1. This means that the default latched value in the bus coupler's memory is "0". If the user changes the latch state to a "0", then the internal latch value must be re-initialized by either of the following two methods described below.
Both of these methods will initialize the internal latch value required for the specific latch state (determined by the latch value) and readies the latching function to become active once the latch enable is issued.

1 Store configuration settings. Then cycle power.

2 Reset latches using Object Index 3111_{hex}, control byte bit 1.

B. Digital Output Modules Object, Index 6200_{hex} - $63FF_{hex}$

Object Indexes 6200_{hex} - $63FF_{hex}$ provide access to digital output data and configuration. Access is provided in the form of groups of 8, groups of 16, and groups of 32 outputs. There is a separate subindex for each group of digital output points available on the device. Example Object Index 6200_{hex} provides access to groups of 8 digital outputs. Reading subindex 0 will tell the user how many groups of 8 are installed on the bus coupler. Writing subindex 1 will set the output states of outputs 1-8. Writing subindex 2 will set the output states of outputs 9-16. See Appendix A for more information.

C. Analog Input Modules Object, Index 6400_{hex} - 6601_{hex}

The Analog Input Objects represent the analog inputs in the CANopen bus coupler. There is a separate subindex for each group of analog input points available on the device. Index 6400_{hex} provides 8-bit access and Object Index 6401_{hex} provides 16-bit access.

By default analog inputs are included in the Transmit Process Data Objects (TPDO) 2, 3 and 4. However, they will not be updated by default. Analog inputs can be enabled by:

– Setting the TPDO event timer to periodically update analog inputs.
– Selecting a trigger interrupt, configure the trigger interrupt, then setting the Analog Input Global Interrupt Object, Index 6423_{hex}. Keep in mind that an adjustment of the transmit PDO inhibit time may be required to minimize network traffic.

(1) Setting a trigger interrupt type

Selecting a trigger interrupt type is accomplished in the Analog Input Interrupt Trigger Selection Object, Index 6421_{hex}. Trigger interrupt types are listed below:

– Upper limit exceeded (bit 0)
– Input below lower limit (bit 1)
– Input changed by more than delta (bit 2)

(2) Configuring a trigger interrupt type

Once the interrupt trigger type is selected the type must be configured. The three possible types used for configuring are:

– Analog Input Interrupt Upper Limit Integer Object, Index 6424_{hex}
– Analog Input Interrupt Lower Limit Integer Object, Index 6425_{hex}
– Analog Input Interrupt Delta Unsigned Object, Index 6426_{hex}

D. Analog Output Modules Object, Index 6410_{hex} - 6411_{hex}

The Analog Output Objects represent the analog outputs in the CANopen bus coupler. There is a separate subindex for each group of analog output points available on the device. Index 6410_{hex} provides 8-bit access and Object Index 6411_{hex} provides 16-bit access.

Analog output subindexes support standard CANopen error states and values. The error states and values can be used to control the analog output during a network or station fault. Each channel can be programmed separately to react in one of the two ways listed below. Refer to Section "Inline fault control" on page 4-17 to select proper fault response modes. "Appendix: CANopen objects and indexes" on page B-1 provides more analog output error state and value information.

1 Hold last value

2 Set to the value determined by the error value object.

E. Inline Special Function Object, Index 3300_{hex} - $330D_{hex}$

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 CANopen object.

– Incremental encoder
– Absolute encoder
– High-speed counter

F. PCP Modules Object, Index 3400_{hex} - 3411_{hex}

The PCP module objects allow the user to send data to, and receive data from modules that have PCP capability. There is a separate subindex for each PCP module attached to the bus coupler. The CANopen bus coupler supports a total of 8 PCP modules.

G. Serial Modules Object, Index 3400_{hex} - 3411_{hex}

Serial module objects give the users special access to serial modules such as the RS-232 and RS-485 modules. Each serial module attached to the bus coupler has a separate instance.

4.7 Inline fault control

4.7.1 Inline Fault Control Objects overview

Objects 1029_{hex} and 3002_{hex} to $300A_{hex}$ allow the user to control how the outputs behave during various faults.

Object 1029_{hex} controls what mode the module will be put in during a communication error. Some examples of communication errors are "bus-off" conditions, receiving PDOs with less data than expected, CAN buffer overruns or Node Guards/Heartbeat timeouts.

Object 3002_{hex} allows the user to control how the bus coupler attempts to control outputs when either a Connection Error occurs or a Module Change Error occurs. The four modes are defined in the paragraphs to follow.

4.7.2 Fault response modes

Fault response is a configurable setting in the Fault Mode Object, Index 3002_{hex}, that will control the behavior of the Inline station in the event that one of the three errors listed below occurs.

All Inline Stations Status fault bits are described in Section 5, "Diagnostics".

1 **Inline Module Change Fault Object, Index 3006_{hex}.** Represented in bit 3 of the Inline station status low byte (Object Index 3101_{hex}). Status of this fault can also be observed in byte 3 (bit 3) of the emergency telegram.

2 **Inline Inactive Local Bus Fault Object, Index 3007_{hex}.** Represented in bit 4 of the Inline station status low byte (Object Index 3101_{hex}). Status of this fault can also be observed in byte 3 (bit 4) of the emergency telegram.

3 **Inline Module Connection Fault Object, Index 3008_{hex}.** Represented in bit 5 of the Inline station status low byte (Object Index 3101_{hex}). Status of this fault can also be observed in byte 3 (bit 5) of the emergency telegram.

A. Inline fault mode 0: "Stop Running Data Cycles on Fault

Outputs are turned OFF automatically when a local failure occurs. Once the failure(s) is(are) fixed, one of the following must occur to bring the node back online:

– Inline Restart Local Bus (Index 3111_{hex}, Subindex 0, bit 2)
– Send Reset Node command
– Change CANopen state back to operational/preoperational

B. Inline fault mode 1 — Auto-Restart (default)

Outputs are turned OFF automatically when a local failure occurs. Once the failure is fixed, the station will handle I/O according to the operating state determined by the fault control object indexes (3003_{hex} to 3009_{hex}).

In the event the bus coupler is unable to auto-recover the local bus, the user may still be able to recover the bus remotely. This can be accomplished by changing the state from stop-to-preoperational, preoperational-to-operational, or by issuing a Restart Bus command (Object 3111_{hex}, bit 2), or Reset Node command. If the bus is still not be recovered, the user should power down the module and check the connections between all modules.

C. Inline fault mode 2 — Goto Fault State

Outputs are turned OFF automatically for up to two seconds when a local fault occurs. After two seconds the station will set the outputs to the preprogrammed CANopen fault states (default is OFF). The station will not recover its lost I/O even after the fault condition has been cleared. A reconfiguration or power cycle (U_{BK}) must be performed. The bus coupler will continue to run preprogrammed fault data event if the state is returned to operational. The module will continue to run preprogrammed fault data (the Fault Cycles bit is set in the Inline Station Status Byte Object, Index 3101_{hex} until a command is issued to restart the local bus. Once the local bus is restarted, new I/O data will update the outputs.

D. Fault response mode 3: Continue on Fault

Outputs are turned OFF automatically for up to two seconds when a local fault occurs. After two seconds the station will continue to update all I/O that is still online. Output data used is defined by Fault Mode Objects, Indexes 3002_{hex} to $300A_{hex}$. Output data can be either fault values or process data. The bus coupler continues to run in this module until a command is issued to restart the local bus. The station will not recover its lost I/O even after the fault condition has been cleared. A reconfiguration or power cycle (U_{BK}) must be performed.

4.7.3 Changing the fault response mode

Fault response mode is a flash memory setting in the bus coupler. A fault response mode cannot be changed by reconfiguring the I/O station or by cycling power (U_{BK}).
Changing the fault response from the default (mode 1) to a different mode can be accomplished by sending an SDO to the Object, Index 3002_{hex} (see "Appendix: CANopen objects and indexes" on page B-1).

4.7.4 Fault Control Object, Indexes 3003_{hex} to $300A_{hex}$

Object indexes 3003_{hex} to 3009_{hex} allow the user to control the state that the node is put into during a local bus fault. Fault control is activated when the corresponding bit in the Inline Status is set. Inline Status is part of the Station Status Object, Index 3101_{hex}. If any fault mode causes the state of the node to change, output data will go to fault values and the Fault Cycles bit is set in the Inline Station Status Byte Object, Index 3101_{hex}.

Fault modes can only activate from the operational state. The activation of the fault mode may or may not cause a state change. By default, Object Indexes 3003_{hex} to $300A_{hex}$ (except for the Peripheral Fault Mode Object, Index 3004_{hex} will activate the go to preoperational state. Peripheral Fault Mode defaults to "No state change".

To re-enter the operational state from a state change, as determined by Object Indexes 3003_{hex} to 3009_{hex}, the user can issue a Start Remote Node command. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be used instead of fault values.

If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can go to the preoperational state and then use the Inline Interface Control Byte Object (Index 3111_{hex}) to clear the latched peripheral fault. This is assuming that the reason for the latched peripheral fault has been corrected.

A. Inline CRC Fault Mode Object, Index 3003_{hex}

This object specifies which state the module is set to when an Inline CRC error is detected, and only when the current state is operational.

0 = Preoperational (default)
1 = No state change
2 = Stopped

B. Inline PF (Peripheral) Fault Mode Object, Index 3004_{hex}

This object specifies which state the module is set to when an Inline PF error is detected, and only if the current state is operational. An example of a PF is a shorted output.

0 = Preoperational (default)
1 = No state change
2 = Stopped

C. Inline Power Fault Mode Object, Index 3005_{hex}

This object specifies which state the module is set to when an Inline Power Fault error is detected, and only if the current state is operational. The faulted power supply can be U_{BK}, U_S, U_M or processor power.

0 = Preoperational (default)
1 = No state change
2 = Stopped

D. Inline Module Change Fault Mode Object, Index 3006_{hex}

This object specifies whether the bus coupler can be put into the run mode when receiving a Start Remote Node command when there has been a module change.

0 = Keep device in preoperational state (default)
1 = Allow operational mode

E. Inline Inactive Local Bus Fault Mode Object, Index 3007_{hex}

This object specifies which state the module is set to when the local bus is inactive and not running data cycles, and only if the current state is operational. This bit is set when a connection or module change error occurs or when the device is trying to initialize the local bus.

0 = Preoperational (default)
1 = No state change
2 = Stopped

F. Inline Connection Fault Mode Object, Index 3008_{hex}

This object specifies which state the module is set to when an Inline Connection Fault error is detected, and only if the current state is operational.

0 = Preoperational (default)
1 = No state change
2 = Stopped

G. Inline Faulted Cycles Mode Object, Index 3009_{hex}

This object specifies which state the module is set to when the bus couple is running data cycles using the fault data information instead of process data, and only if the current state is operational.

0 = Preoperational (default)
1 = No state change
2 = Stopped

H. Processor Power Loss Fault Mode Object, Index $300A_{hex}$

This object defines whether the bus coupler is trying to communicate with modules on the local bus or not when U_{BK} power drops below 11 V. U_{BK} power is used to generate communications power (U_L) and the voltage supply for the analog terminals ($U_{ANA)}$.

0 = Keep running cycles
1 = Reset the local bus (default). This allows all modules to go to their default output states. (See module data sheets for these default values.)

5 Diagnostics

5.1 Diagnostic and status indicators

All terminals are equipped with diagnostic and status indicators for quick local error diagnostics. They enable the clear localization of system errors (bus errors) or peripheral faults.

Diagnostics

The diagnostic indicators (red, yellow or green) provide information about the state of the terminal and, in the event of an error, provide information about the type and location of the error. A terminal is operating correctly if all of its green LEDs are on.

Status

The status indicators (yellow) indicate the status of the relevant input/output and the connected device.

Extended diagnostics

Some Inline terminals have extended diagnostics. A short circuit or overload of the sensor supply is indicated for each input. If a short circuit occurs at an output, each channel is diagnosed individually. Information about the supply voltage is also reported. Information about peripheral faults is sent to the control system with precise details of the error type and is displayed using status indicators.

For information about the diagnostic and status indicators on a specific terminal, please refer to the terminal-specific data sheet.

5.2 Diagnostics and status indicators of the bus coupler

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

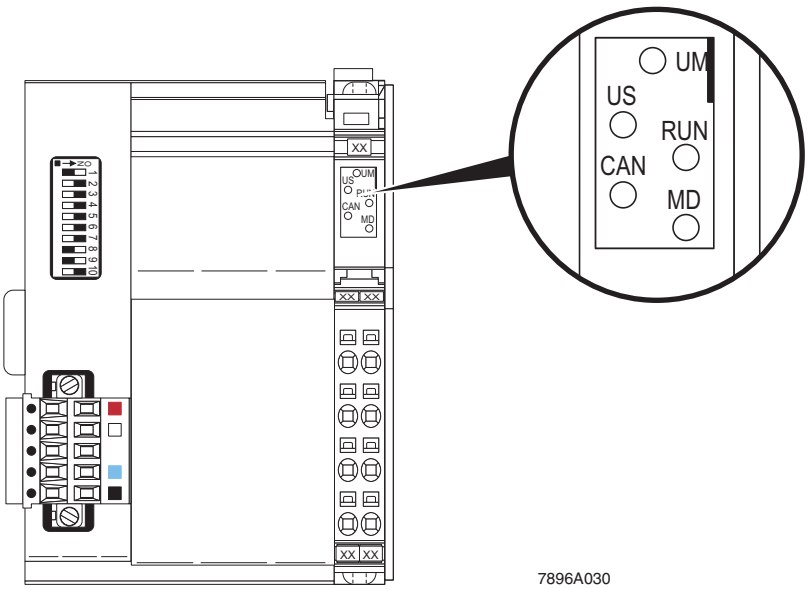


Figure 5-1 Diagnostics and status indicators of the bus coupler

LED		Meaning
CAN	Red/green LED	CANopen status
	The CAN LED may flash green and red at the same time.	
	ON (green):	"Operational" status: Device is ready to operate.
	2 Hz (medium) Flashing (green)	"Pre-Operational" status: PDO transmission not possible
	0.5 Hz (slow) flashing (green):	"Stopped" status: Failsafe status, no PDO transmission possible
	ON (red):	CAN controller is not connected to the bus.
	One flash (red)	At least one error counter has reached the warning level.
	Two flashes (red)	A guard event or heartbeat event has been triggered.
	Three flashes (red)	Sync timeout error
	OFF:	Supply voltage U_{BK} is missing or module in reset state.

LED		Meaning
MD	Green/red LED	Module status
	ON (green)	Device is ready to operate.
	ON (green/red)	Self test/read local bus.
	ON (red)	Serious error, replace device.
	Flashing (red)	Minor error (e.g., DO short circuit)
	OFF	No supply voltage U_{BK} or module in reset.
RUN	LED (multi-color)	Backplane communication status
	ON (green)	Local bus is running data cycles.
	Flashing (green)	Peripheral fault (1 Hz)
	ON (red)	Local bus stopped
	ON (red/yellow)	Configuration mismatch
	ON (green/yellow)	Preprogrammed error values are being written to the outputs.
	OFF	U_{BK} supply not present
US	Green LED	Segment supply
	ON	Segment supply present
	OFF	Segment supply not present
UM	Green LED	Main supply
	ON	Main supply present
	OFF	Main supply not present

The RUN LED will flash green/yellow when preprogrammed error state data is being driven to outputs on the Inline local bus. When the fault that caused the error data to be run is cleared the RUN LED will be permanently green. The preprogrammed error data will continue to be displayed on the digital outputs until the next updated process data object (PDO) is received.

5.3 Indicators on other terminals

For general information about the indicators on other terminals, please refer to the IL SYS INST UM E user manual. For specific information about a particular terminal, please refer to the terminal-specific data sheet.

5.3.1 Error localization using LEDs

Diagnostic and status indicators clearly denote the location of errors. If an error is detected, the error is displayed at the station and the number of the device on which the error has occurred is reported to the control system. Figure 5-2 shows an Inline station with an error at module 3.

IB IL 24 PWR IN power terminals are not numbered because they are not bus devices (they do not contain protocol chips) and therefore do not have indicators for error diagnostics.

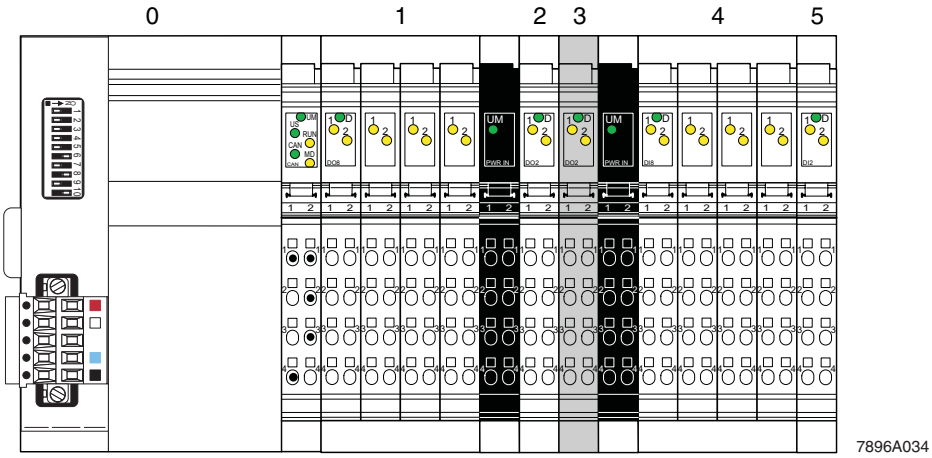


Figure 5-2 Example station for error localization

Terminals used in the example station:

0	IL CAN BK-TC-PAC	**3**	IB IL 24 DO 4-PAC
1	IB IL 24 DO 8-PAC	**4**	IB IL 24 DI 8-PAC
2	IB IL 24 DO 2-2A-PAC	**5**	IB IL 24 DI 2-PAC

Figure 5-3 shows a station with possible error states. It shows that errors have been detected at terminal 4 or that terminal 3 has failed, and shows the behavior of the diagnostic indicators on the adjacent terminals.

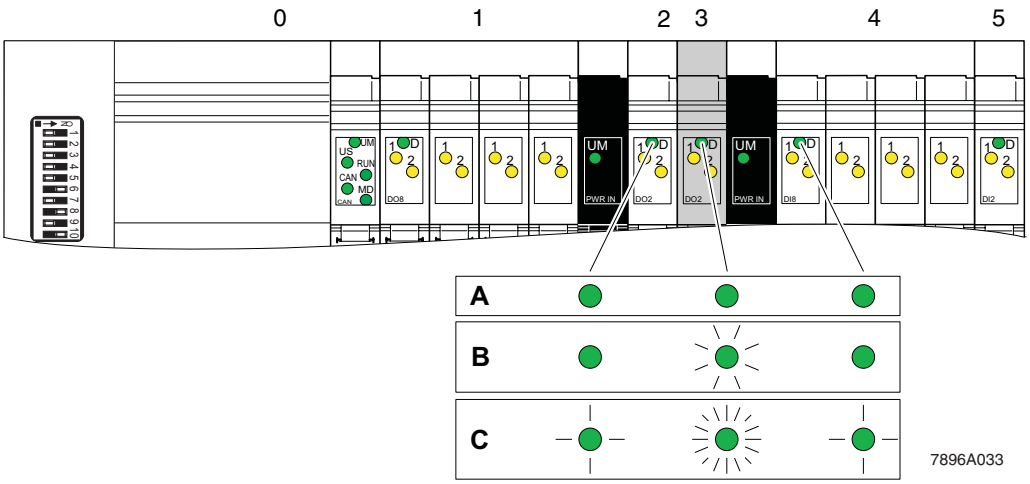


Figure 5-3 Station with diagnostic indicators

A	No error
B	Peripheral fault
C	Bus error

LED ON	LED flashes at		
	0.5 Hz (slow)	2 Hz (medium)	4 Hz (fast)
			

A **No error**
If there are no errors, the green LEDs on the bus coupler and the other modules will remain lit.

B **Peripheral error**
If the LED on module 3 is flashing at medium speed (2 Hz), a digital output module will indicate and I/O error (peripheral fault) such as a short circuit caused by a defective actuator.

C **Bus error**
If the LED on module 3 is flashing fast and the LEDs on modules 2 and 4 are flashing slow, this indicates that a bus error has occurred on module 3 or between modules 2 and 3.

Peripheral fault

(Figure 5-3, detail **B**)

Error:	Short circuit at terminal 3 (IB IL 24 DO 2-2A-PAC)
Effect:	
Control system:	Error message to the control system (peripheral fault)
Bus coupler:	Indicators remain unchanged
Terminal 3:	Green D LED flashes at 2 Hz
Other terminals:	Remain unchanged

Bus error

(Figure 5-3, detail **C**)

Error:	Bus has been interrupted after terminal 2 and before terminal 3
Effect:	
Control system:	Error can be located by the control system
Bus coupler:	Red LD LED (local bus disabled) is ON
Terminal 3:	Green D LED flashes at 4 Hz (bus error)
Other terminals:	Green D LEDs on all other terminals flash at 0.5 Hz

5.4 Emergency message and diagnostic object indexes

A detailed explanation of object indexes can be found in "Appendix: CANopen objects and indexes" on page B-1.

Diagnostic information is made available through the emergency message and several diagnostic object indexes (through SDOs). They allow the user to read the status of the Inline station. The emergency message is automatically generated while SDOs must be send by the user to the object indexes shown in this section.

5.4.1 Emergency message

The CANopen bus coupler produces an emergency message when a fault occurs. This emergency message contains the Inline status and the number of the faulty module along with other useful information. The emergency message consists of the 8 bytes defined below. The emergency message is further defined in "Appendix: CANopen objects and indexes" on page B-1.

Emergency object telegram structure:

Byte number	0	1	2	3	4	5	6	7
	Emergency error code		Error Register Object Index 1001_{hex}	Inline Status (Object 3101_{hex}, low byte)	Inline Status (Object 3101_{hex}, high byte)	Faulted module (Object 3102_{hex})	Not used	Not used

5.4.2 Diagnostic indexes

Table 5-1 Objects

Index (hex)	Name	Function
1001	Error Register	This object index allows the user to receive, through a Service Data Object (SDO), the manufacturer-specific bit 7 which signifies an error on the Inline station. The use of this bit or an emergency message allows the user to retrieve further diagnostic information.
1002	Manufacturer Status Register	Manufacturer-specific status of the Inline station and the number of the faulty Inline module are available. This object index is further defined in Appendix A.
1003	Predefined Error Field	This object index reads up to the last ten errors (subindexes). Subindex 0 designates the total number of available errors in the "stack". This object index is further defined in Appendix A.
3101	Inline Station Status	This object index displays the current Inline station status.
3102	Inline Faulted Module Number	This object index displays the current number of the faulty Inline module.
310C	Inline Latched Fault	Latches and displays the last Inline station status (low byte). This object index has the same bit meaning as the Inline Station Status Object, Index 3101_{hex} except that the bits are latched. Defined further in Appendix A.
310D	Inline Latched Faulted Module	Latches and displays the number of the last faulty Inline module. This object index has the same bit meaning as the Inline Faulted Module Object, Index 3103_{hex} except that the module number is latched. Defined further in "Appendix: CANopen objects and indexes" on page B-1.

Inline station faults will always generate an emergency message after a local fault in the operational state. By default, due to the mode settings of the fault bits, preprogrammed fault data (zero by default) will be sent to the local outputs. It is possible that error information may be overwritten due to intermittent error conditions. For such a condition, it is recommended to access the Predefined Error Field Object, Index 1003_{hex}. Each subindex in Object, Index 1003_{hex} represents a stored fault sequence.

5.4.3 Inline Station Status Object, Index 3101_{hex}

By default, the Inline status data is made available to the user as two bytes of diagnostic data in the emergency object message. These two bytes contain the Inline fault code (byte 3) and a bit that determines whether or not to device address has changed (byte 4).

Bit meanings for Inline Station Status Object, Index 3101_{hex}

Low byte, 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 Inline Max Retry Object, Index 3103_{hex} will allow the module to retransmit the data cycle up to the number of times that is set in the Max Retry parameter. If the transmission does not pass the CRC after Max Retry has expired, then the CRC Error bit is set. The default CANopen state change action for a CRC Error will cause the bus coupler to go to the preoperational state. This default is controlled by Object, Index 3003_{hex}.

Low byte, bit 1: Peripheral Fault

The PF bit is set when any output is shorted or during a loss of power to an intelligent segment module. The default CANopen state change action for a peripheral fault is "no state change". This default is controlled by Object, Index 3004_{hex}.

Low byte, bit 2: Power Fault

The Power Fault bit will be set when any of the power supplies (U_{BK}, U_S, U_M) are in an over/undervoltage condition. The default CANopen state change action for a Power Fault will cause the bus coupler to go to the preoperational state. This default is controlled by Object, Index 3005_{hex}.

Low byte, bit 3: Module Change Fault

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 the flash memory during the last configuration cycle. The default CANopen state change action for a Module Change Fault is to go either or to stay in the preoperational state. This default is controlled by Object, Index 3006_{hex}.

Low byte, bit 4: Inactive Local Bus Fault

The Inactive Local Bus Fault bit will be set when no data cycles are being driven on the local bus. When this condition occurs the bus coupler will not be in the run state. Possible causes include:

– 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.
– The bus coupler is trying to auto-recover the local bus. A fault has occurred and the bus coupler is waiting for the error to be cleared before restarting the local bus.

The default CANopen state change action for an Inactive Local Bus Fault is controlled by the Inactive Local Bus Fault Mode Object, Index 3007_{hex}. The default setting is to go to the preoperational state.

Low byte, bit 5: Inline Connection Fault

The Inline Connection Fault bit will be set when the bus coupler is no longer able to talk to the modules connected to it and when it can determine the failure position. This connection failure occurs due to a broken data path on the local bus. The exact broken path "between what two modules" can be read from the Inline First Faulted Module Object, Index $310A_{hex}$, and the Inline Last Faulted Module Object, Index $310B_{hex}$. The CANopen state change action, generated by bit 5, is determined by the Inline Connection Fault Object, Index 3008_{hex}. The default setting is to go to the preoperational state.

