Inline bus coupler for CANopen®
IL CO BK(-XC)-PAC

User manual

PHOENIX CONTACT
INSPIRING INNOVATIONS

User manual

Inline bus coupler for CANopen® IL CO BK(-XC)-PAC

2017-11-07

Designation: UM EN IL CO BK(-XC)-PAC

Revision: 05

This user manual is valid for:

Designation	Order No.
IL CO BK-PAC	2702230
IL CO BK-XC-PAC	2702635

Please observe the following notes

User group of this manual

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

- Qualified electricians or persons instructed by them, who are familiar with applicable standards and other regulations regarding electrical engineering and, in particular, the relevant safety concepts.
- Qualified application programmers and software engineers, who are familiar with the safety concepts of automation technology and applicable standards.

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.

There are three different categories of personal injury that are indicated with a signal word.

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

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

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

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

This symbol and the accompanying text provide the reader with additional information or refer to detailed sources of information.

How to contact us

Internet

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

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

Subsidiaries

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

Published by

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

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

General terms and conditions of use for technical documentation

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

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

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

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

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

Table of contents

1 Inline CANopen® bus coupler 11

2 Installing a CANopen® system 13

2.1 The CANopen® system 13

2.2 Example topology of a CANopen® system 14

2.3 Network considerations 16

2.3.1 Wiring 16

2.3.2 Central grounding for conductors and cables 17

3 Startup 19

3.1 Introduction 19

3.2 Determining the maximum number of I/O terminals that can be connected 20

3.3 Configuring an Inline station 21

3.3.1 Bus coupler 21

3.3.2 Analog input terminals (AI) 24

3.3.3 Thermocouple and RTD terminals 27

3.3.4 Function terminals 28

3.3.5 Analog output terminals (AO) 28

3.4 Mapping the modules to the process data objects (PDOs) 29

3.4.1 Bus coupler 29

3.4.2 Electronic Data Sheet (EDS) file 29

3.5 I/O data transfer 29

3.5.1 Communication 30

3.5.2 Standard objects for digital, analog, and special function modules 31

3.6 Behavior in the event of an error 35

3.6.1 Objects for controlling the output behavior in the event an error 35

3.6.2 Configuring the behavior in the event of an error 35

3.6.3 Changing the fault response mode 37

3.6.4 ... fault ... mode objects, indexes 3003_{hex} to $300A_{hex}$ 38

4 Diagnostics 41

4.1 Diagnostics and status indicators 41

4.2 Alarm message and diagnostic objects 41

4.2.1 Alarm message 41

4.2.2 Diagnostic information 42

4.2.3 Station status object, index 3101_{hex} 43

4.2.4 Inline faulted module object, index 3102_{hex} 45

4.2.5 Examples of requesting diagnostic information 45

4.2.6 Fault modes 48

4.2.7 Latched diagnostic data 49

4.2.8 Inline interface control byte 49

4.2.9 Error states 50

4.2.10 Error codes of analog inputs, thermocouples and RTD modules 50

A Appendix: CANopen® objects and indexes 51

A 1 General overview 51

A 2 Supported codes 52

A 2.1 SDO abort codes 52

A 2.2 Emergency error codes 53

A 3 Emergency object data 53

A 4 Predefined connection set 54

A 4.1 Broadcast objects 54

A 4.2 Peer-to-peer objects 54

A 5 Data types in the object dictionary 55

A 5.1 Object dictionary structure 56

A 6 Object dictionary overview 56

A 6.1 Objects specific to the communication profile 56

A 6.2 Manufacturer-specific I/O objects 58

A 6.3 Inline configuration objects 58

A 6.4 Inline interface objects 58

A 6.5 Inline module objects 59

A 6.6 Inline special function module objects 59

A 6.7 Objects for serial communication 59

A 6.8 Digital input objects 60

A 6.9 Digital output objects 60

A 6.10 Analog input objects 60

A 6.11 Analog output objects 61

A 6.12 Analog input configuration objects 61