Low byte, bit 6: Inline Faulted Cycles Mode

The Inline Faulted Cycles Mode bit is made available to let the application know that the local outputs have gone to their preprogrammed CANopen error state and will no longer respond to the controller. Faulted data cycles will be run when one of the Inline Station Status Faults cause the CANopen interface to change states or the Fault Response Mode Object, Index 3002_{hex} is set to mode 2 and an actual fault has occurred.

Low byte, bit 7: Reserved

Bit 7 is reserved for future use.

High byte, bit 0: Node Address Change

This bit is set to "1" when the user changes the address of the bus coupler. This bit can be cleared by writing the ASCII string "SAVE" to the Store Parameter Object, Index 1010_{hex} or by setting all address DIP switches to "0" and power cycling the bus coupler, setting the address DIP switches to the desired address, then power cycling the bus coupler more.

5.4.4 Inline Faulted Module Number Object, Index 3102_{hex}

This object contains the number of the module that has faulted. Modules on the station are numbered starting at the bus coupler being assigned "0" and continuing to the right up to 63. This is the maximum number of devices that can be connected to an Inline station.

All Inline local errors will be sent over the CANopen network in the form of an emergency message.

By default, these errors are considered a major error (except for a peripheral fault) and the red RUN LED on the bus coupler will be affected, see Figure 5-1 on page 5-2.

5.4.5 Examples of retrieving diagnostic information

The following information provides and example of how to retrieve diagnostic information. The station described in Section "Error localization using LEDs" on page 5-4 will be used as example station. A peripheral fault (I/O error) (example A) and a bus error (example B) will be described. For each example, explanations will be given using the Emergency Object and the Service Data Object (SDO).

5.4.5.1 Example A — I/O error, short-circuit on module number 3 (IB IL 24 DO 4-PAC)

Diagnostic information pertaining to a short circuit on module number 3 can be retrieved in the following to ways.

A. Receiving fault and location code(s) using the Emergency Object

The user can evaluate byte 3 and byte 5 for the Emergency Object to determine the fault code and the location of errors on the Inline station.

1. Read byte 3, Inline Station Status, low byte. This value will contain the fault code. In this example the output short circuit will cause the Peripheral Fault bit (bit 1) to be set and a value of "2" to be read in byte 3 of the Emergency Object.
2. Read byte 5 (Inline Faulted Module Number). This byte value represents the location of the faulted module on the station. In our example, byte 5 would read the value of 3, thereby indicating that module number 3 is faulted, see Figure 5-2 on page 5-4. Bear in mind that device numbering on a CANopen Inline station starts with 0 (bus coupler) and continues right to number 63.

B. Receiving fault and location code(s) using SDOs

The user can evaluate the fault by sending a Service Data Object (SDO) message to the Inline Station Status Object, Index 3101_{hex} and Inline Faulted Module Object, Index 3102_{hex}. This same information is also available from the Manufacturer Status Register Object, Index 1002_{hex}.

1. Read the Inline Station Status Object, Index 3101_{hex} This value represents the fault code. In our example the output short circuit will cause the Peripheral Fault bit (bit 1) to be set. Bit 1 represents a value of 2 and is read in subindex 0.
2. Read the Inline Faulted Module Object, Index 3102_{hex} This value represents the peripheral fault location. For this example, the value will be 3, see Figure 5-2 on page 5-4. Bear in mind that device numbering on a CANopen Inline station starts with 0 (bus coupler) and continues right to number 63.

5.4.5.2 Example B — Bus error, communications break between modules 2 and 3

Diagnostic information pertaining to a communications break between modules can be retrieved in the following two ways.

A. Receiving fault and location code(s) using the Emergency Object

The user can evaluate byte 3 and byte 5 for the Emergency Object to determine the fault code and the location of errors on the Inline station.

1 Read byte 3, Inline Station Status byte. This value represents the fault code. In our example the break between modules 2 and 3 will cause either the Module Change Fault bit (bit 3) or the Inline Connection Fault bit (bit 5) to be set. Depending on the nature of the break, a value of 8 (Module Change Fault) or 32 (Inline Connection Fault) is read in byte 3 of the Emergency Object.

If the fault is a Module Change Fault, the complete communications path (DI and DO) is broken between modules 2 and 3. If the fault is an Inline Connection Fault, only one communications path is broken (either DI or DO).

2 Read byte 5, Inline Faulted Module Number This value represents the location of the break between modules 2 and 3. For our example there are two possible fault conditions. These are a Module Change Fault or an Inline Connection Fault.

Module Change Fault

For a Module Change Fault, a value of 3 in byte 5 will occur. This error could be viewed by the bus coupler as module 3 is missing.

Inline Connection Fault

For a Connection Fault, a value of 3 in byte 5 will occur. This error could be viewed by the bus coupler as a connection fault between modules 2 and 3. In our example, the Inline Faulted Module byte is 0 (indicating the bus coupler). The actual location of the fault must be obtained by sending an SDO message to the Inline First Faulted Module Object, Index $310A_{hex}$. This will result in receiving the module number for the first end of the connection fault. The location would be a value of either 2 or 3 depending on which communications line is broken.

An SDO message must be sent to the Inline Last Faulted Module, Index $310B_{hex}$ to receive a value indicating the location of the other end of the communications break. This location would be a value of either 3 or 2 depending on which communications line is broken, see Figure 5-2 on page 5-4. Bear in mind that device numbering on a CANopen Inline station starts with 0 (bus coupler) and continues right to number 63.

B. Receiving fault and location code(s) using SDOs

The user can evaluate the fault by sending a Service Data Object (SDO) message to the Inline Station Status Object, Index 3101_{hex} and Inline Faulted Module Object, Index 3102_{hex}. This same information is also available from the Manufacturer Status Register Object, Index 1002_{hex}.

1 Read the Inline Station Status Object, Index 3101_{hex} This value represents the fault code. In our example the break between modules 2 and 3 will cause either the Module Change Fault bit (bit 3) or the Inline Connection Fault bit (bit 5) to be set. Depending on the nature of the break, a value of 8 (Module Change Fault) or 32 (Inline Connection Fault) will be read at subindex 0.

2 Read the Inline Faulted Module Object, Index 3102_{hex} This value represents the location of the break between modules 2 and 3. For our example, there are two possible fault conditions. These are a Module Change Fault or an Inline Connection Fault.

Module Change Fault

For a Module Change Fault, a value of 3 will occur. This error could be viewed by the bus coupler as module 3 is missing.

Inline Connection Fault

If the error is an Inline Connection Fault the location of the error is between modules 2 and 3. In this case the Inline Faulted Module will show the number of the module immediately before the break. The actual location of the fault must be obtained by sending an SDO message to the Inline First Faulted Module Object, Index $310A_{hex}$. This will result in receiving the module number for the first end of the connection fault. The location would be a value of either 2 or 3 depending on which communications line is broken.

An SDO message must be sent to the Inline Last Faulted Module, Index $310B_{hex}$ to receive a value indicating the location of the other end of the communications break. This location would be a value of either 3 or 2 depending on which communications line is broken, see Figure 5-2 on page 5-4. Bear in mind that device numbering on a CANopen Inline station starts with 0 (bus coupler) and continues right to number 63.

5.4.6 Fault modes

By default, all Inline Station Status byte fault bits (byte 0, bit 0, and bits 2 to 6) except for bit 1 are considered major faults. The default action for fault modules is to go to the preoperational state. This will stop all PDO activity and the emergency message will be sent to the master. Default values can be changed b y using the EDS file or by sending an SDO to the Fault Mode Object, Indexes 3003_{hex} - 3009_{hex}. Possible fault action settings are:

0 = Enter preoperational (only if current state is operational)
1 = No state change
2 = Stopped

5.4.7 Latched diagnostics

The bus coupler will latch the last occurring Inline Status fault, module number and connection point failures. This benefits the user by capturing any fault that may occur intermittently and too quickly to be updated by the CANopen master. These latched values can be cleared by writing a 0 to the Latched Inline Status Object, Index $310C_{hex}$, Subindex 0. The following latched diagnostic data is available through the EDS file or by sending an SDO to the Inline Interface Object, Indexes $310C_{hex}$ to $310F_{hex}$.

- **Latched Inline Status** — This parameter will contain the last reported Inline station failure. The bit weights signify the same failures as described in the Inline Station Status byte.
- **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 Faulted Module Number.
- **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.

5.4.8 Inline Interface Control Byte

The Inline Interface Control Byte is used to acknowledge latched peripheral faults (bit 0), clear latched input states (bit 1) or to restart the local bus (bit 2).

For an explanation of latching digital input states, refer to Section "I/O data transfer" on page 4-12.

By default the Inline Interface Control Byte is not included in a PDO. It can be added to a receive PDO by mapping the Inline Interface Object, Index 3111_{hex}, Subindex 0. If the user would rather access this byte through a Service Data Object, an upload or download can be sent to the Inline Interface Control Byte Object, Index 3111_{hex}, Subindex 0. If accessed using an SDO, this bit will self clear.

- Bit 0: When set to a one, this bit will attempt to clear all latched peripheral faults.
- Bit 1: When set to a one, this bit will clear all latched input states.
- Bit 2: When set to a one, this bit will attempt to start the local bus.

The latched peripheral fault can only be generated by certain Inline modules. This includes the IB IL 24 EDI 2-DESINA-PAC module.

5.4.9 Error states (fault states)

The bus coupler supports the standard CANopen digital and analog outputs, error modes and error values. Error mode and error value can be set and read by the use of Service Data Objects (SDOs). Error modes will only occur during a network error, they will not occur after an Inline local error, unless the Fault Mode Object, Index 3002_{hex} is set to a 2. The default setting for the error value is 0.

A. Digital output support

– Holds last state
– Turn OFF during a faulted condition (default)
– Turns ON during a faulted condition

B. Analog output support

– Hold last value
– Set to the value determined by the error value object.

"Appendix: CANopen objects and indexes" on page B-1 will detail the digital output and analog error mode and error value index objects.

5.4.10 Analog input, thermocouple and RTD fault codes

Inline analog inputs, thermocouples and RTDs can report diagnostic codes. These codes must be read from the PDO data or through an SDO message to the Read Analog Input 16-Bit Object, Index 6401_{hex}, see Appendix A. 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 defined. By default, error codes are received in the Inline "IL" format and can be viewed as shown in Table 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	Underrange
8002	Open circuit
8004	Measured value invalid
8008	Cold junction defective
8020	I/O supply voltage faulty
8010	Invalid configuration
8040	Module defective
8080	Overrange

6 Technical data and ordering data

6.1 Technical data

Data presented in the following tables is based on the product being mounted in the preferred vertical mounting position. Please refer to the CANopen specification and to the module-specific data sheet for additional information.

For all technical data that generally applies to Inline, please refer to the IL SYS INST UM E user manual. This section only lists data that applies specifically for use on CANopen.

General data	
Housing dimensions (width x height x depth)	91 mm x 120 mm x 72 mm
Weight	225 g
Ambient temperatures (operation)	-25°C ... +55°C
Ambient temperature (storage/transport)	-25°C ... +85°C
Permissible humidity (operation/storage/transport)	10% ... 95% according to EN 61131-2
Permissible air pressure (operation/storage/transport)	70 kPa ... 106 kPa (up to 3000 m above sea level)
Degree of protection	IP20
Class of protection	Class III, IEC 61140
Connection data for Inline connectors	
Connection method	Spring-cage terminal
Conductor cross-section	0.08 mm^2 ... 1.5 mm^2 (solid or stranded), 28 - 16 AWG

System data CANopen	
Maximum number of I/O modules	63
Number of PDOs	6Tx/4Rx
Number of SDOs	1Tx/1Rx
Maximum number of digital input and output points	510
Maximum number of analog input and output points	254
Communication profile	DS-301 V3.0
Device profile	DS-401 V1.4

System limits of bus couplers	
Settable baud rates	10 kbaud, 20 kbaud, 50 kbaud, 125 kbaud, 250 kbaud, 500 kbaud, 1 Mbaud
Transmission speed on the local bus	500 kbps
Number of I/O devices within an Inline system	63
Maximum power supply at U_L	2 A
Maximum power supply at U_{ANA}	0.5 A
Maximum power supply at U_S	8 A
Maximum power supply at U_M	8 A
Maximum current consumption of the I/O terminals	See terminal-specific data sheet

Observe the current consumption of every device when configuring an Inline station. The current consumption values are listed in each terminal-specific data sheet. Current consumption can differ, depending on the type of terminal. If the maximum current carrying capacity of a power supply is reached, an additional power terminal must be used.

CANopen interface

CAN bus, connected using 2 x 5-pos. TWIN-COMBICON connectors

Common data for 24 V supply U_{BK}, U_S, U_M

Recommended cable lengths	30 m, maximum; routing cables through outdoor areas is not permitted
Continuation	Via potential routing
Nominal value	24 V DC
Permissible range (according to EN 61131-2)	19.2 V to 30 V (ripple included)
Protective equipment	
Surge voltage	Yes, suppressor diode
Polarity reversal	Yes, suppressor diode antiparallel to supply voltage

Provide an external fuse for the 24 V area

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

The communications power U_L (7.5 V) and the analog supply U_{ANA} (24 V) are generated from the bus coupler supply U_{BK}.

Notes on the voltages U_L and U_{ANA}

Communications power U_L (7.5 V)	U_L is not electrically isolated from the 24 V bus coupler voltage U_{BK}. U_L is not electrically isolated from the I/O voltages U_M and U_S. Communications power U_L is electronically short-circuit protected.
Analog supply U_{ANA} (24 V)	Decoupling of the 24 V input voltage by means of a diode. Smoothing via base frequency of 9.8 kHz and attenuation of 40 dB/decade. U_{ANA} is not electrically isolated from the 24 V bus coupler supply and the 7.5 V communications power.

Current consumption/power consumption

Current consumption from U_{BK} (24 V)	
Without connected I/O terminals	<100 mA
With the maximum number of connected I/O terminals	1.25 A, maximum
Current consumption from U_S (24 V) (total current with U_M)	8 A, maximum
Current consumption from U_M (24 V) (total current with U_S)	8 A, maximum
Power dissipation of entire device	2.7 W, typical

Conformance with EMC Directive 2004/108/EC

Noise immunity test according to EN 61000-6-2

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

Noise emission test according to EN 61000-6-4

Noise emission of housing	EN 55011	Class A

Approvals

For the latest approvals, please visit www.phoenixcontact.net/catalog.

6.2 Ordering data

Product

Description	Type	Order No.	Pcs./Pkt.
Inline bus coupler for CANopen, 24 V DC, bus interface 2 x 5-pos. TWIN-COMBICON connector, complete with accessories (Inline connector, labeling field, end clamps)	IL CAN BK-TC-PAC	2718701	1

Documentation

Description	Type	Order No.	Pcs./Pkt.
"Configuring a CANopen system using SyCon" quick start guide	UM QS EN IL CAN BK-TC-PAC	–	–
"Automation terminals of the Inline product range" user manual	IL SYS INST UM E	2698737	1
"Summary of key data for Inline devices" data sheet	DB GB IB IL DEVICE LIST	–	–
" I/O terminals at bus couplers" application note	AH IL BK IO LIST	9015358	–

The documentation listed above and all terminal-specific documentation can be downloaded at www.download.phoenixcontact.net/download.

Make sure you always use the latest documentation.

A Appendix: Tips and examples

This section provides examples, tips and considerations that may be helpful when designing and implementing an Inline station.

A 1 Configuration examples

A 1.1 Inline segment terminal with four fused digital outputs

Figure A-1 shows a circuit diagram that uses a fused segment terminal (IB IL 24 SEG/F-PAC) and a digital output terminal with four outputs.

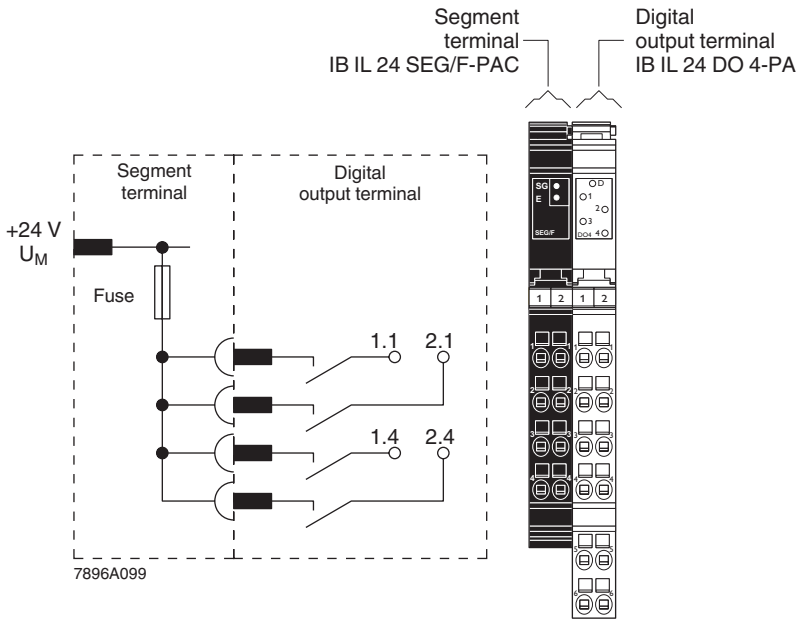


Figure A-1 Configuration example for four fused digital outputs

A 1.2 Emergency stop circuitry with two fused outputs

Figure A-2 shows a diagram for establishing an emergency stop circuit for two fused outputs (U_S power). The emergency stop circuit consists of a segment terminal with fuse (IB IL 24 SEG/F-PAC), a safety relay terminal (IB IL 24 SAFE 1) and a digital output terminal (IB IL 24 DO 2-2A-PAC) with two outputs.

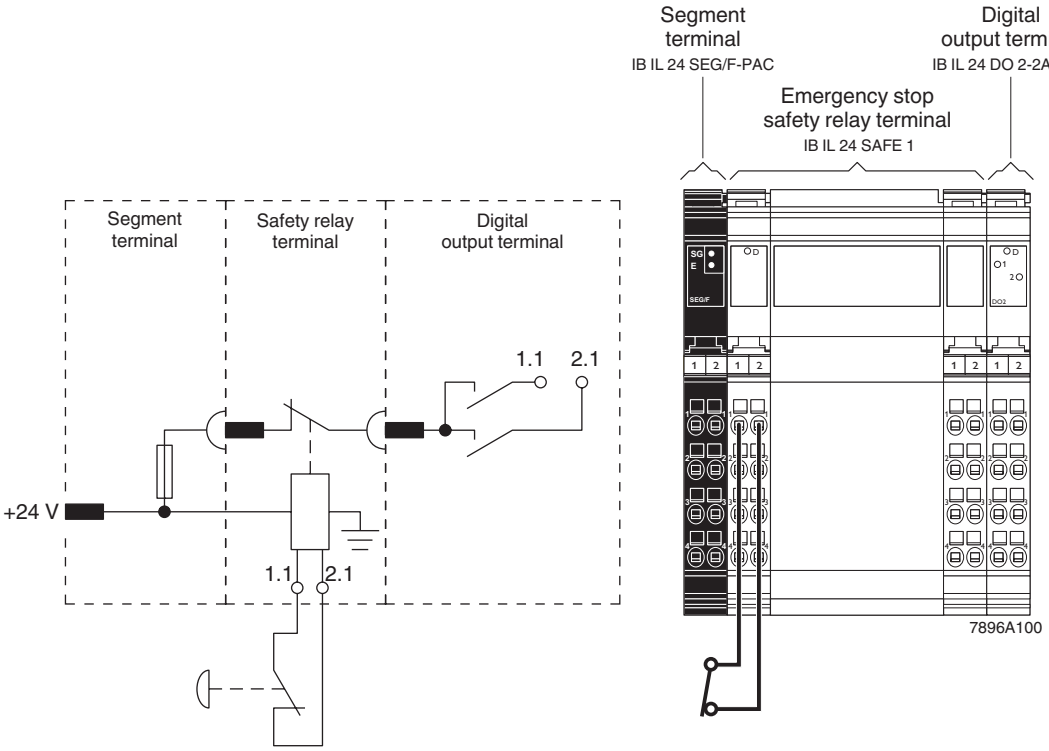


Figure A-2 Configuration example for emergency stop circuitry

A 1.3 Example plant layout

Figure A-3 shows an example plant layout. It is controlled by a PC and using the Inline product family. A brief description of Inline products and their function is provided in the following paragraphs.

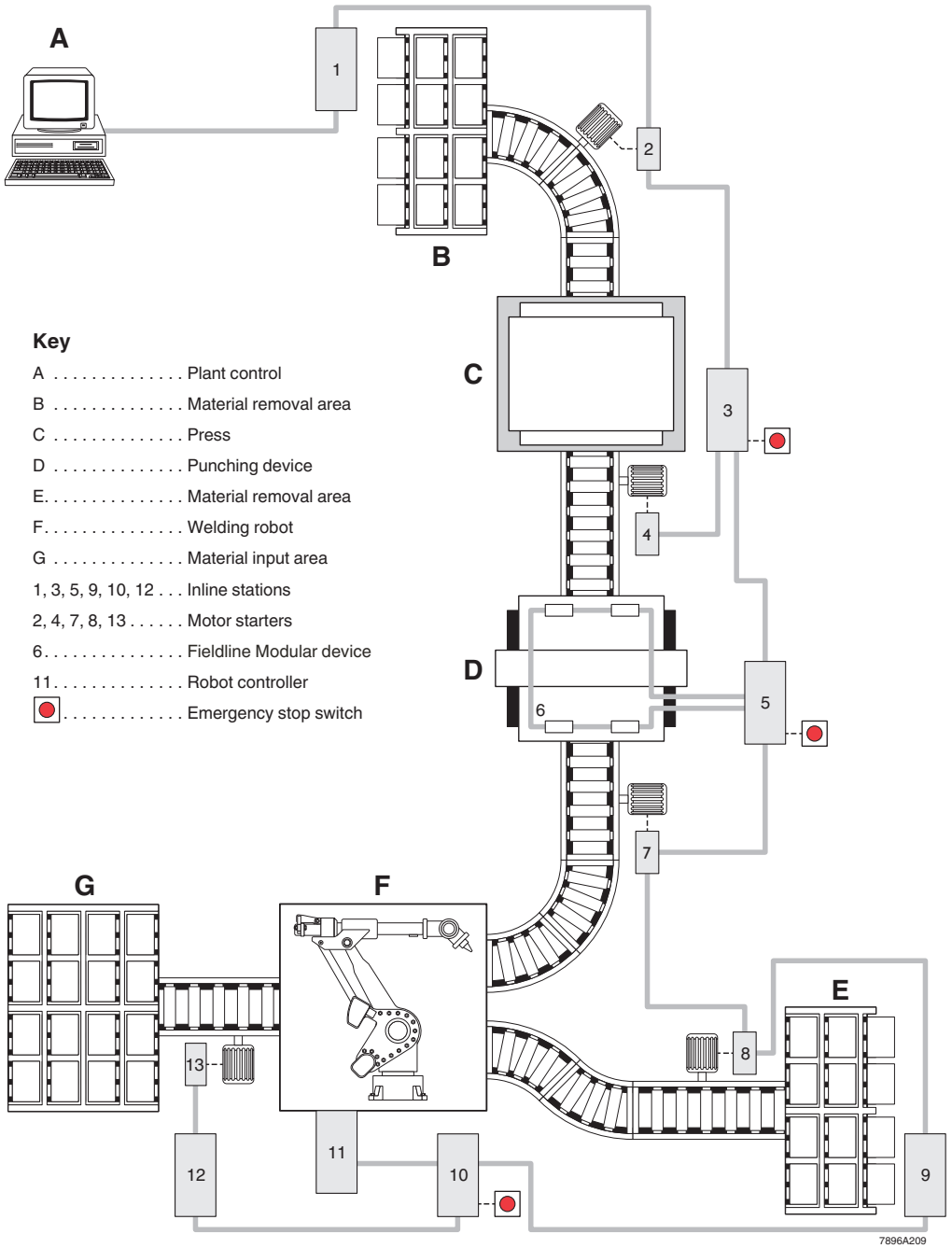


Figure A-3 Example plant layout controlled by a PC and using Inline

A. Tour of the example plant (see Figure A-3)

Inline station 1 modules control the removal of material from area B.

Motor starter (2) is directly connected to the remote bus and controls the conveyor belt motor between area "B" and "C".

Inline station (3) modules control the press at area "C". A remote bus branch from station (3) controls the motor starter (4). The motor starter in turn controls the motor that drives the conveyor belt between area "C" and "D". Because this conveyor must be protected, an emergency stop switch is installed at Inline station (3).

Inline station (5) modules control the punching operation located in area "D". Fieldline Modular devices (6) connected to station (5) are used to monitor punch status. An emergency stop switch is required at Inline station (5) to protect the punching device.

The conveyor belt between area "D" and "F" is controlled by motor starter (7). The conveyor belt between area "F" and "E" is controlled by motor starter (8).

Inline station 9 modules control the removal of material from area "E".

A robotic control system (11) is connected to and controlled by Inline station (10). An emergency stop switch as been installed at this location to protect the robot.

Inline station 12 modules control the storage of materials in the material input area "G".

Motor starter (13) is directly connected to the remote bus and controls the conveyor belt motor.

A 2 Tips on working with the Inline system

A 2.1 Determining a voltage failure with passive power and segment terminals

Passive terminals cannot indicate a voltage failure over the bus. However, by using power and segment terminals that are fused, the red LED on these terminals, when lit, indicates a local voltage failure. We recommend monitoring segment voltage over the bus by using a digital input terminal.

A 2.2 Sequence of the terminals within an Inline station

Please refer to the IL SYS INST UM E user manual for the sequence of the terminals within an Inline station.

B Appendix: CANopen objects and indexes

B 1 General overview

The bus coupler provides the interface between CANopen and the I/O modules of the Inline product range.

Configuration Objects (Indexes 3000_{hex}— $300F_{hex}$) allow the unit to be configured for special Inline fault modes etc.

Inline Interface Objects (Indexes 3100_{hex}— 3111_{hex}) allow the user to configure and monitor the Inline side of the IL CAN BK-TC-PAC. These objects allow the user to see the number of Inline modules attached to the device, the count of bits and the count of bytes, as well as the status of the Inline interface.

Inline Module Objects (Indexes 3100_{hex}— 3111_{hex}) allow the user to gain access to the identification codes etc. of the I/O modules attached to the device. These objects also allow the user quick information on which CANopen index and subindex the Inline module is mapped to.

Inline Special Function Objects (Indexes 3300_{hex}— $330F_{hex}$) allow access and control of any module that does not currently map to a CANopen approved object. These objects transparently pass data to and from the Inline modules. Therefore, an understanding of the formatting of the Inline data for these modules is needed.

B 2 Supported codes

B 2.1 SDO abort codes

Abort code	Description
$0503\ 0000_{hex}$	Toggle bit not alternated
$0504\ 0000_{hex}$	SDO protocol timed out
$0504\ 0001_{hex}$	Client/server command specifier not valid or unknown
$0504\ 0002_{hex}$	Invalid block size (block mode only)
$0504\ 0003_{hex}$	Invalid sequence number (block mode only)
$0504\ 0004_{hex}$	CRC error (block mode only)
$0504\ 0005_{hex}$	Out of memory
$0601\ 0000_{hex}$	Unsupported access to an object
$0601\ 0001_{hex}$	Attempt to read a write only object
$0601\ 0002_{hex}$	Attempt to read a write only object
$0602\ 0000_{hex}$	Object does not exist in the object dictionary
$0604\ 0041_{hex}$	Object cannot be mapped to the PDO
$0604\ 0042_{hex}$	The number and length of the objects to be mapped would exceed PDO length
$0604\ 0043_{hex}$	General parameter incompatibility reason
$0604\ 0047_{hex}$	General internal incompatibility in the device
$0606\ 0000_{hex}$	Access failed due to a hardware error
$0607\ 0010_{hex}$	Data type does not match, length of service parameter does not match
$0607\ 0012_{hex}$	Data type does not match, length of service parameter too high
$0607\ 0013_{hex}$	Data type does not match, length of service parameter too low
$0609\ 0011_{hex}$	Subindex does not exist

The abort codes not listed here are being reserved for future use.

B 2.2 Emergency error codes

Error code (hex)	Description
00xx	Error reset or no error
10xx	Generic error
3120_{hex}	Input voltage to low
3220_{hex}	Internal voltage to low
3320_{hex}	Output voltage to low
81xx	Communication
8110	CAN overrun (objects lost)
8120	CAN in error passive mode
8130	Life guard error or heartbeat error
8140	Recovered from bus off
8150	Transmit COB-ID
82xx	Protocol error
8210	PDO not processed due to length error
8220	PDO length exceeded
FFxx	Device specific: – $FF00_{hex}$ Configuration Status (Object 3101_{hex}) – $FF01_{hex}$ Inline Status Fault (Object 3101_{hex})

B 3 Emergency object data

The emergency telegram consists of 8 bytes with the data as shown in Emergency Object.

Byte number	Content
Byte 0	Emergency Error Code
Byte 1	Emergency Error Code
Byte 2	Error Register (Object 1001_{hex})
Byte 3	Station Status (Object 3101_{hex}) Least Significant Byte
Byte 4	Station Status (Object 3101_{hex}) Most Significant Byte
Byte 5	Faulted Module (Object 3102_{hex})
Byte 6	Reserved
Byte 7	Reserved

B 4 Predefined connection set

Bit number: COB Identifier

Function code	Node ID
10, 9, 8, 7	6, 5, 4, 3, 2, 1, 0

Identifier allocation scheme for the predefined connection set

B 4.1 Broadcast objects

Object	Function code	Resulting COB-ID	Communication parameters at index
NMT	0000	0	-
SYNC	0001	128 (80_{hex})	1005_{hex}
TIME STAMP	0010	256 (100_{hex})	1012_{hex}, 1013_{hex}

B 4.2 Peer-to-peer objects

Object	Function code (binary)	Resulting COB-ID	Communication parameters at index
EMERGENCY	0001	129 (81_{hex}) - 255 (FF_{hex})	1014_{hex}, 1015_{hex}
PDO1 (tx)	0011	385 (181_{hex}) - 511 ($1FF_{hex}$)	1800_{hex}
PDO1 (rx)	0100	513 (201_{hex}) - 639 ($27F_{hex}$)	1400_{hex}
PDO2 (tx)	0101	641 (281_{hex}) - 767 ($2FF_{hex}$)	1801_{hex}
PDO2 (rx)	0110	769 (301_{hex}) - 895 ($37F_{hex}$)	1401_{hex}
PDO3 (tx)	0111	897 (381_{hex}) - 1023 ($3FF_{hex}$)	1802_{hex}
PDO3 (rx)	1000	1025 (401_{hex}) - 1151 ($47F_{hex}$)	1402_{hex}
PDO4 (tx)	1001	1153 (481_{hex}) - 1279 ($4FF_{hex}$)	1803_{hex}
PDO4 (rx)	1010	1281 (501_{hex}) - 1407 ($57F_{hex}$)	1403_{hex}
SDO (tx)	1011	1409 (581_{hex}) - 1535 ($5FF_{hex}$)	1200_{hex}
SDO (rx)	1100	1537 (601_{hex}) - 1663 ($67F_{hex}$)	1200_{hex}
NMT Error Control	1110	1793 (701_{hex}) - 1919 ($77F_{hex}$)	1016_{hex}, 1017_{hex}

1. Transmit and receive PDOs seen from the CAN slave's point of view.
2. The predefined connection set always applies to the standard CAN frame with 11-bit identifier, event if extended CAN frames are present in the network.

a. Assigning COB-IDs

When assigning COB-IDs to SDOs and PDOs, the user should use care. If the user selects a COB-ID for a PDO that is already assigned to the bus coupler's SDO by the predefined connection set, the bus coupler will accept the COB-ID, but the module may not function as the user desires. The COB-ID acceptance is done in order to provide greater flexibility for the advanced user.

B 5 Object dictionary data types

Index	Object	Name
0001	DEFTYPE	BOOLEAN
0002	DEFTYPE	INTEGER8
0003	DEFTYPE	INTEGER16
0004	DEFTYPE	INTEGER32
0005	DEFTYPE	UNSIGNED8
0006	DEFTYPE	UNSIGNED16
0007	DEFTYPE	UNSIGNED32
0008	DEFTYPE	REAL32
0009	DEFTYPE	VISIBLE_STRING
000A	DEFTYPE	OCTET_STRING
000B	DEFTYPE	UNICODE_STRING
000C	DEFTYPE	TIME_OF_DAY
000D	DEFTYPE	TIME_DIFFERENCE
000E	Reserved	
000F	DEFTYPE	DOMAIN
0010	DEFTYPE	INTEGER24
0011	DEFTYPE	REAL64
0012	DEFTYPE	INTEGER40
0013	DEFTYPE	INTEGER48
0014	DEFTYPE	INTEGER56
0015	DEFTYPE	INTEGER64
0016	DEFTYPE	UNSIGNED24
0017	Reserved	
0018	DEFTYPE	UNSIGNED40
0019	DEFTYPE	UNSIGNED48
001A	DEFTYPE	UNSIGNED56
001B	DEFTYPE	UNSIGNED64
001C-001F	Reserved	-
0020	DEFSTRUCT	PDO_COMMUNICATION_PARAMETER
0021	DEFSTRUCT	PDO_MAPPING
0022	DEFSTRUCT	SDO_PARAMETER
0023	DEFSTRUCT	IDENTITY
0024-003F	Reserved	-
0040-005F	DEFSTRUCT	Manufacturer Specific Complex Data Types
0060-007F	DEFTYPE	Device Profile (0) Specific Standard Data Types
0080-009F	DEFSTRUCT	Device Profile (0) Specific Standard Data Types
00A0-00BF	DEFTYPE	Device Profile 1 Specific Standard Data Types
00C0-00DF	DEFSTRUCT	Device Profile 1 Specific Complex Data Types
00E0-00FF	DEFTYPE	Device Profile 2 Specific Standard Data Types
0100-011F	DEFSTRUCT	Device Profile 2 Specific Complex Data Types
0120-013F	DEFTYPE	Device Profile 3 Specific Standard Data Types
0140-015F	DEFSTRUCT	Device Profile 3 Specific Complex Data Types
0160-017F	DEFTYPE	Device Profile 4 Specific Standard Data Types
0180-019F	DEFSTRUCT	Device Profile 4 Specific Complex Data Types

Index	Object	Name
01A0-01BF	DEFTYPE	Device Profile 5 Specific Standard Data Types
01C0-01DF	DEFSTRUCT	Device Profile 5 Specific Complex Data Types
01E0-01FF	DEFTYPE	Device Profile 6 Specific Standard Data Types
0200-021F	DEFSTRUCT	Device Profile 6 Specific Complex Data Types
0220-023F	DEFTYPE	Device Profile 7 Specific Standard Data Types
0240-025F	DEFSTRUCT	Device Profile 7 Specific Complex Data Types

B 5.1 Object dictionary structure

Index (hex)	Object
0000	Not Used
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Complex Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardized Device Profile Area
A000-FFFF	Reserved for further use

B 6 Object dictionary overview

Objects listed below are described in more detail following this overview.

B 6.1 Object dictionary entries for communication

Index (hex)	Object	Name	Data type	Access	M/O
Standard objects					
1000	VAR	Device type	UNSIGNED32	RO	M
1001	VAR	Error register	UNSIGNED8	RO	M
1002	VAR	Manufacturer status register	UNSIGNED32	RO	O
1003	VAR	Predefined error field	UNSIGNED32	RO	O
1005	VAR	COB-ID SYNC	UNSIGNED32	RW	O
1008	VAR	Manufacturer device name	Vis-String	const	O
1009	VAR	Manufacturer hardware version	Vis-String	const	O
100A	VAR	Manufacturer software version	Vis-String	const	O
100C	VAR	Guard time	UNSIGNED16	RW	O
100D	VAR	Life time factor	UNSIGNED8	RW	O
1010	ARRAY	Store parameters	UNSIGNED32	RW	O
1011	ARRAY	Restore default parameters	UNSIGNED32	RW	O
1012	VAR	COB-ID TIME	UNSIGNED32	RW	O

Index (hex)	Object	Name	Data type	Access	M/O
1014	VAR	COB-ID EMCY	UNSIGNED32	RW	O
1015	VAR	Inhibit Time EMCY	UNSIGNED16	RW	O
1016	ARRAY	Consumer heartbeat time	UNSIGNED32	RW	O
1017	VAR	Producer heartbeat time	UNSIGNED16	RW	O
1018	RECORD	Identity ObjecT	Identity (23h)	RO	M
1020	ARRAY	Verify Configuration	UNSIGNED32	RW	O
1027	ARRAY	Module List	UNSIGNED16	RO	O
1029	ARRAY	Error Behavior	UNSIGNED8	RW	O
Server SDO Parameter					
1200	RECORD	1st Server SDO parameter	SDO Par. (22_{hex})	RO	O
1201	RECORD	2nd Server SDO parameter	SDO Par. (22_{hex})	RW	O
Client SDO Parameter					
Not Supported					
Receive PDO Communication Parameter					
1400	RECORD	1st receive PDO Par.	PDO CommPar (20_{hex})	RW	M/O
1401	RECORD	2nd receive PDO Par.	PDO CommPar (20_{hex})	RW	M/O
...	...	...	...	...	...
141F	RECORD	32nd receive PDO Par.	PDO CommPar (20_{hex})	RW	M/O
Receive PDO Mapping Parameter					
1600	RECORD	1st receive PDO	PDO Mapping (21_{hex})	RW	M/O
1601	RECORD	2nd receive PDO	PDO Mapping (21_{hex})	RW	M/O
...	...	...	...	...	...
161F	RECORD	32nd receive PDO	PDO Mapping (21_{hex})	RW	M/O
Transmit PDO Communication Parameter					
1800	RECORD	1st transmit PDO Par.	PDO CommPar (20_{hex})	RW	M/O
1801	RECORD	2nd transmit PDO Par.	PDO CommPar (20_{hex})	RW	M/O
...	...	...	...	...	...
181F	RECORD	32nd transmit PDO Par.	PDO CommPar (20_{hex})	RW	M/O
Transmit PDO Mapping Parameter					
1A00	RECORD	1st transmit PDO	PDO Mapping (21_{hex})	RW	M/O
1A01	RECORD	2nd transmit PDO	PDO Mapping (21_{hex})	RW	M/O
...	...	...	...	...	...
1A1F	RECORD	32nd transmit PDO	PDO Mapping (21_{hex})	RW	M/O

Ranges 1600_{hex} — $161F_{hex}$ and $1A00_{hex}$ — $1A1F_{hex}$ can also be used to map multiplexed PDOs, see Manufacturer's Specifications in the following table:

B 6.2 Manufacturer-specific I/O objects

Index (hex)	Object	Name	Data type	Access
2000	ARRAY	DIP Latch Enable 8-Bit	UNSIGNED8	RW
2001	ARRAY	DIP Latch State 8-Bit	UNSIGNED8	RW
2400	ARRAY	AIP Range	UNSIGNED16	RW
2410	ARRAY	AOP Response Data	UNSIGNED16	RW

B 6.3 Inline configuration objects

Index (hex)	Object	Name	Data type	Access
3000	VAR	Reconfigure I/O	BOOLEAN	RW
3002	VAR	Fault Mode	UNSIGNED8	RW
3003	VAR	CRC Fault Mode	UNSIGNED8	RW
3004	VAR	PF mode	UNSIGNED8	RW
3005	VAR	Power Fault Mode	UNSIGNED8	RW
3006	VAR	Module Change Fault Mode	UNSIGNED8	RW
3007	VAR	Local Bus Inactive Fault Mode	UNSIGNED8	RW
3008	VAR	Connection Fault Mode	UNSIGNED8	RW
3009	VAR	Faulted Cycles Fault Mode	UNSIGNED8	RW
300A	VAR	Processor Power Fault Mode	UNSIGNED8	RW
300F	VAR	Erase Configuration	BOOLEAN	RW
330D	ARRAY	Special Func. Data Out (8 Byte)	UNSIGNED64	RWW
330E	ARRAY	Special Func. Data In (>8 Byte)	UNSIGNED8	RO
330F	ARRAY	Special Func. Data Out (>8 Byte)	UNSIGNED8	RWW

B 6.4 Inline interface objects

Index (hex)	Object	Name	Data type	Access
3100	VAR	Baud Rate	UNSIGNED8	RO
3101	VAR	Station Status	UNSIGNED16	RO
3102	VAR	Faulted Module	UNSIGNED8	RO
3103	VAR	Retry Max	UNSIGNED8	RW
3104	VAR	Number Modules	UNSIGNED8	RO
3105	VAR	Count of Bits	UNSIGNED16	RO
3106	VAR	Count of Bytes	UNSIGNED16	RO
3109	VAR	Loop Diagnostic Count	UNSIGNED16	RO
310A	VAR	Faulted Module (Before Break)	UNSIGNED8	RO
310B	VAR	Faulted Module (After Break)	UNSIGNED8	RO
310C	VAR	Latched Fault	UNSIGNED8	RO
310D	VAR	Latched Faulted Module	UNSIGNED8	RW
310E	VAR	Latched Faulted Module (Before Break)	UNSIGNED8	RO
310F	VAR	Latched Faulted Module (After Break)	UNSIGNED8	RO
3110	VAR	Inline Power Status	UNSIGNED8	RO
3111	VAR	Inline Control Byte	UNSIGNED8	RW

B 6.5 Inline module objects

Index (hex)	Object	Name	Data type	Access
3200	ARRAY	Stored Module ID	UNSIGNED16	RO
3201	ARRAY	Current Module ID	UNSIGNED16	RO
3202	ARRAY	IN Data COP Index	UNSIGNED16	RO
3203	ARRAY	IN Data COP First Subindex	UNSIGNED8	RO
3204	ARRAY	IN Data COP Last Subindex	UNSIGNED8	RO
3205	ARRAY	OUT Data COP Index	UNSIGNED16	RO
3206	ARRAY	OUT Data COP First Subindex	UNSIGNED8	RO
3207	ARRAY	OUT Data COP Last Subindex	UNSIGNED8	RO

B 6.6 Inline special function module objects

Index (hex)	Object	Name	Data type	Access
3300	ARRAY	Special Func. Data Size	UNSIGNED8	RO
3301	ARRAY	Special Func. Status	BOOLEAN	RO
3302	ARRAY	Special Func. Data In (1 Byte)	UNSIGNED8	RO
3303	ARRAY	Special Func. Data Out (1 Byte)	UNSIGNED8	RWW
3304	ARRAY	Special Func. Data In (2 Bytes)	UNSIGNED16	RO
3305	ARRAY	Special Func. Data Out (2 Byte)	UNSIGNED16	RWW
3306	ARRAY	Special Func. Data In (3 Byte)	UNSIGNED24	RO
3307	ARRAY	Special Func. Data Out (3 Byte)	UNSIGNED24	RWW
3308	ARRAY	Special Func. Data In (4 Byte)	UNSIGNED32	RO
3309	ARRAY	Special Func. Data Out (4 Byte)	UNSIGNED32	RWW
330A	ARRAY	Special Func. Data In (6 Byte)	UNSIGNED48	RO
330B	ARRAY	Special Func. Data Out (6 Byte)	UNSIGNED48	RWW
330C	ARRAY	Special Func. Data In (8 Byte)	UNSIGNED64	RO
330D	ARRAY	Special Func. Data Out (8 Byte)	UNSIGNED64	RWW
330E	ARRAY	Special Func. Data In (>8 Byte)	UNSIGNED8	RO
330F	ARRAY	Special Func. Data Out (>8 Byte)	UNSIGNED8	RWW

B 6.7 Digital input objects

Index (hex)	Object	Name	Data type	Category
6000	ARRAY	Read Input 8-bit	UNSIGNED8	C: DI
6005	VAR	Global Int. Enable Dig.	BOOLEAN	O
6006	ARRAY	Int. Mask Any Change 8-bit	UNSIGNED8	O
6100	ARRAY	Read Input 16-bit	UNSIGNED16	O
6106	ARRAY	Int. Mask Any Change 16-bit	UNSIGNED16	O
6120	ARRAY	Read Input 32-bit	UNSIGNED32	O
6126	ARRAY	Int. Mask Any Change 32-bit	UNSIGNED32	O

B 6.8 Digital output objects

Index (hex)	Object	Name	Data type	Category
6200	ARRAY	Write Output 8-Bit	UNSIGNED8	C: DO
6206	ARRAY	Error Mode Output 8-Bit	UNSIGNED8	O
6207	ARRAY	Error State Output 8-Bit	UNSIGNED8	O
6300	ARRAY	Write Output 16-Bit	UNSIGNED16	O
6306	ARRAY	Error Mode Output 8-Bit	UNSIGNED16	O
6307	ARRAY	Error State Output 8-Bit	UNSIGNED16	O
6320	ARRAY	Write Output 32-Bit	UNSIGNED32	O
6326	ARRAY	Error Mode Output 32-Bit	UNSIGNED32	O
6227	ARRAY	Error State Output 32-Bit	UNSIGNED32	O

B 6.9 Analog input objects

Index (hex)	Object	Name	Data type	Category
6400	ARRAY	Read Analog Input 8-Bit	INTEGER8	O
6401	ARRAY	Read Analog Input 16-Bit	INTEGER16	C: AI

B 6.10 Analog output objects

Index (hex)	Object	Name	Data type	Category
6410	ARRAY	Write Analog Output 8-Bit	INTEGER8	O
6411	ARRAY	Write Analog Output 16-Bit	INTEGER16	C:AO

B 6.11 Analog input configuration objects

Index (hex)	Object	Name	Data type	Category
6421	ARRAY	AI Int. Trigger Selection	UNSIGNED8	O
6423	VAR	AI Global Int. Enable	BOOLEAN	C: AI
6424	ARRAY	AI Int. Upper Limit Integer	INTEGER32	O
6425	ARRAY	AI Int. Lower Limit Integer	INTEGER32	O
6426	ARRAY	AI Int. Delta Unsigned	UNSIGNED32	O

B 6.12 Analog output configuration objects

Index (hex)	Object	Name	Data type	Category
6443	ARRAY	Analog Output Error Mode	UNSIGNED8	O
6444	ARRAY	Analog Output Error Value	INTEGER32	O

B 7 Communication profile specific objects

B 7.1 Object 1000_{hex}: Device Type

Contains information about the device type. The object at index 1000_{hex} describes the type of device and its functionality. It is composed of a 16-bit field which describes the device profile that is used and a second 16-bit field which gives additional information about optional functionality of the device. The Additional Information parameter is device profile specific.

a. Object description

Index	1000_{hex}
Name	Device type
Object	VAR
Data type	UNSIGNED32
Category	Mandatory

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	00000191_{hex}

c. Byte description

MSB additional information

Specific functionality

	Bits 7—4	Not defined

I/O functionality

	Bit 3	AOP
	Bit 2	AIP
	Bit 1	DOP
	Bit 0	DIP

LSB general information, device profile (bits 7—0) 401

– DIP = 1, if any digital inputs are present
– DOP = 1, if any digital outputs are present
– AIP = 1, if any analog inputs are present
– AOP = 1, if any analog outputs are present

B 7.2 Object 1001_{hex}: Error Register

This object is an error register for the device. The device can map internal errors in this byte. This entry is mandatory for all devices. It is a part of an Emergency object.

a. Object description

Index	1001_{hex}
Name	Error register
Object	VAR
Data type	UNSIGNED8
Category	Mandatory

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

c. Structure of the error register

Bit	M/O	Supported	Meaning
0	M	Yes	Generic error
1	O	No	Current
2	O	Yes	Voltage
3	O	No	Temperature
4	O	Yes	Communication error (overrun, error state)
5	O	No	Device profile specific
6	O	No	Reserved (always 0)
7	O	Yes	Manufacturer specific

If a bit is set to 1 the specified error has occurred. The generic error is signaled at any error situation.

B 7.3 Object 1002_{hex}: Manufacturer Status Register

This object is a common status register for manufacturer specific purposes. In this document only the size and the location of this object are defined.

a. Object description

Index	1002_{hex}
Name	Manufacturer status register
Object	VAR
Data type	UNSIGNED32
Category	Optional

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	No

c. Byte description

3	**2**	**1**	**0**
Station Status	Station Status	Faulted Module	Reserved
Object 3101_{hex} (MSB)	Object 3101_{hex} (LSB)	Object 310_{hex}	

B 7.4 Object 1003_{hex}: Predefined Error Field

The object at index 1003_{hex} holds the errors that have occurred on the device and errors that have been signaled via the Emergency object. This in turn provides a history of errors related to a device.

– The entry at subindex 0 contains the number of actual errors that are recorded in the ARRAY starting at subindex 1.
– Every new error is stored at subindex 1, the older ones move down the list.
– Writing a "0" to subindex 0 deletes the entire error history (empties the ARRAY). Values higher than 0 are not allowed to write. These have to lead to an abort message (error code: 0609 0030_{hex}).
– The error numbers are of type UNSIGNED32 and are composed of a 16-bit error code and a 16-bit additional error information field which is manufacturer specific. The error code is contained in the lower 2 bytes (LSB) and the additional information is included in the upper 2 bytes (MSB). The additional information consists of the information stored in the Station Status Object (3101_{hex}). The length entry on subindex 0_{hex} and at least one error entry at subindex 1_{hex}.

a. MSB and LSB byte description

Additional information	**Error code**
Structure of the predefined error field	

b. Object description

Index	1003_{hex}
Name	Predefined error field
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

c. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number of errors
Entry category	Mandatory
Access	RW
PDO mapping	No
Value range	0 - 10
Default value	0

Subindex	$\mathbf{1}_{hex}$
Description	Standard error field
Entry category	Optional
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	No

Subindex	$\mathbf{2}_{hex}$ **–** $\mathbf{10}_{hex}$
Description	Standard error field
Entry category	Optional
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	No

B 7.5 Object 1005_{hex}: COB-ID SYNC message

Index 1005_{hex} defines the COB-ID of the Synchronization Object (SYNC). Further, it defines whether the device generates the SYNC.

a. Structure of SYNC COB-ID entry (UNSIGNED32)

MSB	←	←	←	←	←	LSB
Bits	**31**	**30**	**29**	**28-11**		**10-0**
11bitID	X	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-bit identifier	
29bitID	X	0/1	1	29-bit identifier		

b. Description of SYNC COB-ID entry

Bit number	Value	Meaning
31 (MSB)	X	Do not care
30	0	Device does not generate SYNC message
	1	Device generates SYNC message
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 – 11	0 X	If bit 29=0 if bit 29=1: bits 28-11 of 29-bit-SYNC-COB-ID
10-0 (LSB)	X	Bits 10-0 of SYNC-COB-ID

Bits 29 and 30 may be static (not changeable). If a device is not able to generate SYNC messages, an attempt to set bit 30 is responded with an abort message (abort code: $0609\ 0030_{hex}$). Devices supporting the standard CAN frame type only either ignore attempts to change bit 29 or respond with an abort message (abort code: $0609\ 0030_{hex}$). The first transmission of SYNC object starts within 1 sync cycle after setting bits 30 to 1. It is not allowed to change bits 0 - 29, while the object exists (bit 30 = 1).

c. Object description

Index	1005_{hex}
Name	COB-ID SYNC
Object	VAR
Data type	UNSIGNED32
Category	Conditional; mandatory, if PDO communication on a synchronous base is supported.

d. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	00000080_{hex}

B 7.6 Object 1008_{hex}: Manufacturer Device Name

Contains the manufacturer device name.

a. Object description

Index	1008_{hex}
Name	Manufacturer device name
Object	VAR
Data type	Visible String
Category	Optional

b. Entry description

Access	Const
PDO mapping	No
Value range	No
Default value	IL CAN BK-TC-PAC

B 7.7 Object 1009_{hex}: Manufacturer Hardware Version

Contains the manufacturer hardware version description of the device.

a. Object description

Index	1009_{hex}
Name	Manufacturer hardware version
Object	VAR
Data type	Visible String
Category	Optional

b. Entry description

Access	Const
PDO mapping	No
Value range	No
Default value	01

B 7.8 Object $100A_{hex}$: Manufacturer Software Version

Contains the manufacturer software version description.

a. Object description

Index	$100A_{hex}$
Name	Manufacturer software version
Object	VAR
Data type	Visible String
Category	Optional

b. Entry description

Access	Const
PDO mapping	No
Value range	No
Default value	1.00

B 7.9 Object $100C_{hex}$: Guard Time

The objects at Index $100C_{hex}$ and $100D_{hex}$ include the guard time in milliseconds and the life time factor.

The life time factor multiplied with the guard time gives the life time for the Life Guarding Protocol. It is 0 if not used.

a. Object description

Index	$010C_{hex}$
Name	Guard time
Object	VAR
Data type	UNSIGNED16
Category	Conditional; mandatory, if heartbeat is not supported

b.Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

B 7.10 Object $100D_{hex}$: Life Time Factor

The life time factor multiplied with the guard time gives the life time for the Node Guarding Protocol. It is 0 if not used.

a. Object description

Index	$100D_{hex}$
Name	Life time factor
Object	VAR
Data type	UNSIGNED8
Category	Conditional; mandatory, if heartbeat is not supported

b. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	0

B 7.11 Object 1010_{hex}: Store Parameters

This object supports the saving of parameters in non volatile memory. By reading access the device provides information about its saving capabilities. Several parameter groups are distinguished as:

– Subindex 0 contains the largest subindex that is supported.
– Subindex 1 refers to all parameters that can be stored on the device.
– Subindex 2 refers to communication related parameters (Index 1000_{hex} - $1FFF_{hex}$ manufacturer specific communication parameters).
– Subindex 3 refers to application related parameters (Index 6000_{hex} - $9FFF_{hex}$ manufacturer specific application parameters).
– At Subindex 4 - 127 manufacturers may store their choice of parameters individually.
– Subindex 128 - 254 are reserved for future use.

In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate subindex. The signature is "save".

a. Storage and write access signature ISO 8859 (ASCII)

MSB	←	←	LSB
e	**v**	**a**	**s**
65_{hex}	76_{hex}	61_{hex}	73_{hex}

On reception of the correct signature in the appropriate subindex the device stores the parameter and then confirms the SDO transmission (initiate download response). If the storing failed, the device responds with an Abort SDO Transfer (abort code: 0606 0000_{hex}).

If a wrong signature is written, the device refuses to store and responds with Abort SDO Transfer (abort code: 0800 $002x_{hex}$). On read access to the appropriate subindex the device provides information about its storage functionality with the following format:

b. Storage and read access signature (UNSIGNED32)

MSB	←	←	LSB
31-2		1	0
Reserved (=0)		0/1	0/1

c. Structure of the read access

Bit number	Value	Meaning
31-2	0	Reserved (= 0)
1	0	Device does not save parameters autonomously
	1	Device saves parameters autonomously
0	0	Device does not save parameters on command
	1	Device saves parameters on command

Autonomous saving means that a device stores the storable parameters in a nonvolatile manner without user request.

d. Object description

Index	1010_{hex}
Name	Store parameters

Object	ARRAY
Data type	UNSIGNED32
Category	Optional

e. Entry description

Subindex	**0**$_{hex}$
Description	Largest subindex supported
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	1_{hex} - $7F_{hex}$
Default value	4

Subindex	**1**$_{hex}$
Description	Save all parameters
Entry category	Mandatory
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Subindex	**2**$_{hex}$
Description	Save communication parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Subindex	**3**$_{hex}$
Description	Save application parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	02_{hex}

Subindex	**4**$_{hex}$
Description	Save I/O configuration and node address
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Application parameters are stored autonomously. Communication parameters must be stored manually.

B 7.12 Object 1011_{hex}: Restore default parameters

With this object the default values of parameters according to the communication or device profile are restored. By read access the device provides information about its capabilities to restore these values. Several parameter groups are distinguished as:

– Subindex 0 contains the largest subindex that is supported.
– Subindex 1 refers to all parameters that can be restored.
– Subindex 2 refers to communication related parameters (Index 1000_{hex} - $1FFF_{hex}$ manufacturer specific communication parameters).
– Subindex 3 refers to application related parameters (Index 6000_{hex} - $9FFF_{hex}$ manufacturer specific application parameters).
– At Subindex 4 - 127 manufacturers may restore their individual choice of parameters.
– Subindex 128 - 254 are reserved for future use.

In order to avoid the restoring of default parameters by mistake, restoring is only executed when a specific signature is written to the appropriate subindex. The signature is "load".

a. Restoring with write access signature (ASCII)

MSB	←	←	LSB
d	a	o	l
64_{hex}	61_{hex}	$6F_{hex}$	$6C_{hex}$

On reception of the correct signature in the appropriate subindex the device restores the default parameters and then confirms the SDO transmission (initiate download response).

If the storing failed, the device responds with an Abort SDO Transfer (abort code: $0606\ 0000_{hex}$). If a wrong signature is written, the device refuses to restore the defaults and responds with Abort SDO Transfer (abort code: $0800\ 002x_{hex}$).

The default values are set valid after the device is reset (reset node for Subindex 1_{hex} - $7F_{hex}$, reset communication for Subindex 2_{hex}) or power cycled.

On read access to the appropriate subindex the device provides information about its default parameter restoring capability with the following format:

b. Restoring default values and read access structure (UNSIGNED32)

MSB	←	←	LSB
31-1			0
Reserved (=0)			0/1

c. Structure of restore and read access

Bit number	Value	Meaning
31-1	0	Reserved (=0)
0	0	Device does not restore default parameters
	1	Device restores parameters

d. Object description

Index	1011_{hex}
Name	Restore default parameters
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

e. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Largest subindex supported
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	1_{hex} - $7F_{hex}$
Default value	4

Subindex	$\mathbf{1}_{hex}$
Description	Restore all default parameters
Entry category	Mandatory
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Subindex	$\mathbf{2}_{hex}$
Description	Restore communication default parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Subindex	$\mathbf{3}_{hex}$
Description	Restore application default parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

Subindex	**4_{hex} (see note)**
Description	Restore manufacturer specific parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	01_{hex}

The number and type of physically attached modules control most of the manufacturer specific parameters. Therefore Subindex 4 is reserved for future use.

B 7.13 Object 1012_{hex}: COB-ID Time Stamp Object

Index 1012_{hex} defines the COB-ID of the Time-Stamp Object (TIME). Further, it defines whether the device consumes the TIME or whether the device generates the TIME.

a. Structure of TIME COB-ID entry (UNSIGNED32)

MSB	←	←	←	←	←	←	←	**LSB**
Bits	**31**	**30**	**29**	**28-11**				**10-0**
11bitID	0/1	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			11-bit identifier	
29bitID	0/1	0/1	1	29-bit identifier				

b. Description of TIME COB-ID entry

Bit number	Value	Meaning
31 (MSB)	0	Device does not consume TIME message
	1	Device consumes TIME message
30	0	Device does not produce TIME message
	1	Device produces TIME message
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 – 11	0	If bit 29=0
	X	If bit 29=1: bits 28-11 of 29-bit-TIME-COB-ID
10-0 (LSB)	X	Bits 10-0 of TIME-COB-ID

Bits 29 and 30 may be static (not changeable). If a device is not able to generate TIME messages, an attempt to set bit 30 is responded with an abort message (abort code: 0609 0030_{hex}). Devices supporting the standard CAN frame type only, respond to an attempt to change bit 29 with an abort message (abort code: 0609 0030_{hex}). It is not allowed to change bits 0 - 29, while the object exists (bit 30 = 1).

c. Object description

Index	1012_{hex}
Name	COB-ID time stamp message
Object	VAR
Data type	UNSIGNED32
Category	Optional

d. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	100_{hex}

B 7.14 Object 1014_{hex}: COB-ID Emergency Object

Index 1014_{hex} defines the COB-ID of the Emergency Object (EMCY).

a. Structure of the EMCY Identifier entry (UNSIGNED32)

MSB	←	←	←	←	←	←	←	**LSB**
Bits	**31**	**30**	**29**	**28-11**				**10-0**
11bitID	0/1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0				11-bit identifier
29bitID	0/1	0	1	29-bit identifier				

b. Description of EMCY COB-ID entry

Bit number	**Value**	**Meaning**
31 (MSB)	0	EMCY exists / is valid
	1	EMCY does not exist / is not valid
30	0	Reserved (always 0)
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 - 11	0	If bit 29=0
	X	If bit 29=1: bits 28-11 of 29-bit-COB-ID
10-0 (LSB)	X	Bits 10-0 of COB-ID

Devices supporting the standard CAN frame type only, respond to an attempt to change bit 29 with an abort message (abort code: 0609 0030_{hex}). It is not allowed to change bits 0 -29, while the object exists (bit 31 = 0).

c. Object description

Index	1014_{hex}
Name	COB-ID Emergency message
Object	VAR
Data type	UNSIGNED32
Category	Conditional; mandatory, if emergency is supported

d. Entry description

Access	RO; optional RW
PDO mapping	No
Value range	UNSIGNED32
Default value	80h + Node-ID

B 7.15 Object 1015_{hex}: Inhibit Time EMCY

The inhibit time for the EMCY message can be adjusted via this entry. If this entry exists it must be writable in the object dictionary. The time has to be a multiple of 100 ms.

a. Object description

Index	1015_{hex}
Name	Inhibit Time EMCY
Object	VAR
Data type	UNSIGNED16
Category	Optional

b. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

B 7.16 Object 1016_{hex}: Consumer Heartbeat Time

The consumer heartbeat time defines the expected heartbeat cycle time and thus has to be higher than the corresponding producer heartbeat time configured on the device producing this heartbeat. Monitoring starts after the reception of the first heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The time has to be a multiple of 1 ms.

a. Structure of consumer heartbeat time entry (UNSIGNED32)

MSB	←	←	←	←	←	←	**LSB**
Bits		31-24			23-16		15-0
Value		Reserved (value: 00_{hex})			Node ID		Heartbeat time
Encoded as		-			UNSIGNED8		UNSIGNED16

When an attempt to configure several consumer heartbeat times unequal 0 for the same device ID is made, the device aborts the SDO download with abort code 0604 0043_{hex}.

b. Object description

Index	1016_{hex}
Name	Consumer heartbeat time
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

c. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number of entries
Entry category	Mandatory
Access	RO

PDO mapping	No
Value range	1 - 127
Default value	4

Subindex	**1_{hex}**
Description	Consumer heartbeat time
Entry category	Mandatory
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0

Subindex	**2_{hex} - 4_{hex}**
Description	Consumer heartbeat time
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	0

B 7.17 Object 1017_{hex}: Producer Heartbeat Time

The producer heartbeat time defines the cycle time of the heartbeat. The producer heartbeat time is 0 if it is not used. The time has to be a multiple of 1 ms.

a. Object description

Index	1017_{hex}
Name	Producer heartbeat time
Object	VAR
Data type	UNSIGNED16
Category	Conditional; mandatory, if guarding is not supported

b.Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	0

B 7.18 Object 1018_{hex}: Identity Object

The object at Index 1018_{hex} contains general information about the device. The Vendor ID (Subindex 1_{hex}) contains a unique value allocated to each manufacturer. The manufacturer-specific product code (Subindex 2_{hex}) identifies a specific device version. The manufacturer-specific revision number (Subindex 3_{hex}) consists of a major revision number and a minor revision number). The major revision number identifies a specific CANopen behavior. If the CANopen functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions of the same CANopen behavior.

a. Structure of revision number

MSB	←	←	←	←	**LSB**
31		16	15		0
Major revision number			Minor revision number		

The manufacturer-specific serial number (Subindex 4_{hex}) identifies a specific device.

b. Object description

Index	1018_{hex}
Name	Identity Object
Object	RECORD
Data type	Identity
Category	Mandatory

c. Entry description

Subindex	**0_{hex}**
Description	Number of entries
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	1 ... 4
Default value	4

Subindex	**1_{hex}**
Description	Vendor ID
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	00000084_{hex}

Subindex	**2_{hex}**
Description	Product code
Entry category	Optional
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	01000001_{hex}

Subindex	**3_{hex}**
Description	Revision number
Entry category	Optional
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	00010001_{hex}

Subindex	**4_{hex}**
Description	Serial number
Entry category	Optional
Access	RO
PDO mapping	No
Value range	UNSIGNED32
Default value	No

B 7.19 Object 1020_{hex}: Verify Configuration

If a device supports the saving of parameters in non-volatile memory, a network configuration tool or a CANopen manager can use this object to verify the configuration after a device reset and to check if a reconfiguration is necessary. The configuration tool shall store the date and time in that object and shall store the same values in DCF. Now the configuration tool lets the device save its configuration by writing to Index 1010_{hex}, Subindex 1_{hex} the signature "save". After a reset the device shall restore the last configuration and the signature automatically or by reset.

If any other command changes the boot-up configuration values, the device shall reset the Verify Configuration Object to 0. The Configuration Manager compares signature and configuration with the value from the DCF and decides if a reconfiguration is necessary or not.

Index	Object	Name	Type	Access	M/O
1020_{hex}	ARRAY	Verify Configuration	UNSIGNED32	RW	O

The sub-objects for the Verify Configuration Object are:

Index	Subindex	Field in configuration verify	Data type
1020_{hex}	0_{hex}	Number of supported entries	UNSIGNED8
	1_{hex}	Configuration date	UNSIGNED32
	2_{hex}	Configuration time	UNSIGNED32

Configuration data shall contain the number of days since January 1, 1984. Configuration time shall be the number of ms after midnight.

Application hint: The usage of this object allows a significant speedup of the boot-up process. If it is used, the system integrator has to consider that a user may change a configuration value and afterwards activate the Store Configuration command, 1010_{hex} without changing the value of 1020_{hex}. So the system integrator has to ensure a 100% consequent usage of this feature.

B 7.20 Object 1027_{hex}: Module List

a. Modular devices

A common method to provide modular devices is the usage of a bus coupler that allows to connect several combinations of modules. Object 1027_{hex}, Module List, describes the concretely attached modules.

b. Object description

Index	Object	Name	Type	Access	M/O
1027_{hex}	ARRAY	Module List	UNSIGNED 16	RO	Mandatory for modular devices, otherwise optional

The sub-objects for the Module List Object are:

c. Entry description

Index	Subindex	Field in configuration verify	Data type
1027_{hex}	0_{hex}	Number of connected modules	UNSIGNED8
1027_{hex}	1_{hex}	Module 1	UNSIGNED16
1027_{hex}	…	…	…
1027_{hex}	N	Module N	UNSIGNED16

The corresponding subindexes ($1 \leq N \leq 254$) describe the corresponding modules in the order they are attached. Each entry contains a number that identifies the module. For this the number must be unique within all module types that can be attached to this bus coupler device type.

d. Module list

High byte	Low byte
Data Length Code	Module ID

B 7.21 Object 1029_{hex}: Error Behavior object

If a serious device failure is detected in operation state, the module shall enter by default autonomously the preoperational state. If object 1029_{hex} is implemented, the device can be configured to enter alternatively the stopped state or remain in the current state in case of a device failure. Device failures shall include the following communication errors:

– Bus-OFF conditions of the CAN interface
– Life guarding event with the state 'occurred'
– Heartbeat event with the state 'occurred'

Serious device errors can also be caused by device internal failures. The value of the error classes is as follows:

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

3 ... 127 = Reserved

a. Object description

Index	1029_{hex}
Name	Error Behavior
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	No. of error classes
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	1_{hex}
Default value	1

Subindex	$\mathbf{1}_{hex}$
Description	Communication error
Entry category	Mandatory
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	0

B 7.22 Object 1200_{hex} – 1201_{hex}: Server SDO Parameter

In order to describe the SDOs used on a device the data type SDO Parameter is introduced. The data type has the Index 22_{hex} in the object dictionary. The number of supported entries in the SDO object record is specified at Subindex 0_{hex}. The values at 1_{hex} and 2_{hex} specify the COB-ID for this SDO. Subindex 3 gives the server of the SDO, in case the record describes and SDO for which the device is client, and gives the client of the SDO if the record describes an SDO for which the device is server.

a. Structure of SDO COB-ID entry (UNSIGNED32)

MSB	←	←	←	←	←	←	←	**LSB**
Bits	**31**	**30**	**29**	**28-11**				**10-0**
11-bit-ID	0/1	0	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0				11-bit identifier
29-bit-ID	0/1	0	1	29-bit identifier				

a. Description of SDO COB-ID entry

Bit number	Value	Meaning
31 (MSB)	0	SDO exists / is valid
	1	SDO does not exist / is not valid
30	0	Reserved (always 0)
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 – 11	0	If bit 29=0
	X	If bit 29=1: bits 28-11 of 29-bit-COB-ID
10-0 (LSB)	X	Bits 10-0 of COB-ID

An SDO is only valid if both SDO valid bits are 0. Devices supporting the standard CAN frame type only, respond to an attempt to change bit 29 with an abort message (abort code: $0609\ 0030_{hex}$). These objects contain the parameters for the SDOs for which the device is the server. If a device handles more than one server SDO the default SDO must be locates at Index 1200_{hex} as the first server SDO. This entry is read only 2. All additional server SDOs are invalid by default (invalid bit - see description of SDO COB-ID entry table), their description is located at subsequent indexes. It is not allowed to change the COB-ID while the SDO exists.

The SDO client description (Subindex 3_{hex}) is optional. It is not available for the default SDO (no Subindex 3_{hex} at Index 1200_{hex}), as this entry is read only.

c. Object description

Index	1200_{hex}—1201_{hex}
Name	Server SDO parameter
Object	RECORD
Data type	SDO parameter
Category	Conditional – Index 1200_{hex}: Optional – Index 1201_{hex}—$127F_{hex}$: Mandatory for each additionally supported server SDO

d. Entry description

Subindex	**0**$_{hex}$
Description	Number of entries
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	Index 1200_{hex}: 2 Index 1201_{hex} - $127F_{hex}$: 2 – 3
Default value	No

Subindex	**1**$_{hex}$
Description	COB-ID Client->Server (rx)
Entry category	Mandatory
Access	Index 1200_{hex}: RO, Index 1201_{hex}-$127F_{hex}$: RW
PDO mapping	No
Value range	UNSIGNED32 (Table 53)
Default value	Index 1200_{hex}: 600h+Node-ID, Index 1201_{hex}-$127F_{hex}$: No

Subindex	**2**$_{hex}$
Description	COB-ID Server -> Client (tx)
Entry category	Mandatory
Access	Index 1200_{hex}: RO Index 1201_{hex}-$127F_{hex}$: RW
PDO mapping	No
Value range	UNSIGNED32 (Table 53)
Default value	Index 1200_{hex}: 580_{hex}+Node-ID, Index 1201_{hex}-$127F_{hex}$: No

B 7.23 Object 1400_{hex} – $141F_{hex}$: Receive PDO Communication Parameter

Contains the mapping for the PDOs the device is able to receive. The type of the PDO communication parameter (20_{hex}). The Subindex 0_{hex} contains the number of valid entries within the communication record. Its value is at least 2. If inhibit time is supported the value is 3. Subindex 1_{hex} contains the COB-ID of the PDO. This entry has been defined as UNSIGNED32 in order to cater for 11-bit CAN identifiers (CAN 2.0A) as well as for 29-bit CAN identifiers (CAN 2.0B).

The entry has to be interpreted as defined in structure of PDO COB-ID entry table and description of POD COB-ID entry table.

a. Structure of PDO COB-ID entry (UNSIGNED32)

MSB	←	←	←	←	←	←	←	**LSB**
Bits	**31**	**30**	**29**	**28-11**				**10-0**
11bitID	0/1	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0				11-bit identifier
29bitID	0/1	0/1	1	29-bit identifier				

b. Description of PDO COB-ID entry

Bit number	Value	Meaning
31 (MSB)	0	PDO exists / is valid
	1	PDO does not exist / is not valid
30	0	RTR allowed on this PDO
	1	No RTR allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28 – 11	0	If bit 29 = 0
	X	If bit 29 = 1: bits 28-11 of 29-bit-COB-ID
10-0 (LSB)	X	Bits 10-0 of COB-ID

The PDO valid/not valid allows to select which PDOs are used in the operational state. There may be PDOs fully configured (e.g. by default) but not used, and therefore set to "not valid" (deleted). This feature is necessary for devices supporting more than four RPDOs or four TPDOs, because each device has only default identifiers for the first four RPDOs/TPDOs. Devices supporting the standard CAN frame type only or do not support Remote Frames, respond to an attempt to change bit 29 to 1 or bit 30 to 0 with an abort message (abort code: 0609 0030_{hex}). It is not allowed to change bits 0 - 29, while the PDO exists (bit 31 = 0).

The transmission type (Subindex 2) defines the transmission/reception character of the PDO. On an attempt to change the value of the transmission type to a value that is not supported by the device an abort message (abort code: 0609 0030_{hex}) is generated.

c. Transmission types (PDO transmission)

	Cyclic	Acyclic	Synchronous	Asynchronous	RTR only
0	-	X	X	-	-
1-240	X	-	X	-	-
241-251	Reserved				
252	-	-	X	-	X
253	-	-	-	X	X
254	-	-	-	X	-
255	-	-	-	X	-

Synchronous (transmission types 0 - 240 and 252) means that the transmission of the PDO shall be related to the SYNC object. Preferably the devices use the SYNC as a trigger to output or actuate based on the previous synchronous Receive PDO respectively to update the data transmitted at the following synchronous Transmit PDO. Details of this mechanism depend on the device type and are defined in the device profile if applicable.

Asynchronous means that the transmission of the PDO is not related to the SYNC object. A transmission type of zero means that the message shall be transmitted synchronously with the SYNC object but not periodically.

A value between 1 and 240 means that the PDO is transferred synchronously and cyclically. The transmission type indicating the number of SYNC which are necessary to trigger PDO transmissions.

Receive PDOs are always triggered by the following SYNC upon reception of data independent of the transmission types 0 - 240.

The transmission types 252 and 253 mean that the PDO is only transmitted on remote transmission request. At transmission type 252, the data is updated (but not sent) immediately after reception of the SYNC object. At transmission type 253, the data is updated at the reception of the remote transmission request (hardware and software restrictions may apply). These values are only possible for TPDOs.

For TPDOs transmission type 254 means, the application event is manufacturer specific (manufacturer specific part of the object dictionary). Transmission type 255 means the application event is defined in the device profile. RPDOs with the type trigger the update of the mapped data with the reception.

Subindex 3_{hex} contains the inhibit time. This time a minimum interval for PDO transmission. The value is defined as multiple of 100 ms. It is not allowed to change the value while the PDO exists (bit 31 of Subindex 1 = 0).

Subindex 4_{hex} is reserved. It does not have to be implemented, in this case read or write access leads to Abort SDO Transfers (abort code: 0609 0011_{hex}).

In mode 254/255 additionally an event time can be used for TPDO. If an event timer exists for a TPDO (value not equal to 0) the elapsed timer is considered to be an event. The event timer elapses as multiple of 1 ms of the entry in Subindex 5_{hex} of the TPDO. This event will cause the transmission of this TPDO in addition to otherwise defined events. The occurrence of the events set the timer.

Independent of the transmission type the RPDO event timer is used to recognize the expiration of the RPDO.

d. Object description

Index	1400_{hex} — $141F_{hex}$
Name	Receive PDO parameter
Object	RECORD
Data type	PDO CommPar
Category	Conditional; mandatory for each supported PDO

e. Entry description

Subindex	**0**$_{hex}$
Description	Largest subindex supported
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	2 – 5

Subindex	**1**$_{hex}$
Description	COB-ID used by PDO
Entry category	Mandatory
Access	RO; RW if variable COB-ID is supported
PDO mapping	No
Value range	UNSIGNED32 (Table 54)
Default value	Index 1400_{hex}: 200_{hex} + Node-ID, Index 1401_{hex}: 300_{hex} + Node-ID, Index 1402_{hex}: 400_{hex} + Node-ID, Index 1403_{hex}: 500_{hex} + Node-ID, Index 1404_{hex} - $15FF_{hex}$: Disabled

Subindex	**2**$_{hex}$
Description	Transmission type
Entry category	Mandatory
Access	RO; RW if variable transmission type is supported
PDO mapping	No
Value range	UNSIGNED8 (Table 55)
Default value	(Device profile dependent)

Subindex	**3**$_{hex}$
Description	Inhibit time (not used for RPDO)
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	No

Subindex	**4_{hex}**
Description	Compatibility entry
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	No

Subindex	**5_{hex}**
Description	Event timer
Entry category	Optional (not used for RPDO)
Access	RW
PDO mapping	No
Value range	0 = not used UNSIGNED16
Default value	No

B 7.24 Object 1600_{hex} – $161F_{hex}$: Receive PDO Mapping Parameter

Contains the mapping for the PDOs (parameter 21_{hex}) the device is able to receive. The Subindex 0_{hex} contains the number of valid entries within the mapping record. This number of entries is also the number of the application variable which shall be transmitted/received with the corresponding PDO.

The Subindexes 1_{hex} to the number of entries contain the information about the mapped application variables. These entries describe the PDO contents by their index, subindex and length. All three values are hexadecimal coded. The length entry contains the length of the object in bit (1_{hex} ... 40_{hex}). This parameter can be used to verify the overall mapping length. It is mandatory.

The structure of the entries from Subindex 1_{hex} — 40_{hex} is as follows:

a. Structure of PDO mapping entry

Upper word:	Index (16 bit)
Lower word, upper byte:	Subindex (8 bit)
Lower word, lower byte:	Object length (8 bit)

If the change of the PDO mapping cannot be executed (e.g. the PDO length is exceeded or the SDO client attempts to map an object that cannot be mapped) the device responds with an Abort SDO Transfer Service.

Subindex 0 determines the value number of objects that have been mapped. For changing the PDO mapping first the PDO has to be deleted and Subindex 0 must be set to 0 (mapping is deactivated). Then the objects can be remapped. When a new object is mapped by writing a subindex between 1 and 64, the device may check whether the object specified by index/subindex exists. If the object does not exist or the object cannot be mapped, the SDO transfer must be aborted with the Abort SDO Transfer Service with one of the abort codes 0602 0000_{hex} or 0604 0041_{hex}.

After all objects are mapped Subindex 0 is set to the valid number of mapped objects. Finally the PDO will be created by writing to its communication parameter COB-ID. When Subindex 0 is set to a value >0 the device may validate the new PDO mapping before transmitting the response of the SDO service. If an error is detected the device has to transmit the Abort SDO Transfer Service with one of the abort codes 0602 0000_{hex}, 0604 0041_{hex} or 0604 0042_{hex}. When Subindex 0 is read the actual number of valid mapped objects is returned.

If data types (Index 1_{hex} —7_{hex}) are mapped they serve as "dummy entries". The corresponding data in the PDO is not evaluated by the device. This optional feature is useful e.g. to transmit data to several devices using one PDO, each device only utilizing a part of the PDO. It is not possible to create a dummy mapping for a TPDO.

b. Object description

Index	1600_{hex} – $161F_{hex}$
Name	Receive PDO mapping
Object	RECORD
Data type	PDO mapping
Category	Conditional; mandatory for each supported PDO

c. Entry description

Subindex	**0_{hex}**
Description	Number of mapped application objects in PDO
Entry category	Mandatory
Access	RO; RW if dynamic mapping is supported
PDO mapping	No
Value range	0: Deactivated 1 - 64: Activated
Default value	(Device profile dependent)

Subindex	**1_{hex} - 40_{hex}**
Description	PDO mapping for the nth application object to be mapped
Entry category	Conditional, depends on number and size of objects to be mapped
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	(Device profile dependent)

B 7.25 Object 1800_{hex} – $181F_{hex}$: Transmit PDO Communication Parameter

Contains the mapping for the PDOs the device is able to transmit. A description of the entries can be found in Section "Object 1400_{hex} – $141F_{hex}$: Receive PDO Communication Parameter" on page B-35.

a. Object description

Index	1800_{hex} — $181F_{hex}$
Name	Transmit PDO Communication Parameter
Object	RECORD
Data type	PDO CommPar
Category	Conditional; mandatory for each supported PDO

b. Entry description

Subindex	**0_{hex}**
Description	Largest subindex supported
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	2 – 5

Subindex	**1_{hex}**
Description	COB-ID used by PDO
Entry category	Mandatory
Access	RO; RW if COB-ID can be configured

PDO mapping	No
Value range	UNSIGNED32 (Figure 65)
Default value	Index 1800_{hex}: 180_{hex} + Node-ID, Index 1801_{hex}: 280_{hex} + Node-ID, Index 1802_{hex}: 380_{hex} + Node-ID, Index 1803_{hex}: 480_{hex} + Node-ID, Index 1804_{hex} - $18FF_{hex}$: Disabled

Subindex	**2_{hex}**
Description	Transmission type
Entry category	Mandatory
Access	RO; RW if transmission type can be changed
PDO mapping	No
Value range	UNSIGNED8 (Table 54)
Default value	(Device profile dependent)

Subindex	**3_{hex}**
Description	Inhibit time
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED16
Default value	No

Subindex	**4_{hex}**
Description	Reserved
Entry category	Optional
Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	No

Subindex	**5_{hex}**
Description	Event timer
Entry category	Optional
Access	RW
PDO mapping	No
Value range	0 = not used UNSIGNED16
Default value	No

B 7.26 Object $1A00_{hex}$ – $1A1F_{hex}$: Transmit PDO Mapping Parameter

Contains the mapping for the PDOs, which the device can transmit. A detailed description of the entries can be found in Section "Object 1600_{hex} – $161F_{hex}$: Receive PDO Mapping Parameter" on page B-39.

a. Object description

Index	$1A00_{hex}$ — $1A1F_{hex}$
Name	Transmit PDO Mapping Parameter
Object	RECORD
Data type	PDO mapping
Category	Conditional; mandatory for each supported PDO

b. Entry description

Subindex	**0_{hex}**
Description	Number of mapped application objects in PDO
Entry category	Mandatory
Access	RO; RW if dynamic mapping is supported
PDO mapping	No
Value range	0: Deactivated 1 - 64: Activated
Default value	(Device profile dependent)

Subindex	**1_{hex} - 40_{hex}**
Description	PDO mapping for the n-th application object to be mapped
Entry category	Conditional, depends on number and size of objects to be mapped
Access	RW
PDO mapping	No
Value range	UNSIGNED32
Default value	(Device profile dependent)

c. Destination Address Mode (DAM)

The *addr* and *m* fields of the MPDO refer to the consumer. They allow access to the consumer's object dictionary in an SDO-like manner. With *addr* = 0, it allows multicasting and broadcasting to write into the object dictionaries of more than one node simultaneously, without having a PDO for each single object. Initiating a DAM-MPDO is application-dependent, like it is for SDOs.

d. Source Address Mode (SAM)

The *addr* and *m* fields of the MPDO refer to the producer. Only one producer MPDO of this type is allowed for each node. Transmission type has to be 254 or 255.

The producer uses an Object Scanner List in order to know which objects are to send. The consumer uses an Object Dispatcher List as a "cross reference" 3. The restriction about using 32-bit transfers only will not present problems in practice since all of the participating devices know the data types (and sizes) of their related objects.

e. Object dictionary entries

PDO mapping record

The meaning of Subindex 0 is that the number of mapped objects is extended. The valid range for non-multiplexed PDOs is 0 to 64. A value of 255 indicates a DAM-MPDO, a value of 254 indicates a SAM-MPDO.

For Source Address Mode (SAM) further entries in the Mapping Record (MR) are ignored.

For Destination Address Mode (DAM) the first object describes the local object (there can be mapped only one object into an MPDO).

Index	Object	Name	Type
$16XX_{hex}$— $1AXX_{hex}$	0_{hex}	Number of mapped objects in the PDO: 0 ... 64: Valid range for number of mapped objects 254: Formatted as SAM- MPDO 255: formatted as DAM- MPDO.	UNSIGNED8

B 8 Predefinitions

B 8.1 Introduction

If a device supports a specific type of I/O functionality (analog/digital I/O) it shall support the related default PDOs. However, the module can support additional manufacturer-specific PDOs. If variable mapping is supported the PDO default settings can be changed by means of configuration. There shall be up to 4 enabled TPDOs and up to 4 enabled RPDOs with default mappings. If a module does not support a specific I/O function, the related default PDOs shall remain unused. If a device supports more than the default digital input or output channels, the related analog default PDOs shall remain unused and the additional digital I/Os can use additional PDOs. This shall be the same for additional analog channels.

All TPDOs with transmission type 255 shall be transmitted when entering the OPERATIONAL state.

B 8.2 PDO mapping

a. 1st RPDO mapping (digital outputs)

This RPDO receives asynchronously the values of maximum 64 digital outputs at an I/O module. The default transmission type shall be 255. The default values of the mapped outputs are described in the Default State objects.

After power-on and application reset these default objects are valid.

Index	Subindex	Comment	Default value
1600_{hex}	0_{hex}	Number of mapped objects	No
1600_{hex}	1_{hex}	1^{st} object to be mapped	6200 01 08_{hex}
1600_{hex}	2_{hex}	2^{nd} object to be mapped	6200 02 08_{hex}
1600_{hex}	3_{hex}	3^{rd} object to be mapped	6200 03 08_{hex}
1600_{hex}	4_{hex}	4^{th} object to be mapped	6200 04 08_{hex}
1600_{hex}	5_{hex}	5^{th} object to be mapped	6200 05 08_{hex}
1600_{hex}	6_{hex}	6^{th} object to be mapped	6200 06 08_{hex}
1600_{hex}	7_{hex}	7^{th} object to be mapped	6200 07 08_{hex}
1600_{hex}	8_{hex}	8^{th} object to be mapped	6200 08 08_{hex}

The number of mapped objects into the PDO depends on the hardware.

b. 1st TPDO mapping (digital inputs)

This TPDO transmits event-driven the values of maximum 64 digital inputs. The default transmission type shall be 255; the default values for inhibit and event timer are 0. If one digital input changes its value, this PDO shall be transmitted immediately. If an interrupt mask is enabled, the PDO shall be transmitted only if the interrupt condition is fulfilled.

Index	Subindex	Comment	Default value
$1A00_{hex}$	0_{hex}	Number of mapped objects	No
$1A00_{hex}$	1_{hex}	1^{st} object to be mapped	6000 01 08_{hex}
$1A00_{hex}$	2_{hex}	2^{nd} object to be mapped	6000 02 08_{hex}
$1A00_{hex}$	3_{hex}	3^{rd} object to be mapped	6000 03 08_{hex}
$1A00_{hex}$	4_{hex}	4^{th} object to be mapped	6000 04 08_{hex}
$1A00_{hex}$	5_{hex}	5^{th} object to be mapped	6000 05 08_{hex}
$1A00_{hex}$	6_{hex}	6^{th} object to be mapped	6000 06 08_{hex}
$1A00_{hex}$	7_{hex}	7^{th} object to be mapped	6000 07 08_{hex}
$1A00_{hex}$	8_{hex}	8^{th} object to be mapped	6000 08 08_{hex}

The number of mapped objects into the PDO depends on the hardware.

c. 2nd RPDO mapping (analog outputs)

This RPDO receives asynchronously the 16-bit values of maximum 4 analog outputs to the module. The default transmission type shall be 255. The default values of the mapped outputs are described in the Default State objects.

After power-on and application reset these default objects are valid.

Index	Subindex	Comment	Default value
1601_{hex}	0_{hex}	Number of mapped objects	No
1601_{hex}	1_{hex}	1^{st} object to be mapped	6411 01 10_{hex}
1601_{hex}	2_{hex}	2^{nd} object to be mapped	6411 02 10_{hex}
1601_{hex}	3_{hex}	3^{rd} object to be mapped	6411 03 10_{hex}
1601_{hex}	4_{hex}	4^{th} object to be mapped	6411 04 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

d. 2nd TPDO mapping (analog inputs)

This TPDO transmits event-driven the 16-bit values of maximum 4 analog inputs. The default transmission type shall be 255; the default values for inhibit and event timer are 0. By default interrupt source (object 6423_{hex}) is disabled. If one analog input changes its value and object 6423_{hex} is enabled, the PDO shall be transmitted immediately. If an analog interrupt condition is enabled, the PDO shall be transmitted only if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO shall be transmitted if one of these conditions is fulfilled.

Index	Subindex	Comment	Default value
$1A01_{hex}$	0_{hex}	Number of mapped objects	No
$1A01_{hex}$	1_{hex}	1^{st} object to be mapped	6401 01 10_{hex}
$1A01_{hex}$	2_{hex}	2^{nd} object to be mapped	6401 02 10_{hex}
$1A01_{hex}$	3_{hex}	3^{rd} object to be mapped	6401 03 10_{hex}
$1A01_{hex}$	4_{hex}	4^{th} object to be mapped	6401 04 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

e. 3rd RPDO mapping (analog outputs)

This RPDO receives asynchronously the 16-bit values of maximum 4 analog outputs to the module. The default transmission type shall be 255.

Index	Subindex	Comment	Default value
1602_{hex}	0_{hex}	Number of mapped objects	No
1602_{hex}	1_{hex}	1^{st} object to be mapped	6411 05 10_{hex}
1602_{hex}	2_{hex}	2^{nd} object to be mapped	6411 06 10_{hex}
1602_{hex}	3_{hex}	3^{rd} object to be mapped	6411 07 10_{hex}
1602_{hex}	4_{hex}	4^{th} object to be mapped	6411 08 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

f. 3rd TPDO mapping (analog inputs)

This TPDO transmits event-driven the 16-bit values of maximum 4 analog inputs. The default transmission type shall be 255. By default interrupt source (object 6423_{hex}) is disabled. If one analog input changes its value and object 6423_{hex} is enabled, the PDO shall be transmitted immediately. If an analog interrupt condition is enabled, the PDO shall be transmitted only if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO shall be transmitted if one of these conditions is fulfilled.

Index	Subindex	Comment	Default value
$1A02_{hex}$	0_{hex}	Number of mapped objects	No
$1A02_{hex}$	1_{hex}	1^{st} object to be mapped	6401 05 10_{hex}
$1A02_{hex}$	2_{hex}	2^{nd} object to be mapped	6401 06 10_{hex}
$1A02_{hex}$	3_{hex}	3^{rd} object to be mapped	6401 07 10_{hex}
$1A02_{hex}$	4_{hex}	4^{th} object to be mapped	6401 08 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

g. 4th RPDO mapping (analog outputs)

This RPDO receives asynchronously the 16-bit values of maximum 4 analog outputs to the module. The default transmission type shall be 255.

Index	Subindex	Comment	Default value
1603_{hex}	0_{hex}	Number of mapped objects	No
1603_{hex}	1_{hex}	1^{st} object to be mapped	6411 09 10_{hex}
1603_{hex}	2_{hex}	2^{nd} object to be mapped	6411 0A 10_{hex}
1603_{hex}	3_{hex}	3^{rd} object to be mapped	6411 0B 10_{hex}
1603_{hex}	4_{hex}	4^{th} object to be mapped	6411 0C 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

h. 4th TPDO mapping (analog inputs)

This TPDO transmits event-driven the 16-bit values of maximum 4 analog inputs. The default transmission type shall be 255. By default interrupt source (object 6423_{hex}) is disabled. If one analog input changes its value and object 6423_{hex} is enabled, the PDO shall be transmitted immediately. If an analog interrupt condition is enabled, the PDO shall be transmitted only if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO shall be transmitted if one of these conditions is fulfilled.

Index	Subindex	Comment	Default value
$1A03_{hex}$	0_{hex}	Number of mapped objects	No
$1A03_{hex}$	1_{hex}	1^{st} object to be mapped	6401 09 10_{hex}
$1A03_{hex}$	2_{hex}	2^{nd} object to be mapped	6401 0A 10_{hex}
$1A03_{hex}$	3_{hex}	3^{rd} object to be mapped	6401 0B 10_{hex}
$1A03_{hex}$	4_{hex}	4^{th} object to be mapped	6401 0C 10_{hex}

The number of mapped objects into the PDO depends on the hardware.

B 9 Detailed specification of DS-401 defined object dictionary

Each I/O module compliant with this device profile shall share the CANopen object dictionary entries from 6000_{hex} to $67FF_{hex}$. These entries are common to all I/O modules and each module only implements those objects relevant to its functions.

B 10 Digital input modules

B 10.1 Object 6000_{hex}: Read Digital Input 8-Bit

This object shall read groups of 8 input lines as 8-bit information. A maximum of 254 x 8-bit inputs is addressable (2032 inputs). This object is mandatory for digital input modules and shall support all implemented input lines.

a. Object description

Index	6000_{hex}
Name	Read input 8 bit
Object	ARRAY
Data type	UNSIGNED8
Category	Conditional: Device with digital inputs

b. Entry description

Subindex	**0**$_{hex}$
Description	Number of input 8 bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1**$_{hex}$
Description	Read Input 1_{hex} to 8_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Default
Value range	UNSIGNED8
Default value	No

Subindex	**2_{hex} ... 8_{hex}**
Description	Read Input 9_{hex} to 10_{hex} ... Read Input 39_{hex} to 40_{hex}
Access	RO
Entry category	Optional
PDO mapping	Default
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**9_{hex} ... FE_{hex}**
Description	Read Input 41_{hex} to 48_{hex} ... Read Input Read Input $7E8_{hex}$ to $7F0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	No

CANopen presents the data to the user on multi-byte accesses as least significant byte to most significant byte. The user can access the same data using 8 bit and multi-byte accesses (such as 6000_{hex} and 6100_{hex}). The 8-bit access on the odd subindexes will indicate the lower 8 bits of the 16-bit access and even subindexes will indicate the upper 8 bits of the 16 bit access.

B 10.2 Object 6005_{hex}: Global Interrupt Enable Digital 8-Bit

This object shall enable and disable globally the interrupt behavior without changing the interrupt masks. In event-driven mode the device transmits the input values depending on the interrupt masks in objects 6006_{hex}, 6007_{hex} and 6008_{hex} (resp. 6050_{hex} .. 6057_{hex}, 6060_{hex} .. 6067_{hex}, 6070_{hex} .. 6077_{hex}, or 6106_{hex}, 6107_{hex}, 6108_{hex}, or 6126_{hex}, 6127_{hex}, 6128_{hex}) and the PDO transmission type.

TRUE = Global interrupt enabled

FALSE = Global interrupt disabled

a. Object description

Index	6005_{hex}
Name	Global Interrupt Enable Digital 8-Bit
Object	Variable
Data type	BOOLEAN
Category	Optional

b. Entry description

Subindex	0_{hex}
Access	RW
PDO mapping	No
Value range	BOOLEAN
Default value	TRUE

B 10.3 Object 6006_{hex}: Interrupt Mask Any Change 8-Bit

This object determines which input port lines shall activate an interrupt by positive and/or negative edge detection.

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6006_{hex}
Name	Interrupt Mask Any Change 8-Bit
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Input 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Interrupt Any Change 1_{hex} to 8_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	FF_{hex}

Subindex	**2_{hex}**
Description	Interrupt Any Change 9_{hex} to 10_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	FF_{hex}
...	...
Subindex	**FE_{hex}**
Description	Interrupt Any Change $7E8_{hex}$ to $7F0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	FF_{hex}

B 10.4 Object 6100_{hex}: Read Input 16-Bit

This object shall read groups of 16 input lines as 16-bit information. A maximum of 254 x 16-bit inputs is addressable (4064 inputs).

a. Object description

Index	6100_{hex}
Name	Read Input 16-Bit
Object	ARRAY
Data type	UNSIGNED16
Category	Conditional: optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of input 16 bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Read Input 1_{hex} to 10_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

Subindex	**2_{hex}**
Description	Read Input 11_{hex} to 20_{hex}
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**FE_{hex}**
Description	Read Input $FD0_{hex}$ to $FE0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

B 10.5 Object 6106_{hex}: Interrupt Mask Any Change 16-Bit

This object determines which input port lines shall activate an interrupt by positive and/or negative edge detection.

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6106_{hex}
Name	Interrupt Mask Any Change 16-Bit
Object	ARRAY
Data type	UNSIGNED16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Input 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Interrupt Any Change 1_{hex} to 10_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$

Subindex	**2_{hex}**
Description	Interrupt Any Change 11_{hex} to 20_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$
...	...
Subindex	**FE_{hex}**
Description	Interrupt Any Change $FD1_{hex}$ to $FE0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$

B 10.6 Object 6120_{hex}: Read Input 32-Bit

This object shall read groups of 16 input lines as 32-bit information. A maximum of 254 x 32-bit inputs is addressable (8128 inputs).

a. Object description

Index	6120_{hex}
Name	Read Input 32-Bit
Object	ARRAY
Data type	UNSIGNED32
Category	Conditional: optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Input 32 Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Read Input 1_{hex} to 20_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No

Subindex	**2_{hex}**
Description	Read Input 21_{hex} to 40_{hex}
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No
...	...
Subindex	**FE_{hex}**
Description	Read Input $1FA0_{hex}$ to $1FC0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No

B 10.7 Object 6126_{hex}: Interrupt Mask Any Change 32-Bit

This object determines which input port lines shall activate an interrupt by positive and/or negative edge detection.

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6126_{hex}
Name	Interrupt Mask Any Change 32-Bit
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Input 32-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Interrupt Any Change 1_{hex} to 20_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED32
Default value	FFFF $FFFF_{hex}$

Subindex	**2_{hex}**
Description	Interrupt Any Change 21_{hex} to 40_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED32
Default value	FFFF $FFFF_{hex}$
...	...
Subindex	**FE_{hex}**
Description	Interrupt Any Change $1FA1_{hex}$ to $1FC0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED32
Default value	FFFF $FFFF_{hex}$

B 11 Digital output modules

B 11.1 Object 6200_{hex}: Write Output 8-Bit

This object shall set a group of 8 output lines as a byte of information. A maximum of 254 x 8-bit output blocks is addressable.

a. Object description

Index	6200_{hex}
Name	Write Output 8-Bit
Object	ARRAY (8_{hex})
Data type	UNSIGNED8
Category	Conditional: Device with digital outputs

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Write Output 1_{hex} to 8_{hex}
Access	RWW
Entry category	Mandatory
PDO mapping	Default
Value range	UNSIGNED8
Default value	0_{hex}

Subindex	**2_{hex} .. 8_{hex}**
Description	Write Output 9_{hex} to 10_{hex} ... Write Output 39_{hex} to 40_{hex}
Access	RWW
Entry category	Optional
PDO mapping	Default
Value range	UNSIGNED8
Default value	0_{hex}
...	...
Subindex	**9_{hex} .. FE_{hex}**
Description	Write Output 41_{hex} to 48_{hex} ... Write Output $7E9_{hex}$ to $7F0_{hex}$
Access	RWW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	0_{hex}

CANopen presents the data to the user on multi-byte accesses as least significant byte to most significant byte. The user can access the same data using 8 bit and multi-byte accesses (such as 6200_{hex} and 6300_{hex}). The 8-bit access on the odd subindexes will indicate the lower 8 bits of the 16-bit access and even subindexes will indicate the upper 8 bits of the 16 bit access.

B 11.2 Object 6206_{hex}: Error Mode Output 8-Bit

This object indicates whether an output is set to a predefined error value (see object 6207_{hex}) in case of an internal device failure.

1 = Output value shall take the predefined condition specified in object 6207_{hex}.

0 = Output value shall be kept if an error occurs.

a. Object description

Index	6206_{hex}
Name	Error Mode Output 8-Bit
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Error Mode Output 1_{hex} to 8_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	Optional
Value range	UNSIGNED8
Default value	FF_{hex}

Subindex	**2_{hex}**
Description	Error Mode Output 9_{hex} to 10_{hex}
Access	RW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	FF_{hex}
...	...
Subindex	**FE_{hex}**
Description	Error Mode Output $7E9_{hex}$ to $7F0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	FF_{hex}

B 11.3 Object 6207_{hex}: Error Value Output 8-Bit

On condition that the corresponding error mode is active, device failures shall set the outputs to the value configured by this object.

0 = Output is set to "0" in case of fault, if Object 6206_{hex} is enabled.

1 = Output is set to "1" in case of fault, if Object 6206_{hex} is enabled.

a. Object description

Index	6207_{hex}
Name	Error Value Output 8-Bit
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	**0**$_{hex}$
Description	Number of Output 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1**$_{hex}$
Description	Error Value Output 1_{hex} to 8_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	Optional
Value range	UNSIGNED8
Default value	0_{hex}

Subindex	**2**$_{hex}$
Description	Error Value Output 9_{hex} to 10_{hex}
Access	RW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	0_{hex}
...	...
Subindex	**FE**$_{hex}$
Description	Error Value Output $7E9_{hex}$ to $7F0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED8
Default value	0_{hex}

B 11.4 Object 6300_{hex}: Write Output 16-Bit

This object shall set a group of 16 output lines as 2 byte of information. A maximum of 254 x 16-bit output blocks is addressable.

a. Object description

Index	6300_{hex}
Name	Write Output 16-Bit
Object	ARRAY
Data type	UNSIGNED16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 16-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Write Output 1_{hex} to 10_{hex}
Access	RWW
Entry category	Mandatory
PDO mapping	Default
Value range	UNSIGNED16
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Write Output 11_{hex} to 20_{hex}
Access	RWW
Entry category	Optional
PDO mapping	Default
Value range	UNSIGNED16
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Write Output $FE0_{hex}$ to $FF0_{hex}$
Access	RWW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED16
Default value	0_{hex}

B 11.5 Object 6306_{hex}: Error Mode Output 16-Bit

This object indicates whether an output is set to a predefined error value (see object 6307_{hex}) in case of an internal device failure.

1 = Output value shall take the predefined condition specified in object 6307_{hex}.

0 = Output value shall be kept if an error occurs.

a. Object description

Index	6306_{hex}
Name	Error Mode Output 16-Bit
Object	ARRAY
Data type	UNSIGNED16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 16-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Error Mode Output 1_{hex} to 10_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$

Subindex	**2_{hex}**
Description	Error Mode Output 11_{hex} to 20_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$
...	...
Subindex	**FE_{hex}**
Description	Error Mode Output $FE0_{hex}$ to $FF0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	$FFFF_{hex}$

B 11.6 Object 6307_{hex}: Error Value Output 16-Bit

On condition that the corresponding error mode is active, device failures shall set the outputs to the value configured by this object.

0 = Output is set to "0" in case of fault, if Object 6306_{hex} is enabled.

1 = Output is set to "1" in case of fault, if Object 6306_{hex} is enabled.

a. Object description

Index	6307_{hex}
Name	Error Value Output 16-Bit
Object	ARRAY
Data type	UNSIGNED16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 16-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Error Value Output 1_{hex} to 10_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Error Value Output 11_{hex} to 20_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Error Value Output $FE0_{hex}$ to $FF0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	0_{hex}

B 11.7 Object 6320_{hex}: Write Output 32-Bit

This object shall set a group of 32 output lines as 4 byte of information. A maximum of 254 x 32-bit output blocks is addressable.

a. Object description

Index	6320_{hex}
Name	Write Output 32-Bit
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Output 32-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Write Output 1_{hex} to 20_{hex}
Access	RWW
Entry category	Mandatory
PDO mapping	Default
Value range	UNSIGNED32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Write Output 21_{hex} to 40_{hex}
Access	RWW
Entry category	Optional
PDO mapping	Default
Value range	UNSIGNED32
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Write Output $1FC0_{hex}$ to $1FE0_{hex}$
Access	RWW
Entry category	Optional
PDO mapping	Optional
Value range	UNSIGNED32
Default value	0_{hex}

B 12 Analog input modules

B 12.1 Object 6400_{hex}: Read Analog Input 8-Bit

This object shall read the value of the input channel "n". The value is 8-bit wide or less. The value shall always be left adjusted. The remaining bits on the right side of the LSB shall be set to zero.

a. Object description

Index	6400_{hex}
Name	Read Analog Input 8-Bit
Object	ARRAY
Data type	INTEGER8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Analog Input 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Input 1_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Optional
Value range	INTEGER8
Default value	No

Subindex	**2_{hex} ... $1C_{hex}$**
Description	Analog Input 2_{hex} ... Analog Input $1C_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER8
Default value	No
...	...
Subindex	**$1D_{hex}$ ... FE_{hex}**
Description	Analog Input $1D_{hex}$ ... Analog Input FE_{hex}
Access	RO
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER8
Default value	No

CANopen presents the data to the user on multi-byte accesses as least significant byte to most significant byte. The user can access the same data using 8 bit and multi-byte accesses (such as 6400_{hex} and 6401_{hex}). The 8-bit access on the odd subindexes will indicate the lower 8 bits of the 16-bit access and even subindexes will indicate the upper 8 bits of the 16 bit access.

B 12.2 Object 6401_{hex}: Read Analog Input 16-Bit

This object shall read the value of the input channel "n". The value is 16-bit wide or less. The value shall always be left adjusted. The remaining bits on the right side of the LSB shall be set to zero.

a. Object description

Index	6401_{hex}
Name	Read Analog Input 16-Bit
Object	ARRAY
Data type	INTEGER16
Category	Conditional: Device with analog inputs

b. Entry description

Subindex	**0_{hex}**
Description	Number of Analog Input 16-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Input 1_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Default
Value range	INTEGER16
Default value	No

Subindex	**2_{hex} ... $1C_{hex}$**
Description	Analog Input 2_{hex} ... Analog Input $1C_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Default
Value range	INTEGER
Default value	No
...	...
Subindex	**$1D_{hex}$ ... FE_{hex}**
Description	Analog Input $1D_{hex}$ .. Analog Input FE_{hex}
Access	RO
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER
Default value	No

B 13 Analog output modules

B 13.1 Object 6410_{hex}: Write Analog Output 8-Bit

This object shall write an INTEGER8 value to the output channel "n". The value shall always be left adjusted.

a. Object description

Index	6410_{hex}
Name	Write Analog Output 8-Bit
Object	ARRAY
Data type	INTEGER8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of Analog Output 8-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Output 1_{hex}
Access	RWW
Entry category	Mandatory
PDO mapping	Default
Value range	INTEGER8
Default value	0_{hex}

Subindex	**2_{hex} ... $1C_{hex}$**
Description	Analog Output 2_{hex} ... Analog Output $1C_{hex}$
Access	RWW
Entry category	Default
PDO mapping	Optional
Value range	INTEGER8
Default value	0_{hex}
...	...
Subindex	**$1D_{hex}$ ... FE_{hex}**
Description	Analog Output $1D_{hex}$ ... Analog Output FE_{hex}
Access	RWW
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER8
Default value	0_{hex}

CANopen presents the data to the user on multi-byte accesses as least significant byte to most significant byte. The user can access the same data using 8 bit and multi-byte accesses (such as 6410_{hex} and 6411_{hex}). The 8-bit access on the odd subindexes will indicate the lower 8 bits of the 16-bit access. And even subindexes will indicate the upper 8 bits of the 16 bit access.

B 13.2 Object 6411_{hex}: Write Analog Output 16-Bit

This object shall write an INTEGER16 value to the output channel "n". The value shall always be left adjusted.

a. Object description

Index	6411_{hex}
Name	Write Analog Output 16-Bit
Object	ARRAY
Data type	INTEGER16
Category	Conditional: Device with analog outputs

b. Entry description

Subindex	**0_{hex}**
Description	Number of Analog Output 16-Bit
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Output 1_{hex}
Access	RWW
Entry category	Mandatory
PDO mapping	Default
Value range	INTEGER16
Default value	0_{hex}

Subindex	**2_{hex} ... $1C_{hex}$**
Description	Analog Output 2_{hex} ... Analog Output $1C_{hex}$
Access	RWW
Entry category	Default
PDO mapping	Optional
Value range	INTEGER16
Default value	0_{hex}
...	...

Subindex	**$1D_{hex}$ ... FE_{hex}**
Description	Analog Output $1D_{hex}$ .. Analog Output FE_{hex}
Access	RWW
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER16
Default value	0_{hex}

Analog input setups

The user should use care when enabling the Analog Input Interrupts. The user should first configure all of the Analog Interrupt Deltas, Upper, and Lower Limits. Failure to configure the interrupts before enabling the Analog Input Interrupt Global Enable will cause PDO messages to be generated on every bit change of every analog input and can lead to CAN "bus saturation". This means that the bus will be so full of Analog Input PDO messages that it may be impossible to get messages with a CAN ID that has a lower priority onto the bus. Another method of controlling the number of interrupt generated messages is to set the inhibit time in the communication parameter of the associated PDO.

B 14 Analog input setups

B 14.1 Object 6421_{hex}: Analog Input Interrupt Trigger Selection

This object determines which events shall cause an interrupt for a specific channel. Bits set in the list below shall refer to ways in which interrupts may be triggered.

Bit No.	Interrupt trigger
0	Upper limit exceeded
1	Input below lower limit
2	Input changed by more than delta
3	Input reduced by more than negative delta (not supported)
4	Input increased by more than positive delta (not supported)
5	Reserved for future use
6	Reserved for future use
7	Reserved for future use

a. Object description

Index	6421_{hex}
Name	Analog Input Interrupt Trigger Selection
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number of analog inputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	7

Subindex	$\mathbf{1}_{hex}$
Description	Analog Input 1_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	See above
Default value	7

Subindex	**2_{hex}**
Description	Analog Input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	See above
Default value	7
...	...
Subindex	**FE_{hex}**
Description	Analog Input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	See above
Default value	7

B 14.2 Object 6423_{hex}: Analog Input Global Interrupt Enable

This object shall enable and disable globally the interrupt behavior without changing the interrupt mask.

By default, no analog input activates an interrupt.

TRUE = Global interrupt enabled

FALSE = Global interrupt disabled

a. Object description

Index	6423_{hex}
Name	Analog Input Global Interrupt Enable
Object	Var
Data type	BOOLEAN
Category	Conditional: Device with analog inputs

b. Object description

Subindex	**0_{hex}**
Access	RW
PDO mapping	Optional
Value range	BOOLEAN
Default value	FALSE

B 14.3 Object 6424_{hex}: Analog Input Interrupt Upper Limit Integer

When enabled (see Object 6423_{hex}), an interrupt is triggered when the analog input rises above the given value. The value shall always be left adjusted. As long as the trigger condition is met, every change of the analog input data generated a new interrupt, as long as there is no additional trigger condition, e.g. an input interrupt delta (6426_{hex}).

a. Object description

Index	6424_{hex}
Name	Analog Input Interrupt Upper Limit Integer
Object	ARRAY
Data type	INTEGER32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0_{hex} to FE_{hex}
Default	No

Subindex	**1_{hex}**
Description	Analog Input 1_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Analog Input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	Optional
Value range	INTEGER32
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Analog Input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

B 14.4 Object 6425_{hex}: Analog Input Interrupt Lower Limit Integer

a. Object description

Index	6425_{hex}
Name	Analog Input Interrupt Lower Limit Integer
Object	ARRAY
Data type	INTEGER32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Input 1_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Analog Input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}
....	...
Subindex	**FE_{hex}**
Description	Analog Input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

B 14.5 Object 6426_{hex}: Analog Input Interrupt Delta Unsigned

a. Object description

Index	6426_{hex}
Name	Analog Input Interrupt Delta Unsigned
Object	ARRAY
Data type	UNSIGNED32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Input 1_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Analog Input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED32
Default value	0_{hex}
…	...
Subindex	**FE_{hex}**
Description	Analog Input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED32
Default value	0_{hex}

B 15 Analog output setups

B 15.1 Object 6443_{hex}: Analog Output Error Mode

a. Object description

Index	6443_{hex}
Name	Analog Output Error Mode
Object	ARRAY
Data type	UNSIGNED8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog outputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Error Mode Analog Output 1_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	1_{hex}

Subindex	**2_{hex}**
Description	Error Mode Analog Output 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	1_{hex}
...	...
Subindex	**FE_{hex}**
Description	Error Mode Analog Output FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	1_{hex}

B 15.2 Object 6444_{hex}: Analog Output Error Value Integer

a. Object description

Index	6444_{hex}
Name	Analog Output Error Value Integer
Object	ARRAY
Data type	INTEGER32

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog outputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	No

Subindex	**1_{hex}**
Description	Analog Output 1_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Analog Output 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Analog Output FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	INTEGER32
Default value	0_{hex}

B 16 Manufacturer specific

When set this corresponding digital input is latched when it goes to the defined state (see Object 2001_{hex}).

B 16.1 Object 2000_{hex}: DIP Latch Enable (8-Bit)

a. Object description

Index	2000_{hex}
Name	DIP Latch Enable (8-Bit)
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Digital Inputs (8-Bit)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-64
Default value	0

Subindex	**1_{hex}**
Description	Latch Enable Inputs 1-8
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of digital inputs (8-bit)**
Description	Latch Enable Inputs N (8-Bit)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

After changing object 2000_{hex} or 2001_{hex} be sure to reset the latches using the Clear DIP Latches command (bit 1 of the Inline Interface Control Byte, see object 3111_{hex}) or power cycle in order to clear any previously latched values due to the latch configuration change. Failure to send this command or power cycle the bus coupler may result in erroneously latched values.

B 16.2 Object 2001_{hex}: DIP Latch State (8-Bit)

When set the corresponding digital input is latched when it goes high. When set to 0 the corresponding input is latched when it goes low. The latching must be enabled (see Object 2000_{hex}).

a. Object description

Index	2000_{hex}
Name	DIP Latch State (8-Bit)
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Digital Inputs (8-Bit)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-64
Default value	0

Subindex	**1_{hex}**
Description	Latch State Inputs 1-8
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	255
...	...
Subindex	**(N) = Number of digital inputs (8-bit)**
Description	Latch State Inputs N (8-Bit)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	255

After changing object 2000_{hex} or 2001_{hex} be sure to reset the latches using the Clear DIP Latches command (bit 1 of the Inline Interface Control Byte, see object 3111_{hex}) or power cycle in order to clear any previously latched values due to the latch configuration change. Failure to send this command or power cycle the bus coupler may result in erroneously latched values.

B 16.3 Object 2400_{hex}: AIP Range

Determines the input range for a corresponding analog input, see AIP module-specific data sheet.

a. Object description

Index	2400_{hex}
Name	AIP Range
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-254
Default value	0

Subindex	**1_{hex}**
Description	Analog Input 1 Range
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of analog inputs**
Description	N = Number of analog inputs
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 16.4 Object 2410_{hex}: AOP Response Data

This value is the response that is returned from analog output modules that can be programmed via process data (i.e. AO 2 modules).

a. Object description

Index	2410_{hex}
Name	AOP Response Data
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	N = Number of Analog Outputs with Response Data
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-254
Default value	0

Subindex	**1_{hex}**
Description	Analog Output 1 Response Data
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of analog inputs with response data**
Description	Analog Output (N) Response Data
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

B 16.5 Object 3000_{hex}: Reconfigure I/O

Setting this value to 1 causes all connected modules to be added to the I/O scan and automatically re-maps the PDOs as defined in DS-401. This object can only be written to when in the preoperational state. The configuration is only temporary until the user does a store command.

a. Object description

Index	3000_{hex}
Name	Reconfigure I/O
Object	VAR
Data type	BOOL
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-1
Default value	0

B 17 Fault control objects

B 17.1 Object 3002_{hex}: Inline Fault Mode

This object determines how the bus coupler controls outputs when a fault occurs that prevents the bus coupler from communicating with all I/O modules. Inline fault modes are initiated by configuration error(s), module change error(s), and connection-break error(s). Once the bus coupler detects and error, it will respond in one of the four error modes described below.

Fault modes

0 = Stop running data cycles on the local bus until the problem is corrected and a command is sent across the bus to restart the bus.

1 = Stop running data cycles on the local bus and continuously attempt to correct the problem, for example a loop connection if broken and then repaired, see note 1.

2 = Run data cycles with fault values to the remaining modules that the bus coupler can communicate with. The data used is always the fault values defined by the user. The module continues to run in this mode until a command is issued to restart the local bus. The Fault Cycles bit is set in the Inline status byte (see below) if either of these faults occur, see note 2.

3 = Run data cycles to the remaining modules that the bus coupler can communicate with. The output data that is used is defined by the fault modules 3002_{hex}-$300A_{hex}$. Output data can be either fault values or process data. The module continues to run in this mode until a command is issued to restart the local bus, see note 2.

a. Object description

Index	3002_{hex}
Name	Inline Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-3
Default value	1

In the event the bus coupler is unable to auto-recover the local bus, the user may still be able to recover the bus remotely. This is done by changing the state from stop to preoperational, to operational (issuing a Restart Bus command (Object 3111_{hex}, bit 2), or by issuing a Reset Node command. If the local bus is still not be recovered, the user should power down the module and check the connections between all modules.

B 17.2 Object 3003_{hex}: Inline CRC Fault Mode

This object specifies which state the module is set to when an Inline CRC error is detected.

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3003_{hex}
Name	Inline CRC Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	1

To clear the Inline CRC Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.3 Object 3004_{hex}: Inline PF (Peripheral Fault) Mode

This object specifies which state the module is set to when an Inline PF error is detected. An example of a PF is a shorted output.

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3004_{hex}
Name	Inline PF Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	1

To clear the Inline PF mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.4 Object 3005_{hex}: Inline Power Fault Mode

This object specifies which state the I/O module is set to when an Inline Power Fault error is detected. The fault becomes active at a voltage of less than approximately 18 V. The faulted power supply can be U_{BK}, U_S, U_M or processor power.

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3005_{hex}
Name	Inline Power Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	1

To clear the Inline Power Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.5 Object 3006_{hex}: Inline Module Change Fault Mode

This object specifies whether the module can be put into the run mode when receiving a Start Remote Node command when there has been a module change.

0 = Keep device in preoperational state

1 = Allow operational mode

a. Object description

Index	3006_{hex}
Name	Inline Module Change Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-1
Default value	1

1. If the user tries to put the bus coupler in the run mode when a module change is detected and the Inline Module Change Fault mode is set to 0, the Start Error bit is set tin the Station Status Object (see Object 3101_{hex}) and an emergency message is generated to inform the user that the module is not going into the run mode.
2. To clear the Inline Module Change Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.6 Object 3007_{hex}: Inactive Local Bus Fault Mode

This object specifies which state the module is set to when the local bus is inactive and not running data cycles. Object 3007_{hex} is initiated when a connection or module change error occurs or when the device is trying to initialize the local bus.

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3007_{hex}
Name	Inline Local Bus Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	0

To clear the Incative Local Bus Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.7 Object 3008_{hex}: Inline Connection Fault Mode

This object specifies which state the module is set to when an Inline Connection Fault error is detected

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3008_{hex}
Name	Inline Connection Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	0

To clear the Inline Connection Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.8 Object 3009_{hex}: Inline Faulted Cycles Mode

This object specifies which state the module is set to when the bus couple is running data cycles using the fault data information instead of process data.

0 = Preoperational (only if current state is operational)

1 = No state change

2 = Stopped

a. Object description

Index	3009_{hex}
Name	Inline Faulted Cycles Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0-2
Default value	1

To clear the Inline Faulted Cycles mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.9 Object $300A_{hex}$: Processor Power Loss Fault Mode

This object specifies whether or not to turn OFF all I/O module outputs during a processor blowout condition. A processor blowout condition occurs when processor power (U_{BK}) falls below 11 V, but does not drop low enough to shut down all the I/O modules. U_{BK} power is necessary to power all communications on the local bus. To restart the bus, a Restart Bus command must be sent by the user. This is bit 1 of the object 3111_{hex}.

0 = Outputs remain current state

1 = Reset the local bus (default): This allows all modules to go to their default output states. (See module data sheets for these default values.)

a. Object description

Index	$300A_{hex}$
Name	Processor Power Loss Fault Mode
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	1

To clear the Processor Power Loss Fault mode the user can issue a Start Remote Node command. This attempt will correct the problem. If the reason for the fault is no longer active, the Fault Cycles bit will be cleared in the Inline Status byte. Furthermore, process data will again be used instead of fault values. If any module causes the module to go to the stop state and that fault needs special handling to be cleared (such as a module that has a latched PF), it may not be possible to sent the module back to the operational state. In this case, the user can issue a Reset Node command to restart the local bus. This in turn should clear any latched faults, as long as the reason for the fault has been eliminated. It is also possible to change to the preoperational state and then acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

B 17.10 Object $300F_{hex}$: Erase Configuration

When set to a 1, it causes the configuration to be erased. This allows the device to be auto-configured upon first power up. Additionally the user can use this object to reconfigure the whole device. A power cycle or a "node reset" must be performed immediately after a write to this object. Failure to do so may require the device to be reconfigured manually by setting the DIP switches to all zeros and doing a power cycle.

a. Object description

Index	$300F_{hex}$
Name	Erase Configuration
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	BOOLEAN
Default value	0

B 17.11 Object 3101_{hex}: Station Status

a. Object description

Index	3101_{hex}
Name	Station Status
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	0

c. Module status

Inline status (low byte)		
Bit	**Name**	**Description**
0	CRC	An Inline CRC error is detected.
1	Peripheral Fault (PF)	An Inline Peripheral Fault is detected.
2	Power Fault	One or more power inputs is below the required voltage or not connected. Read power status to determine which supply/supplies are faulty.
3	Module Change	The configuration of I/O modules does not match the stored configuration.
4	Inactive Local Bus	The bus coupler is not running data cycles on the local bus.
5	Connection	There is a break between two or more I/O modules.
6	Fault Cycles	There was a fault and the bus coupler is now using the defined fault values for the output modules instead of process data.
7	Reserved	

Configuration status (high byte)		
Bit	**Name**	**Description**
0	Node ID Change	See note 1
1	Reserved	
2	Start Error	See note 2
3	Reserved	
4	Reserved	
5	Reserved	
6	Reserved	
7	Reserved	

1. Node ID Change: This bit is set to "1" when the user changes the address of the device. This bit can be cleared by writing a "save" to Object 1010_{hex}, or by setting the Node ID switches to zero, power cycling the device and then to the desired ID and finally doing another power cycle.
2. Start Error: The module configuration has changed and the user is requesting to go to the run mode by issuing a Start Remote Node service. The configuration of the bus coupler is set not to allow the run mode with a module change (see Object 3006_{hex}). Therefore, this bit is set and an emergency message is generated.

B 17.12 Object 3102_{hex}: Inline Faulted Module

Contains the number of the first faulty module

0 = IL CAN BK-TC-PAC

1 = First attached I/O module

2 = Second attached I/O module

3 = ...N attached I/O module

a. Object description

Index	3102_{hex}
Name	Inline Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.13 Object 3103_{hex}: Inline Retry Max

Sets the number of bad responses the IL CAN BK-TC-PAC will accept before indicating a CRC error. Default = 32

a. Object description

Index	3103_{hex}
Name	Inline Retry Max
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	UNSIGNED8
Default value	32

B 17.14 Object 3104_{hex}: Inline Number of Modules

Displays the number of Inline modules that the IL CAN BK-TC-PAC detected.

a. Object description

Index	3104_{hex}
Name	Inline Number of Modules
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	0-63
Default value	No

B 17.15 Object 3105_{hex}: Inline Count of Bits

Displays the data size of the Inline modules in bits. This value reflects the total number of bits, process data and PCP.

a. Object description

Index	3105_{hex}
Name	Inline Count of Bits
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.16 Object 3106_{hex}: Count of Bytes

Displays the data size of the Inline modules in bytes. This value reflects the total number of bytes, process data and PCP.

a. Object description

Index	3106_{hex}
Name	Inline Count of Bytes
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.17 Object 3109_{hex}: Inline Loop Diag Count

Displays the loop diagnostic count during a connection failure. This is most useful during the first power up where the configuration is unknown.

a. Object description

Index	3109_{hex}
Name	Inline Loop Diag Count
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.18 Object $310A_{hex}$: Inline First Faulted Module

When a connection error occurs this value will indicate the first module before the break where backplane communications was interrupted.

a. Object description

Index	$310A_{hex}$
Name	Inline First Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.19 Object $310B_{hex}$: Inline Last Faulted Module

When a connection error occurs this value will indicate the last module after the break where backplane communications was interrupted.

a. Object description

Index	$310B_{hex}$
Name	Inline Last Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.20 Object $310C_{hex}$: Inline Latched Fault

This value contains any errors that have been latched by the module. See Object 3101_{hex} for a description.

a. Object description

Index	$310C_{hex}$
Name	Inline Latched Fault
Objects	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

If any of the fault modes causes the outputs to use fault values, the original cause of the fault is cleared, the value in this object will indicate the Fault Cycles bit only.

B 17.21 Object $310D_{hex}$: Inline Latched Faulted Module

This object contains the number of the last faulted module.

a. Object description

Index	$310D_{hex}$
Name	Inline Latched Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

If any of the fault modes causes the outputs to use fault values, the original cause of the fault is cleared, the value in this object will be 0.

B 17.22 Object $310E_{hex}$: Inline Latched First Faulted Module

This value contains the number of the first module before a connection break latched during the last connection failure.

a. Object description

Index	$310E_{hex}$
Name	Inline Latched First Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.23 Object $310F_{hex}$: Inline Latched Last Faulted Module

This value contains the number of the last module after a connection break latched during the last connection failure.

a. Object description

Index	$310F_{hex}$
Name	Inline Latched Last Faulted Module
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.24 Object 3110_{hex}: Inline Power Status

Indicates the status of power supplies connected to the IL CAN BK-TC-PAC.

0 = Power supply is OK

1 = Power supply out of range

Bit	Description
0	Processor
1	U_{BK}
2	U_S
3	U_M
4	Reserved
5	Reserved
6	Reserved
7	Reserved

a. Object description

Index	$310F_{hex}$
Name	Inline Power Status
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.25 Object 3111_{hex}: Inline Interface Control Byte

a. Object description

Index	3111_{hex}
Name	Inline Control Byte
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Access	RWW
PDO mapping	Yes
Value range	UNSIGNED8
Default value	0

Bit	Name	Description
0	Acknowledge Peripheral Faults	Used to acknowledge peripheral faults on modules that latch the peripheral fault such as the IB IL 24 SEG/F-PAC module. Setting the value to a 1 will acknowledge the fault and clear it if the reason for the fault has gone away. After the fault is removed this value should be set back to a 0 to enable normal operation.
1	Clear All Latched Digital Inputs	When set to a 1, it will clear all DIP latches if enabled. This value should be set back to a 0 for normal operation.
2	Restart Local Bus	When set to a 1, it will restart the Inline local bus. This is the method used to recover from faults that cause the Inline cycles to be stopped.
3	Reserved	
4	Reserved	
5	Reserved	
6	Reserved	
7	Reserved	

B 17.26 Object 3200_{hex}: Inline Module Stored ID

Displays the Inline ID for the original device configuration.

a. Object description

Index	3200_{hex}
Name	Inline Module Stored ID
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Inline Module 1 Stored ID
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	Inline Module (N) Stored ID
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.27 Object 3201_{hex}: Inline Module Current ID

Displays the current Inline module ID.

a. Object description

Index	3201_{hex}
Name	Inline Module Current ID
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Inline Module 1 Current ID
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	Inline Module (N) Current ID
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.28 Object 3202_{hex}: IN Data COP Index

Reflects the CANopen index number that the module's input data is mapped to (i.e. DIP module = 6000_{hex}, AIP module = 6401_{hex}, etc.).

a. Object description

Index	3202_{hex}
Name	IN Data COP Index
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	IN Data COP Index Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	IN Data COP Index Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.29 Object 3203_{hex}: IN Data COP First Subindex

Reflects the first subindex of the CANopen index that this module's input data is mapped to.

a. Object description

Index	3203_{hex}
Name	IN Data COP First Subindex
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	$\mathbf{1}_{hex}$
Description	IN Data COP First Subindex Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	IN Data COP First Subindex Index Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.30 Object 3204_{hex}: IN Data COP Last Subindex

Reflects the last subindex of the CANopen index that this module's input data is mapped to.

a. Object description

Index	3204_{hex}
Name	IN Data COP Last Subindex
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	IN Data COP Last Subindex Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	IN Data COP Last Subindex Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.31 Object 3205_{hex}: OUT Data COP Index

Reflects the CANopen index number that the module's output data is mapped to (i.e. DOP module = 6200_{hex}, AOP module = 6411_{hex}, etc.).

a. Object description

Index	3205_{hex}
Name	OUT Data COP Index
Object	VAR
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	OUT Data COP Index Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	OUT Data COP Index Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

B 17.32 Object 3206_{hex}: OUT Data COP First Subindex

Reflects the first subindex of the CANopen index that this module's output data is mapped to.

a. Object description

Index	3206_{hex}
Name	OUT Data COP First Subindex
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	OUT Data COP First Subindex Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	OUT Data COP First Subindex Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.33 Object 3207_{hex}: OUT Data COP Last Subindex

Reflects the last subindex of the CANopen index that this module's output data is mapped to.

a. Object description

Index	3207_{hex}
Name	OUT Data COP Last Subindex
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	OUT Data COP Last Subindex Module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached I/O modules**
Description	OUT Data COP Last Subindex Module n
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.34 Object 3300_{hex}: Special Function DataSize

DataSize pertains to the number of bytes of process data.

a. Object description

Index	3300_{hex}
Name	Special Function DataSize
Object	VAR
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataSize
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataSize
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

B 17.35 Object 3301_{hex}: Special Function Status

Reflects the status of the special function module.

0 = OK

1 = Error

a. Object description

Index	3301_{hex}
Name	Special Function Status
Object	VAR
Data type	BOOLEAN
Category	Mandatory

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 Status
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-1
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) Status
Access	RO
Entry category	Optional
PDO mapping	No
Value range	0-1
Default value	No

B 17.36 Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules

Special function modules can be accessed through object indexes 3302_{hex} - $330F_{hex}$.

Object indexes 3302_{hex} - $330D_{hex}$ provide access to modules with a maximum of 8 bytes of special function data.
Object indexes $330E_{hex}$ and $330F_{hex}$ provide access to special function modules with greater than 8 bytes of data.

Data length determines which object index should be used to access special function module data.

For example:

The IB IL 400 MLR 1-8A module uses 1 byte of special function input/output data. It should be accessed by object indexes 3302_{hex} and 3303_{hex}.

The IB IL INC-PAC module has 4 bytes of special function input/output data. It should be accessed by object indexes 3308_{hex} and 3309_{hex}.

Partial access of a particular special function module is possible by using object indexes with a lesser data length. (Lowest byte of object indexes 3308_{hex} and 3309_{hex} is also accessible by object indexes 3302_{hex} and 3303_{hex}.)

B 17.37 Object 3302_{hex}: Special Function DataIn 1 Byte

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	3302_{hex}
Name	Special Function DataIn 1 Byte
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 1 Byte
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 1 Byte
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.38 Object 3303_{hex}: Special Function DataOut 1 Byte

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	3303_{hex}
Name	Special Function DataOut 1 Byte
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 1 Byte
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 1 Byte
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED8

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.39 Object 3304_{hex}: Special Function DataIn 2 Bytes

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	3304_{hex}
Name	Special Function DataIn 2 Bytes
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 2 Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 2 Bytes
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.40 Object 3305_{hex}: Special Function DataOut 2 Bytes

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	3305_{hex}
Name	Special Function DataOut 2 Bytes
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 2 Bytes
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 2 Bytes
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED16
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.41 Object 3306_{hex}: Special Function DataIn 3 Bytes

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	3306_{hex}
Name	Special Function DataIn 3 Bytes
Object	ARRAY
Data type	UNSIGNED24
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 3 Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED24
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 3 Bytes
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED24
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.42 Object 3307_{hex}: Special Function DataOut 3 Bytes

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	3307_{hex}
Name	Special Function DataOut 3 Bytes
Object	ARRAY
Data type	UNSIGNED24
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 3 Bytes
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED24
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 3 Bytes
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED24
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.43 Object 3308_{hex}: Special Function DataIn 4 Bytes

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	3308_{hex}
Name	Special Function DataIn 4 Bytes
Object	ARRAY
Data type	UNSIGNED32
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 4 Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No
..	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 4 Bytes
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.44 Object 3309_{hex}: Special Function DataOut 4 Bytes

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	3309_{hex}
Name	Special Function DataOut 4 Bytes
Object	ARRAY
Data type	UNSIGNED32
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 4 Bytes
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 4 Bytes
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED32
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.45 Object $330A_{hex}$: Special Function DataIn 6 Bytes

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	$330A_{hex}$
Name	Special Function DataIn 6 Bytes
Object	ARRAY
Data type	UNSIGNED48
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 6 Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED48
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 6 Bytes
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED48
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.46 Object $330B_{hex}$: Special Function DataOut 6 Bytes

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	$330B_{hex}$
Name	Special Function DataOut 6 Bytes
Object	ARRAY
Data type	UNSIGNED48
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 6 Bytes
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED48
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 6 Bytes
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED48
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.47 Object $330C_{hex}$: Special Function DataIn 8 Bytes

IN data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	$330C_{hex}$
Name	Special Function DataIn 8 Bytes
Object	ARRAY
Data type	UNSIGNED64
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataIn 8 Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED64
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataIn 8 Bytes
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED64
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.48 Object $330D_{hex}$: Special Function DataOut 8 Bytes

OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	$330D_{hex}$
Name	Special Function DataOut 8 Bytes
Object	ARRAY
Data type	UNSIGNED64
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	63

Subindex	**1_{hex}**
Description	Special Function Module 1 DataOut 8 Bytes
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED64
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special Function Module (N) DataOut 8 Bytes
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED64
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.49 Object $330E_{hex}$: Special Function DataIn Large Data

This object provides access to modules with a data sizes greater than 8 bytes. The user can map the data across more than one PDO. IN data: Data returned from the Inline module to the IL CAN BK-TC-PAC. The data size is determined by the module.

a. Object description

Index	$330E_{hex}$
Name	Special Function DataIn Large Data
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Special Function Large Data Bytes
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-254
Default value	63

Subindex	**1_{hex}**
Description	Number of Special Function Module Large Data Bytes
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of special function module large data bytes**
Description	Special Function Module Large Data Byte
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 17.50 Object $330F_{hex}$: Special Function DataOut Large Data

This object provides access to modules with a data sizes greater than 8 bytes. The user can map the data across more than one PDO. It is up to the user to ensure data consistency. OUT data sent from the IL CAN BK-TC-PAC to the Inline module. The data size is determined by the module.

a. Object description

Index	$330F_{hex}$
Name	Special Function DataOut (Large Data > 8 Bytes)
Object	ARRAY
Data type	UNSIGNED64
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Inline Special Function Large Data Bytes
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-254
Default value	63

Subindex	**1_{hex}**
Description	Number of Special Function Module Large Data Byte 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of special function module large data bytes**
Description	Special Function Module DataOut Large Data
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	UNSIGNED8
Default value	No

Please note Section "Objects 3302_{hex} to $330F_{hex}$: Notes on accessing special function modules" on page B-112.

B 18 CANopen PCP implementation

Manufacturer-specific data types

B 18.1 PCP_FRAG_REQUEST record data types

Index	Subindex	Field in PCP_FRAG_REQUEST record	Type
0040_{hex}	0_{hex}	Number of supported entries in the record	UNSIGNED8
	1_{hex}	Service	UNSIGNED8
	2_{hex}	Request Data Byte 1	UNSIGNED8
	3_{hex}	Request Data Byte 2	UNSIGNED8
	4_{hex}	Request Data Byte 3	UNSIGNED8
	5_{hex}	Request Data Byte 4	UNSIGNED8
	6_{hex}	Request Data Byte 5	UNSIGNED8
	7_{hex}	Request Data Byte 6	UNSIGNED8
	8_{hex}	Request Data Byte 7	UNSIGNED8

B 18.2 PCP_FRAG_RESPONSE record specification

Index	Subindex	Field in PCP_FRAG_RESPONSE record	Type
0041_{hex}	0_{hex}	Number of supported entries in the record	UNSIGNED8
	1_{hex}	Service	UNSIGNED8
	2_{hex}	Response Data Byte 1	UNSIGNED8
	3_{hex}	Response Data Byte 2	UNSIGNED8
	4_{hex}	Response Data Byte 3	UNSIGNED8
	5_{hex}	Response Data Byte 4	UNSIGNED8
	6_{hex}	Response Data Byte 5	UNSIGNED8
	7_{hex}	Response Data Byte 6	UNSIGNED8
	8_{hex}	Response Data Byte 7	UNSIGNED8

B 19 PCP interface objects

Index (hex)	Object	Name	Data type	Access
3400	ARRAY	PCP PDU Size	UNSIGNED8	RO
3401	ARRAY	PCP PCP Size	UNSIGNED8	RO
3402	ARRAY	PCP Status	UNSIGNED8	RO
3403	ARRAY	PCP Request	DOMAIN	RW
3404	ARRAY	PCP Response	DOMAIN	RO
3405	ARRAY	PCP Module	UNSIGNED8	RW
3406	ARRAY	PCP Write Index	UNSIGNED16	RW
3407	ARRAY	PCP Write Subindex	UNSIGNED8	RW
3408	ARRAY	PCP Write Data	DOMAIN	RW
3409	ARRAY	PCP Read Index	UNSIGNED16	RW
340A	ARRAY	PCP Read Subindex	UNSIGNED8	RW
340B	ARRAY	PCP Read Data	DOMAIN	RO
340C	ARRAY	PCP Request Fragment	PCP_FRAG_REQUEST	RW
340C	ARRAY	PCP Response Fragment	PCP_FRAG_RESPONSE	RO
340D	ARRAY	PCP Write Invoke ID	UNSIGNED8	RW
340E	ARRAY	PCP Read Invoke ID	UNSIGNED8	RW
340F	ARRAY	PCP Write Data Confirmation	DOMAIN	RO
3410	ARRAY	PCP Read Data Confirmation	DOMAIN	RO

B 19.1 Object 3400_{hex}: PCP PDU Size

a. Object description

Index	3400_{hex}
Name	PCP PDU Size
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Module 1 PDU Size
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Module (N) PDU Size
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

The PCP PDU Size object contains the value of the PCP size used for the PDU channel of the module. Typically 64 bytes. This value is the maximum number of bytes that the PCP channel can transmit per request/response.

B 19.2 Object 3401_{hex}: PCP Module Channel Size

a. Object description

Index	3401_{hex}
Name	PCP Module Channel Size
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Module 1 Channel Size
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Module (N) Channel Size
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

The PCP Channel Size object contains the value of the PCP channel size. This indicates the number of bytes that are transferred during each Inline scan.

B 19.3 Object 3402_{hex}: PCP Module Status

a. Object description

Index	3402_{hex}
Name	PCP Module Status
Object	ARRAY
Data type	UNSIGNED8
Category	Mandatory

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0
...	...
Subindex	**1_{hex}**
Description	PCP Module 1 Status
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Module (N) Status
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

c. Status — Reflects the status of the PCP module

Bit	Name	Description
0	Module COMM Error	When set, communication with the module is no longer possible.
1	PCP Error	A PCP service-specific error has occurred. See the Error Class and Error Code bytes.
2	PCP Channel Busy	A PCP transaction is already in progress, such as from an SDO or explicit request.
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).
4	PCP Parameter Error	PCP parameter error: The user has sent a service with invalid parameters.
5	PCP Reject Error	The PCP module has rejected the PCP request (see PCP reject response below).
6	PCP Abort Error	The PCP module has aborted the PCP request (see PCP abort response below).
7	Reserved	

B 19.4 Object 3403_{hex}: PCP Request

a. Object description

Index	3403_{hex}
Name	PCP Request
Object code	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Request Module (1)
Access	RWW
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Request Module (N)
Access	RWW
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

c. Format

Services - No Invoke ID	
Byte 1	Service
Byte 2	Index low
Byte 3	Index high
Byte 4	Subindex
Byte 5	Length
Byte 6	Data block, if necessary
...	...
Byte N	Data block, if necessary

Services - With Invoke ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Index low
Byte 4	Index high
Byte 5	Subindex
Byte 6	Length
Byte 7	Data block, if necessary
...	...
Byte N	Data block, if necessary

B 19.5 Object 3404_{hex}: PCP Response

a. Object description

Index	3404_{hex}
Name	PCP Response
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Response Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Response Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

c. Format

	Status byte definition (see Object 3402_{hex})	
Bit	**Name**	**Description**
0	Module COMM Error	When set, communication with the module is no longer possible.
1	PCP Error	A PCP service-specific error has occurred. See the Error Class and Error Code bytes.
2	PCP Channel Busy	A PCP transaction is already in progress, such as from an SDO or explicit request.
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).
4	PCP Parameter Error	The user has sent a service with invalid parameters.

5	PCP Reject Error	The PCP module has rejected the PCP request. See PCP reject response below.
6	PCP Abort Error	The PCP module has aborted the PCP request. See PCP abort response below.
7	Reserved	

d. Successful responses

Services - No Invoke ID	
Byte 1	Service
Byte 2	Status
Byte 3	Length
Byte 4	Data block, if necessary
...	...
Byte N	Data block, if necessary

Services - With Invoke ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Length
Byte 5	Data block, if necessary
...	...
Byte N	Data block, if necessary

e. PCP error response

PCP error response - No Invoke-ID	
Byte 1	Service
Byte 2	Status
Byte 3	Error class
Byte 4	Error code
Byte 5	Additional Error Code LSB, if necessary
Byte 6	Additional Error Code MSB, if necessary

PCP error response - With Invoke-ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Error class
Byte 5	Error code
Byte 6	Additional Error Code LSB, if necessary
Byte 7	Additional Error Code MSB, if necessary

f. PCP reject response

PCP reject response - No Invoke-ID	
Byte 1	Service
Byte 2	Status
Byte 3	PDU type
Byte 4	Reject code

PCP reject response - With Invoke ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	PDU type
Byte 5	Reject code

g. PCP abort response

PCP abort response - No Invoke ID	
Byte 1	Service
Byte 2	Status
Byte 3	Abort ID
Byte 4	Reason code
Bytes 5-8	Abort detail

PCP abort response - With Invoke ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Abort ID
Byte 5	Reason code
Byte 5	Abort detail

B 19.6 Object 3405_{hex}: PCP Module Number

PCP abort response - No Invoke ID	
Byte 1	Service
Byte 2	Status
Byte 3	Abort ID
Byte 4	Reason code
Bytes 5-8	Abort detail

PCP abort response - With Invoke ID	
Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Abort ID
Byte 5	Reason code
Bytes 6-9	Abort detail

a. Object description

Index	**3405_{hex}**
Name	PCP Module Number
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Module Number (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	0 - Number of attached PCP modules
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Module Number (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	0 - Number of attached PCP modules
Default value	0

The PCP Module object attribute allows the user to set the PCP module number that read and write objects (3408_{hex} and $340C_{hex}$) will access.

B 19.7 Object 3406_{hex}: PCP Write Index

a. Object description

Index	3406_{hex}
Name	PCP Write Index
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Write Index (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Write Index (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	0

The PCP Write Index object sets the index of the PCP object dictionary that will be written when the user writes to the PCP Write Data object (3408_{hex}).

B 19.8 Object 3407_{hex}: PCP Write SubIndex

a. Object description

Index	3407_{hex}
Name	PCP Write SubIndex
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Write SubIndex (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Write SubIndex (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

The PCP Write SubIndex object sets the subindex of the PCP object dictionary that will be written when the user writes to the PCP Write Data object (3408_{hex}).

B 19.9 Object 3408_{hex}: PCP Write Data

a. Object description

Index	3408_{hex}
Name	PCP Write Data
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Write Data (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Write Data (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

The PCP Write Data object allows the user to write data to the PCP object dictionary that is referenced by the PCP Module, PCP Write Index, and PCP Write SubIndex objects.

c. Format of write data

UNSIGN8	PCP write size
ARRAY	[PCP write size] of UNSIGNED8 data

If an SDO Abort occurs, the reason for the abort can be read at object 3410_{hex}. See PCP Write Data Confirmation object (3410_{hex}) for details.

B 19.10 Object 3409_{hex}: PCP Read Index

a. Object description

Index	3409_{hex}
Name	PCP Read Index
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Read Index (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Read Index (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	0

The PCP Read Index object sets the index of the PCP object dictionary that will be read when the user reads from the PCP Read Data object ($340B_{hex}$).

B 19.11 Object $340A_{hex}$: PCP Read SubIndex

a. Object description

Index	$340A_{hex}$
Name	PCP Read SubIndex
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Read SubIndex (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Read SubIndex (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

The PCP Read SubIndex object sets the subindex of the PCP object dictionary that will be read when the user reads from the PCP Read Data object ($340B_{hex}$).

B 19.12 Object $340B_{hex}$: PCP Read Data

a. Object description

Index	$340B_{hex}$
Name	PCP Read Data
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Read Data (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Read Data (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

The PCP Read Data object allows the user to read data from the PCP object dictionary that is referenced by the PCP Module, PCP Read Index, and PCP Read SubIndex objects.

c. Format of read data

UNSIGN8	Response Data Size
ARRAY	[Response Data Size] of UNSIGNED8 data

If an SDO Abort occurs, the reason for the abort can be read at object 3411_{hex}. See PCP Read Data Confirmation object for details.

B 19.13 Object $340C_{hex}$: PCP Request Fragment

a. Object description

Index	$340C_{hex}$
Name	PCP Request Fragment
Object	ARRAY
Data type	PCP_FRAG_REQUEST(40_{hex})
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	0

Subindex	**1_{hex}**
Description	PCP Request Fragment Module (1)
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	PCP_FRAG_REQUEST(40_{hex})
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Request Fragment Module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	PCP_FRAG_REQUEST(40_{hex})
Default value	No

See also Section "Fragmented services" on page C-7.

B 19.14 Object $340D_{hex}$: PCP Response Fragment

a. Object description

Index	$340D_{hex}$
Name	PCP Response Fragment
Object	ARRAY
Data type	PCP_FRAG_RESONSE(41_{hex})
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Response Fragment Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	PCP_FRAG_RESONSE(41_{hex})
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Response Fragment Module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	PCP_FRAG_RESONSE(41_{hex})
Default value	No

See also Section "Fragmented services" on page C-7.

B 19.15 Object $340E_{hex}$: PCP Write Invoke ID

a. Object description

Index	$340E_{hex}$
Name	PCP Write Invoke ID
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0

Subindex	**1_{hex}**
Description	PCP Write Invoke ID (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Write Invoke ID (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

The PCP Write Invoke ID object allows the user to set the PCP Invoke ID that the PCP Write Data object (3408_{hex}) will access.

B 19.16 Object $340F_{hex}$: PCP Read Invoke ID

a. Object description

Index	$340F_{hex}$
Name	PCP Read Invoke ID
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0

Subindex	**1_{hex}**
Description	PCP Read Invoke ID (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Read Invoke ID (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

The PCP Read Invoke ID object allows the user to set the PCP Invoke ID that the PCP Read Data object ($340B_{hex}$) will access.

B 19.17 Object 3410_{hex}: PCP Write Data Confirmation

a. Object description

Index	3410_{hex}
Name	PCP Write Data Confirmation
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Write Data Confirmation Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Write Data Confirmation (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

1. This object can be read to determine the cause of the SDO abort when using the PCP Write Data object.
2. The status byte of object 3402_{hex} (PCP Module Status) must be examined to determine the type of response. The format is as follows:

c. Format

Status byte definition (see Object 3402_{hex})		
Bit	**Name**	**Description**
0	Module COMM Error	When set, communication with the module is no longer possible.
1	PCP Error	A PCP service-specific error has occurred. See the Error Class and Error Code bytes.
2	PCP Channel Busy	A PCP transaction is already in progress, such as from an SDO or explicit request.
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).
4	PCP Parameter Error	The user has sent a service with invalid parameters.
5	PCP Reject Error	The PCP module has rejected the PCP request. See PCP reject response below.
6	PCP Abort Error	The PCP module has aborted the PCP request. See PCP abort response below.
7	Reserved	

d. Successful responses

Byte 1	Invoke ID
Byte 2	Status

e. PCP error response

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

f. PCP reject response

Byte 1	Invoke ID
Byte 2	Status
Byte 3	PDU type
Byte 4	Reject code

g. PCP abort response

Byte 1	Invoke ID
Byte 2	Status
Byte 3	Abort ID
Byte 4	Reason code
Bytes 5-7	Abort detail

B 19.18 Object 3411_{hex}: PCP Read Data Confirmation

a. Object description

Index	3411_{hex}
Name	PCP Read Data Confirmation
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	PCP Read Data Confirmation Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached PCP modules**
Description	PCP Read Data Confirmation (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

1. This object can be read to determine the cause of the SDO abort when using the PCP Read Data object.
2. 2. The status byte of object 3402_{hex} (PCP Module Status) must be examined to determine the type of response. The format is as follows:

c. Format

Status byte definition (see Object 3402_{hex})		
Bit	Name	Description
0	Module COMM Error	When set, communication with the module is no longer possible.
1	PCP Error (see Error Codes)	A PCP service-specific error has occurred. See the Error Class and Error Code bytes.
2	PCP Channel Busy	A PCP transaction is already in progress, such as from an SDO or explicit request.
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).
4	PCP Parameter Error	The user has sent a service with invalid parameters.
5	PCP Reject Error	The PCP module has rejected the PCP request. See PCP reject response below.
6	PCP Abort Error	The PCP module has aborted the PCP request. See PCP abort response below.
7	Reserved	

d. Successful responses

Byte 1	Invoke ID
Byte 2	Status

e. PCP error response

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

f. PCP reject response

Byte 1	Invoke ID
Byte 2	Status
Byte 3	PDU type
Byte 4	Reject code

g. PCP abort response

Byte 1	Invoke ID
Byte 2	Status
Byte 3	Abort ID
Byte 4	Reason code
Bytes 5-7	Abort detail

B 20 Serial communication objects

Index (hex)	Object	Name	Type	Access	NV
3500	ARRAY	Module Type	UNSIGNED8	RO	V
3501	ARRAY	Module Status	BOOLEAN	RO	
3502	ARRAY	Serial Status Word	UNSIGNED16	RO	
3503	ARRAY	Serial Control Word	UNSIGNED16	RWW	V
3504	ARRAY	Serial Receive Data	DOMAIN	RO	V
3505	ARRAY	Serial Transmit Data	DOMAIN	RWW	V
3506	ARRAY	Serial Receive Data Fragment	SERIAL_RECEIVE	RO	V
3507	ARRAY	Serial Transmit Data Fragment	SERIAL_TRANSMIT	RWW	V
3508	ARRAY	Protocol	UNSIGNED8	RW	NV
3509	ARRAY	Baud Rate	UNSIGNED8	RW	NV
350A	ARRAY	Data Width	UNSIGNED8	RW	NV
350D	ARRAY	Error Pattern	UNSIGNED8	RW	NV
350E	ARRAY	First Delimiter	UNSIGNED8	RW	NV
350F	ARRAY	Second delimiter	UNSIGNED8	RW	NV
3510	ARRAY	3964R Priority	UNSIGNED8	RW	NV
3511	ARRAY	Output Type	UNSIGNED8	RW	NV
3512	ARRAY	DTR Control	UNSIGNED8	RW	NV
3513	ARRAY	Rotation Switch	UNSIGNED8	RW	NV
3514	ARRAY	XON Pattern	UNSIGNED8	RW	NV
3515	ARRAY	XOFF Pattern	UNSIGNED8	RW	NV
351C	ARRAY	SCO Enable	BOOLEAN	RW	NV

B 20.1 Object 3500_{hex}: Serial Module Type

a. Object description

Index	3500_{hex}
Name	Serial Module Type
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Type Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Type Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	No

c. Serial module type

This value indicates the type of the serial module.

0 = RS-232

1 = RS-485

2 = RS-422

B 20.2 Object 3501_{hex}: Serial Module Status

a. Object description

Index	3501_{hex}
Name	Serial Module Status
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Status Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	BOOLEAN
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Status Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	BOOLEAN
Default value	No

c. Serial module status

0 = OK

1 = Faulted

B 20.3 Object 3502_{hex}: Serial Status Word

a. Object description

Index	3502_{hex}
Name	Serial Status Word
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Status Word Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Status Word Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

c. Serial status word

Byte 1	Name
15 **MSB**	Number of Received Characters (Mode Dependent)
14	Number of Received Characters (Mode Dependent)
13	Number of Received Characters (Mode Dependent)
12	Number of Received Characters (Mode Dependent)
11	Number of Received Characters (Mode Dependent)
10	Number of Received Characters (Mode Dependent)
9	Number of Received Characters (Mode Dependent)
8	Number of Received Characters (Mode Dependent)

Byte 2	Name
7	Reserved
6	Transmit Buffer Not Full
5	Transmit Buffer Full
4	Receive Buffer Full
3	Re-Init Executed
2	Send Error
1	Received Error
0 **LSB**	Receive Buffer Not Empty

B 20.4 Object 3503_{hex}: Serial Control Word

a. Object description

Index	3503_{hex}
Name	Serial Control Word
Object	ARRAY
Data type	UNSIGNED16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Control Word Module (1)
Access	RWW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED16
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Control Word Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED16
Default value	No

c. Serial control word

Byte 1	Name
15 **MSB**	Reserved
14	Reserved
13	Reserved
12	Reserved
11	Reserved
10	Reserved
9	Reserved
8	Reserved

Byte 2	Name
7	DTR
6	Reserved
5	Reserved
4	Reserved
3	Execute Re-Init
2	Reset Send Error
1	Reset Received Error
0 **LSB**	Reserved

B 20.5 Object 3504_{hex}: Serial Receive Data

a. Object description

Index	3504_{hex}
Name	Serial Receive Data
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Receive Data Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Receive Data Module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	DOMAIN
Default value	No

c. Format

Byte 1	Data length (bytes received)
Byte 2	Receive data, if necessary
...	...
Byte N	Receive data, if necessary

B 20.6 Object 3505_{hex}: Serial Transmit Data

a. Object description

Index	3505_{hex}
Name	Serial Transmit Data
Object	ARRAY
Data type	DOMAIN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Transmit Data Module (1)
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	DOMAIN
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Transmit Data Module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	DOMAIN
Default value	No

c. Format

Byte 1	Data length (bytes to transmit)
Byte 2	Transmit data, if necessary
...	...
Byte N	Transmit data, if necessary

B 20.7 Object 3506_{hex}: Serial Receive Data Fragment

a. Object description

Index	3506_{hex}
Name	Serial Receive Fragment
Object	ARRAY
Data type	SERIAL_RECEIVE(42_{hex})
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Receive Data Fragment Module (1)
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	SERIAL_RECEIVE(42_{hex})
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Receive Data Fragment Module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	SERIAL_RECEIVE(42_{hex})
Default value	No

See also Section "Fragmented services" on page C-7.

B 20.8 Object 3507_{hex}: Serial Transmit Data Fragment

a. Object description

Index	3507_{hex}
Name	Serial Transmit Data Fragment
Object	ARRAY
Data type	SERIAL_TRANSMIT(43_{hex})
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-63
Default value	0

Subindex	**1_{hex}**
Description	Serial Transmit Data Fragment Module (1)
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	SERIAL_TRANSMIT(43_{hex})
Default value	No
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Transmit Data Fragment Module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	SERIAL_TRANSMIT(43_{hex})
Default value	No

See also Section "Fragmented services" on page C-7.

B 20.9 Object 3508_{hex}: Serial Protocol

a. Object description

Index	3508_{hex}
Name	Serial Protocol
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Protocol Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Protocol Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

c. Protocol

Code	Meaning
00_{hex}	**Transparent (default)**
01_{hex}	End-to-end
02_{hex}	Dual buffer
03_{hex}	3964R
04_{hex}	XON/XOFF

B 20.10 Object 3509_{hex}: Serial Baud Rate

a. Object description

Index	3509_{hex}
Name	Serial Baud Rate
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Baud Rate Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	07_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Baud Rate Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	07_{hex}

c. Baud rate

Code	Value
00_{hex}	110
01_{hex}	300
02_{hex}	600
03_{hex}	1200
04_{hex}	1800
05_{hex}	2400
06_{hex}	4800
07_{hex}	**9600 (default)**
08_{hex}	19200

B 20.11 Object $350A_{hex}$: Serial Data Width

a. Object description

Index	$350A_{hex}$
Name	Serial Data Width
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Data Width Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	02_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Data Width Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	02_{hex}

c. Data width:

Code	Meaning		
Data	**Bits**	**Parity**	**Stop bits**
00_{hex}	7	Even	1
01_{hex}	7	Odd	1
02_{hex}	**8**	**Even**	**1 (default)**
03_{hex}	8	Odd	1
04_{hex}	8	Without 1	1
05_{hex}	7	Without 1	1
06_{hex}	7	Even	2
07_{hex}	7	Odd	2
08_{hex}	8	Even	2
09_{hex}	8	Odd	2
$0A_{hex}$	8	Without 2	2
$0B_{hex}$	7	Without 2	2

B 20.12 Object $350D_{hex}$: Serial Error Pattern

a. Object description

Index	$350D_{hex}$
Name	Serial Error Pattern
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Error Pattern Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	24_{hex} (‘$’)
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Error Pattern Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	24_{hex} (‘$’)

c. Error pattern

Code	Meaning
24_{hex}	**$ (default)**
xx_{hex}	Any character

B 20.13 Object $350E_{hex}$: Serial First Delimiter

a. Object description

Index	$350E_{hex}$
Name	Serial First Delimiter
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial First Delimiter Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	$0D_{hex}$ (CR) Carriage Return
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial First Delimiter Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	$0D_{hex}$ (CR) Carriage Return

c. First delimiter

Code	**Meaning**
$0D_{hex}$	**Carriage Return (CR), (default)**
xx_{hex}	Any character

B 20.14 Object $350F_{hex}$: Serial Second Delimiter

a. Object description

Index	$350F_{hex}$
Name	Serial Second Delimiter
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Second Delimiter Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	$0A_{hex}$ (LF) Line Feed
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial First Delimiter Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	$0A_{hex}$ (LF) Line Feed

c. Second delimiter

Code	Meaning
$0A_{hex}$	**Line feed (LF), (default)**
xx_{hex}	Any character

B 20.15 Object 3510_{hex}: 3964R Priority

a. Object description

Index	3510_{hex}
Name	3964R Priority
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	3964R Priority Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0
...	...
Subindex	**(N) = Number of attached serial modules**
Description	3964R Priority Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0

c. 3964R priority

Code	**Meaning**
00_{hex}	**Low priority (default)**
01_{hex}	High priority

B 20.16 Object 3511_{hex}: Serial Output Type

a. Object description

Index	3511_{hex}
Name	Serial Output Type
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Output Type Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	0 for 232; 1 for 485; 2 for 422
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Output Type Module (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	0 for 232; 1 for 485; 2 for 422

c. Output type

Code	Meaning
00_{hex}	**RS-232 (default for the RS-232 module type)**
01_{hex}	**RS-485 (default for the RS-485 module type)**
02_{hex}	RS-422

B 20.17 Object 3512_{hex}: Serial DTR Control

a. Object description

Index	3512_{hex}
Name	Serial DTR Control
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial DTR Control Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	00_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial DTR Control Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	00_{hex}

c. DTR control (only valid for RS-232 type)

Code	**Meaning**
00_{hex}	**Automatic (default)**
01_{hex}	Via process data

B 20.18 Object 3513_{hex}: Serial Rotation Switch

a. Object description

Index	3513_{hex}
Name	Serial Rotation Switch
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Rotation Switch Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	00_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Rotation Switch Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	00_{hex}

c. Rotation switch

Code	**Meaning**
00_{hex}	**No rotation (default)**
01_{hex}	Rotation

B 20.19 Object 3514_{hex}: Serial XON Pattern

a. Object description

Index	3514_{hex}
Name	Serial XON Pattern
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial XON Pattern Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	11_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial XON Pattern Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	11_{hex}

c. XON pattern

Code	Meaning
11_{hex}	**(Default)**
xx_{hex}	Any character (not the same as XOFF pattern)

B 20.20 Object 3515_{hex}: Serial XOFF Pattern

a. Object description

Index	3515_{hex}
Name	Serial XOFF Pattern
Object	ARRAY
Data type	UNSIGNED8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial XOFF Pattern Module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	UNSIGNED8
Default value	13_{hex}
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial XOFF Pattern Module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	UNSIGNED8
Default value	13_{hex}

c. XOFF pattern

Code	**Meaning**
13_{hex}	**(Default)**
xx_{hex}	Any character (not the same as XON pattern)

B 20.21 Object $351C_{hex}$: Serial Module Enable

a. Object description

Index	$351C_{hex}$
Name	Serial Module Enable
Object	ARRAY
Data type	BOOLEAN
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of Serial Modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0-8
Default value	0

Subindex	**1_{hex}**
Description	Serial Enable Module (1)
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	BOOLEAN
Default value	1
...	...
Subindex	**(N) = Number of attached serial modules**
Description	Serial Enable Module (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	BOOLEAN
Default value	1

Module Enable: This object allows the user to disable the serial module so the PCP objects can be used to communicate with the module.

0 = Disabled

1 = Enabled

C Appendix: PCP implementation

C 1 General overview

The IL CAN BK-TC-PAC allows PCP information access through SDOs or PDOs. Using SDOs, the user has the choice of configuring the individual PCP interface objects and simply reading or writing the raw data, or the user can use the request/response method to access the PCP information. When accessing PCP information through process data the request/response method is used. But the user must fragment the request due to the eight byte length limitation imposed by CANopen. The response back from the module is also fragmented for the same reason. The handshaking required for the fragmentation is described in detail in the following sections.

C 2 PCP interface objects

Several PCP objects are available to the user. The PCP objects are defined as follows:

Index	Object	Name	Type	Access
3400	ARRAY	PCP PDU Size	UNSIGNED8	RO
3401	ARRAY	PCP PCP Size	UNSIGNED8	RO
3402	ARRAY	PCP Status	UNSIGNED8	RO
3403	ARRAY	PCP Request	DOMAIN	RW
3404	ARRAY	PCP Response	DOMAIN	RO
3405	ARRAY	PCP Module	UNSIGNED8	RW
3406	ARRAY	PCP Write Index	UNSIGNED16	RW
3407	ARRAY	PCP Write SubIndex	UNSIGNED8	RW
3408	ARRAY	PCP Write Data	DOMAIN	RW
3409	ARRAY	PCP Read Index	UNSIGNED16	RW
340A	ARRAY	PCP Read SubIndex	UNSIGNED8	RW
340B	ARRAY	PCP Read Data	DOMAIN	RO
340C	ARRAY	PCP Request Fragment	PCP_FRAG_REQUEST	RW
340D	ARRAY	PCP Response Fragment	PCP_FRAG_RESPONSE	RO
340E	ARRAY	PCP Write Invoke ID	UNSIGNED8	RW
340F	ARRAY	PCP Read Invoke ID	UNSIGNED8	RW
3410	ARRAY	PCP Write Data Confirmation	DOMAIN	RO
3411	ARRAY	PCP Read Data Confirmation	DOMAIN	RO

A complete list of all objects with full descriptions can be found in Appendix A.

C 3 PCP access using SDOs

The user has the option of two methods for accessing PCP information. The difference between the two methods is that method 1 configures the Invoke ID, Index, and Subindex individually, then the user simply writes or reads the data to be transferred. In method 2 a request is sent tot he PCP Request Object and the response is read using the PCP Response Object. The request must contain the Service, Index, Subindex, Invoke ID and data. Method 1 is more efficient when reading or writing the same object repeatedly. A good example would be reading received data from, and writing transmit data to, serial modules. Method 1 is more efficient for operations such as configuring a module where several reads and writes to different objects must be completed. Whichever method is chosen, the user should only use on method at a time for each PCP module. After the PCP access is complete the user can choose to use the same method or a different method.

C 3.1 PCP read data/write data implementation

Method 1 requires that the user set the PCP Read or PCP Write objects (Invoke ID, Index, Subindex) depending on whether a read service or write service is desired then access the corresponding data object to actually transfer the data. The PCP module to access is indicated by the CANopen Subindex of each of the PCP Access Objects (i.e. Subindex 1 indicates PCP module 1, Subindex 2 indicates PCP module 2 etc.).
To do a write, the user sets the PCP Write Invoke ID, PCP Write Index and PCP Write Subindex. Then data is written to the PCP Write Data object. The first byte must contain the the number of bytes to be written. If the write succeeds, the SDO write is confirmed. If the write fails, the user can read the result from the PCP Write Confirmation object.
To do a read, the user sets the PCP Read Invoke ID, PCP Read Index and PCP Read Subindex. Then the user must read data from the PCP Read Data object. If the read is valid, the number of data bytes is returned followed by the data read. If an error occurs, the user can get the result in the PCP Read Confirmation object.

An example of using method 1 to write serial data "Hello" to an RS-232 module would be as follows:

Where x = PCP module number of the RS-232 module.

1. Set the PCP Write Invoke ID Object Subindex x to 0.
2. Set the PCP Write Index Object Subindex x to $5FE0_{hex}$.
3. Set the PCP Write SubIndex Object Subindex x to 0.
4. Send the data to the PCP Write Data Object Subindex x to 05_{hex}, 48_{hex}, 65_{hex}, $6C_{hex}$, $6C_{hex}$, $6F_{hex}$.

The BK will send the PCP request to the indicated PCP module, wait for a response and then return a successful response or error response. If an error is returned, the user should read the PCP Write Data Confirmation Object Subindex x for the details of why the PCP write fails. The user should take care when using this method, as the CANopen channel cannot process any further data when a response is pending. This should not be a factor when configuring the module, but should be taken into consideration when running process data.

C 3.2 PCP request/response implementation

Method 2 requires the user to sent the module a PCP request, then read the PCP response to get the status and or data back. The user fist creates a request and then uses an SDO to send the request to the bus coupler. The bus coupler will first examine the request to make sure that is of proper length and has a valid service. If there is an error, the bus coupler returns an error immediately. If the request is valid, the bus coupler sends the request to the correct PCP module and continues to process other CANopen data. The user must then read the PCP Response Object to get the response and the data if the request was a read. The bus coupler will return a busy status until the PCP module returns the response to the request. Depending on the module configuration and the PCP channel size, several reads may have to be sent before the PCP request is complete. Alternatively the user may insert a delay between the request write and response read. The time to complete most read/writes will be in the range of a few milliseconds, again determined by the configuration, PCP channel size, and request type.

a. Format of request

Services - No Invoke ID

Byte 1	Service
Byte 2	Index low
Byte 3	Index high
Byte 4	Subindex
Byte 5	Length
Byte 6	Data block, if necessary
...	...
Byte N	Data block, if necessary

Services - With Invoke ID

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

b. Successful responses

Services - No Invoke ID

Byte 1	Service
Byte 2	Status
Byte 3	Length
Byte 4	Data block, if necessary
...	...
Byte N	Data block, if necessary

Services - With Invoke ID

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Length
Byte 5	Data block, if necessary
...	...
Byte N	Data block, if necessary

c. Error responses

Services - No Invoke ID

Byte 1	Service
Byte 2	Status
Byte 3	Error class
Byte 4	Error code
Byte 5	Additional error code 1 LSB
Byte 6	Additional error code 1 MSB

Services - With Invoke ID

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Error class
Byte 5	Error code
Byte 6	Additional error code LSB
Byte 7	Additional error code MSB

d. Example 1

An example of setting the RS-232 baud rate to 9600 using the request/response method would be as shown in the following example:

1 Send the following to the PCP Request Object (Index 3403_{hex} Subindex x):

Byte 1	Service $0A_{hex}$ (Write with Invoke ID)
Byte 2	Invoke ID = 0 (Invoke ID 0)
Byte 3	Index low = FF_{hex} (Index for baud is $5FFF_{hex}$)
Byte 4	Index high = $5F_{hex}$
Byte 5	Subindex = 02_{hex} (Baud rate is Subindex 2)
Byte 6	Length = 01_{hex} (One byte)
Byte 7	Data = 07_{hex} (Baud rate = 9600)

Or

$0A_{hex}$, 00_{hex}, 01_{hex}, FF_{hex}, $5f_{hex}$, 02_{hex}, 01_{hex}, 07_{hex}

2 The following successful response should be read from the PCP Response Object (Index 3404_{hex} Subindex x):

Byte 1:	Service $0A_{hex}$ (Write with Invoke ID)
Byte 2:	Invoke ID = 00_{hex} (Invoke ID 0)
Byte 3:	Status = 00_{hex} (No error)

Or

$0A_{hex}$,00_{hex},00_{hex}

e. Example 2

An example of reading the RS-232 data width using the request/response method would be as shown in the following example:

1 Send the following to the PCP Request Object (Index 3403_{hex} Subindex x):

Byte 1	Service 09_{hex} (Write with Invoke ID)
Byte 2	Invoke ID = 0 (Invoke ID 0)
Byte 3	Index low = FF_{hex} (Index for data width is $5FFF_{hex}$)
Byte 4	Index high = $5F_{hex}$
Byte 5	Subindex = 03_{hex} (Data width is Subindex 3)
Byte 6	Length = 00_{hex} (No data)

Or

09_{hex}, 00_{hex}, 01_{hex}, FF_{hex}, $5f_{hex}$, 03_{hex}, 00_{hex}

2 The successful response should be as follows:

Byte 1	Service = 09_{hex} (Read with Invoke ID)
Byte 2	Invoke ID = 00_{hex} (Invoke ID 0)
Byte 3	Status = 00_{hex} (No error)
Byte 4	Length = 01_{hex} (One byte)
Byte 5	Data = 02_{hex} (Default data width format)

Or

09_{hex}, 00_{hex}, 00_{hex}, 01_{hex}, 02_{hex}

C 4 PCP fragmentation implementation

C 4.1 PCP communication over CANopen process data

PCP transfer can also be completed by using process data. This uses the same method as the request/response method previously described, with the exception that request and response must be broken down into fragments of eight bytes.

C 4.2 Fragmented services

There are four transfer fragments that are distinguished by the service byte:

– Start fragment
– Middle fragment
– Last fragment
– Abort/error fragment

a. Start fragment

Services - No Invoke ID

The start fragment has the following format:

Byte 1	Service
Byte 2	Index low
Byte 3	Index high
Byte 4	Subindex
Byte 5	Length
Byte 6	Data block, if necessary
...	...
Byte N	Data block, if necessary

Services - With Invoke ID

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

Service byte 1, definition of the start fragment

Bit	Name	Description	
0–3	PCP Service	hex values	PCP service
		00_{hex}	No action (clears input bytes in response)
		01_{hex}	Read
		02_{hex}	Write
		03_{hex}	Read PDU length
		04_{hex} - 08_{hex}	Reserved
		09_{hex}	Read with Invoke ID
		$0A_{hex}$	Write with Invoke ID
		$0B_{hex}$ - $0F_{hex}$	Reserved
4	No Frag/Frag	0 = Does not fragment, 1 = Fragments	
5	0 (zero)	00 = Start fragment	
6	0 (zero)	00 = Start fragment	
7	Request/Response	0 = Request, 1 = 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.	

b. Middle fragment

The middle fragment has the following format:

	Format
Byte 1	Service
Byte 2	Data block, if necessary
...	...
Byte 8	Data block, if necessary

Service byte 1, definition of the middle fragment

Bit	Name	Description
0–4	Fragment Number (01_{hex}- $01F_{hex}$)	Count = 1–$1F_{hex}$ Fragment number; if more fragments are needed the fragment number should roll over to 0.
5	1	Bit 5 and 6 combined, determine fragment type: 1= Middle fragment
6	0	Bit 5 and 6 combined, determine fragment type: 0 = Middle fragment
7	Request/Response	0 = Request 1 = Response

c. Last fragment

The last fragment has the following format:

	Format
Byte 1	Service
Byte 2	Data block, if necessary
...	...
Byte 8	Data block, if necessary

Service byte1, definition of the last fragment

Bit	Name	Description
0–4	Reserved	
5	0	Fragment type: 10 = Last fragment
6	1	Fragment type: 10 = Last fragment
7	Request/Response	0 = Request 1 = Response

d. Abort/error fragment

Service byte 1, Definition of a PCP abort/error fragment

Bit	Name	Description
0–4	Reserved	
5	1	Fragment type: 11 = Abort/error fragment
6	1	Fragment type: 11 = Abort/error fragment
7	Request/Response	0 = Request 1 = Response

Status byte 2, definition of the PCP abort/error fragment

Bit	Name	Description
0	Module COMM Error	When set, communication with the module is no longer possible.
1	PCP Error	A PCP service-specific error has occurred. See the Error Class and Error Code bytes.
2	PCP Channel Busy	A PCP transaction is already in progress, such as from an SDO or explicit request.
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).
4	PCP Parameter Error	PCP parameter error: The user has sent a service with invalid parameters.
5	PCP Reject Error	The PCP module has rejected the PCP request. See PCP reject response below.
6	PCP Abort Error	The PCP module has aborted the PCP request. See PCP abort response below.
7	Reserved	

e. PCP responses

PCP error response - No Invoke-ID

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

PCP error response - With Invoke-ID

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

PCP reject response - No Invoke-ID

Byte 1	Service
Byte 2	Status
Byte 3	PDU type
Byte 4	Reject code

PCP reject response - With Invoke ID

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	PDU type
Byte 5	Reject code

PCP abort response - No Invoke ID

Byte 1	Service
Byte 2	Status
Byte 3	Abort ID
Byte 4	Reason code
Bytes 5-8	Abort detail

PCP abort response - With Invoke ID

Byte 1	Service
Byte 2	Invoke ID
Byte 3	Status
Byte 4	Abort ID
Byte 5	Reason code
Bytes 6-9	Abort detail

f. Example: Fragmented services - Read

IDLE

Master -> Slave

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
00_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
00_{hex}	xx	xx	xx	xx	xx	xx	xx

START

Master -> Slave

Service	**Mod. No.**	**Index high**	**Index low**	**Sub. Index**	**Length**	**-**	**-**
01_{hex}	01_{hex}	$5F_{hex}$	$E0_{hex}$	00_{hex}	xx	xx	xx

Slave -> Master

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

Master -> Slave

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
91_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

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

Master -> Slave

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
$A1_{hex}$	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	**D12**	**D13**	**D14**	**D15**	**-**	**-**	**-**
$C0_{hex}$	$0D_{hex}$	$1E_{hex}$	$0F_{hex}$	10_{hex}	xx	xx	xx

Master -> Slave							
Service	-	-	-	-	-	-	-
$C0_{hex}$	xx	xx	xx	xx	xx	xx	xx
IDLE							
Master -> Slave							
Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx
Slave -> Master							
Service	-	-	-	-	-	-	-
00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

g. Example: Fragmented services - Write

IDLE

Master -> Slave

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx

START

Master -> Slave

Service	**Mod. No.**	**Index low**	**Index high**	**Sub. Index**	**Length**	**D0**	**D1**
12_{hex}	01_{hex}	$5F_{hex}$	$E0_{hex}$	00_{hex}	10_{hex}	01_{hex}	02_{hex}

Slave -> Master

Service	-	-	-	-	-	-	-
12_{hex}	xx	xx	xx	xx	xx	xx	xx

Master -> Slave

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

Slave -> Master

Service	-	-	-	-	-	-	-
21_{hex}	xx	xx	xx	xx	xx	xx	xx

Master -> Slave

Service	**D9**	**D10**	**D11**	**D12**	**D13**	**D14**	**D15**
40_{hex}	09_{hex}	$0A_{hex}$	$0B_{hex}$	$0C_{hex}$	$0D_{hex}$	$0E_{hex}$	$0F_{hex}$

Slave -> Master

Service	**Status**	-	-	-	-	-	-
82_{hex}	00_{hex}	xx	xx	xx	xx	xx	xx

Master - > Slave

Service	-	-	-	-	-	-	-
82_{hex}	xx	xx	xx	xx	xx	xx	xx

IDLE

Master - > Slave

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

h. Example 1: Fragmented error handling

Error = DI or DO break, error occurs during a read.

IDLE

Master -> Slave

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx

START

Master -> Slave

Service	**Mod. No.**	**Index high**	**Index low**	**Sub. index**	**Length**	**-**	**-**
01_{hex}	01_{hex}	$5F_{hex}$	$E0_{hex}$	00_{hex}	xx	xx	xx

Slave -> Master

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

Master -> Slave

Service	-	-	-	-	-	-	-
91_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

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

Master -> Slave

Service	-	-	-	-	-	-	-
$A1_{hex}$	xx	xx	xx	xx	xx	xx	xx

DI or DO line break

Slave -> Master

Service	**D12**	**D13**	**D14**	**D15**	**-**	**-**	**-**
$E0_{hex}$	01_{hex}	xx	xx	xx	xx	xx	xx

Master -> Slave

Service	-	-	-	-	-	-	-
xx	xx	xx	xx	xx	xx	xx	xx

DI or DO line repaired

Continue as if no error, data is already buffered in BK

Slave -> Master

Service	**D12**	**D13**	**D14**	**D15**	**-**	**-**	**-**
$C0_{hex}$	$0D_{hex}$	$1E_{hex}$	$0F_{hex}$	10_{hex}	xx	xx	xx

Master -> Slave

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
$C0_{hex}$	xx	xx	xx	xx	xx	xx	xx

Data transfer is complete. User should transition back to IDLE state.

IDLE

Master -> Slave

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
00_{hex}	xx	xx	xx	xx	xx	xx	xx

Slave -> Master

Service	**-**	**-**	**-**	**-**	**-**	**-**	**-**
00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

i. Example 2: Fragmented error handling

Error = Loss of communications between PLC and BK during PCP write.

IDLE

Master -> Slave

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	Xx	xx	Xx	xx

Slave -> Master

Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	Xx	xx

START

Master -> Slave

Service	**Mod. No.**	**Index low**	**Index high**	**Sub. index**	**Length**	**D0**	**D1**
12_{hex}	01_{hex}	$5F_{hex}$	$E0_{hex}$	00_{hex}	10_{hex}	01_{hex}	02_{hex}

Slave -> Master

Service	-	-	-	-	-	-	-
12_{hex}	xx	xx	xx	xx	xx	xx	xx

Loss of communication - communication repaired

Continue as normal since PCP request is just being buffered on BK until fragmentation is complete.

Master -> Slave

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

Slave -> Master

Service	-	-	-	-	-	-	-
21_{hex}	xx	xx	xx	xx	xx	xx	xx

Master -> Slave

Service	**D9**	**D10**	**D11**	**D12**	**D13**	**D14**	**D15**
40_{hex}	09_{hex}	$0A_{hex}$	$0B_{hex}$	$0C_{hex}$	$0D_{hex}$	$0E_{hex}$	$0F_{hex}$

Slave -> Master

Service	**Status**	-	-	-	-	-	-
82_{hex}	00_{hex}	xx	xx	xx	xx	xx	xx

Master -> Slave

Service	-	-	-	-	-	-	-
82_{hex}	Xx	xx	xx	xx	xx	xx	xx

IDLE

Master -> Slave							
Service	-	-	-	-	-	-	-
00_{hex}	xx	xx	xx	xx	xx	xx	xx
Slave -> Master							
Service	-	-	-	-	-	-	-
00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}	00_{hex}

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

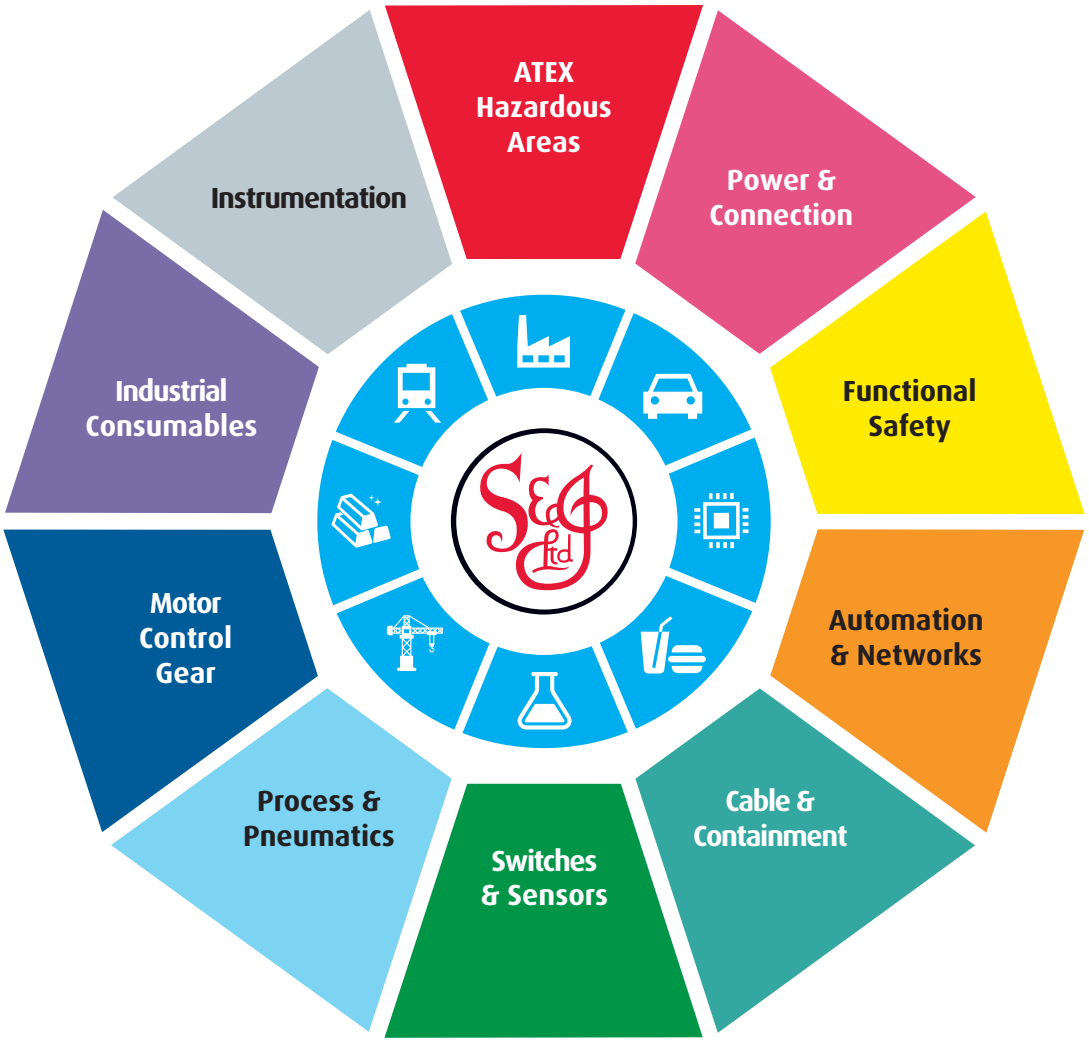
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

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