A 6.13 Analog output configuration objects 61

A 6.14 PCP interface objects 61

A 7 Objects specific to the communication profile 62

A 7.1 Object 1000_{hex}: Device type 62

A 7.2 Object 1001_{hex}: Error register 63

A 7.3 Object 1002_{hex}: Manufacturer status register 64

A 7.4 Object 1003_{hex}: Pre-defined error field 64

A 7.5 Object 1005_{hex}: COB-ID SYNC message 66

A 7.6 Object 1008_{hex}: Manufacturer device name 67

A 7.7 Object 1009_{hex}: Manufacturer hardware version 68

A 7.8 Object $100A_{hex}$: Manufacturer software version 68

A 7.9 Object $100C_{hex}$: Guard time 69

A 7.10 Object $100D_{hex}$: Life time factor 69

A 7.11 Object 1010_{hex}: Store parameters 70

A 7.12 Object 1011_{hex}: Restore default parameters 72

A 7.13 Object 1012_{hex}: COB-ID time stamp 75

A 7.14 Object 1014_{hex}: COB-ID emergency message 76

A 7.15 Object 1015_{hex}: Inhibit time EMCY 77

A 7.16 Object 1016_{hex}: Consumer heartbeat time 78

A 7.17 Object 1017_{hex}: Producer heartbeat time 79

A 7.18 Object 1018_{hex}: Identity object ... 80
A 7.19 Object 1020_{hex}: Verify configuration ... 82
A 7.20 Object 1027_{hex}: Module list ... 83
A 7.21 Object 1029_{hex}: Error behavior ... 84
A 7.22 Objects 1200_{hex} to 1201_{hex}: Server SDO parameter ... 85
A 7.23 Objects 1400_{hex} to $141F_{hex}$: RPDO communication parameter ... 87
A 7.24 Objects 1600_{hex} to $161F_{hex}$: RPDO mapping parameter ... 90
A 7.25 Objects 1800_{hex} to $181F_{hex}$: TPDO communication parameter ... 92
A 7.26 Objects $1A00_{hex}$ to $1A1F_{hex}$: TPDO mapping parameter ... 94

A 8 Manufacturer-specific I/O objects ... 96
A 8.1 Object 2000_{hex}: Digital input latch enable (8 bits) ... 96
A 8.2 Object 2001_{hex}: Digital input latch state (8 bits) ... 97
A 8.3 Object 2400_{hex}: Analog input range, remanent ... 98
A 8.4 Object 2401_{hex}: Analog input range, mapping ... 99
A 8.5 Object 2410_{hex}: Analog output response ... 100

A 9 Inline configuration objects for reading the Inline local bus again ... 101
A 9.1 Object 3000_{hex}: Reconfigure I/O ... 101

A 10 Inline configuration objects for setting the bus coupler behavior in the event of an error ... 102
A 10.1 Object 3002_{hex}: Inline fault mode ... 102
A 10.2 Object 3003_{hex}: Inline CRC fault mode ... 103
A 10.3 Object 3004_{hex}: Inline peripheral fault mode ... 104
A 10.4 Object 3005_{hex}: Inline power fault mode ... 105
A 10.5 Object 3006_{hex}: Inline module change fault mode ... 106
A 10.6 Object 3007_{hex}: Inactive local bus fault mode ... 107
A 10.7 Object 3008_{hex}: Inline connection fault mode ... 108
A 10.8 Object 3009_{hex}: Inline faulted cycles mode ... 109
A 10.9 Object $300A_{hex}$: Processor power fault mode ... 110
A 10.10 Object $300F_{hex}$: Erase configuration ... 111

A 11 Inline interface objects ... 112
A 11.1 Object 3101_{hex}: Station status ... 112
A 11.2 Object 3102_{hex}: Inline faulted module ... 113
A 11.3 Object 3103_{hex}: Inline error max retry ... 114
A 11.4 Object 3104_{hex}: Inline number of modules ... 114
A 11.5 Object 3105_{hex}: Inline count of bits ... 115
A 11.6 Object 3106_{hex}: Inline count of bytes ... 115
A 11.7 Object 3109_{hex}: Inline Loop diagnostic count ... 116
A 11.8 Object $310A_{hex}$: Inline first faulted module ... 117
A 11.9 Object $310B_{hex}$: Inline last faulted module ... 117
A 11.10 Object $310C_{hex}$: Inline fault (latched) ... 118
A 11.11 Object $310D_{hex}$: Inline faulted module (latched) ... 118
A 11.12 Object $310E_{hex}$: Inline first faulted module (latched) ... 119
A 11.13 Object $310F_{hex}$: Inline last faulted module (latched) ... 119
A 11.14 Object 3110_{hex}: Inline power status ... 120
A 11.15 Object 3111_{hex}: Inline interface control byte ... 121

A 12 Inline module objects 122
A 12.1 Object 3200_{hex}: Inline module ID (stored) 122
A 12.2 Object 3201_{hex}: Inline module ID (current) 123
A 12.3 Object 3202_{hex}: Inline module IN data index 124
A 12.4 Object 3203_{hex}: Inline module IN data first subindex 125
A 12.5 Object 3204_{hex}: Inline module IN data last subindex 126
A 12.6 Object 3205_{hex}: Inline module OUT data index 127
A 12.7 Object 3206_{hex}: Inline module OUT data first subindex 128
A 12.8 Object 3207_{hex}: Inline module OUT data last subindex 129
A 13 Inline special function module objects 130
A 13.1 Object 3300_{hex}: Special function data size 130
A 13.2 Object 3301_{hex}: Special function status 131
A 13.3 Objects 3302_{hex} to $330F_{hex}$: Access to special function modules 132
A 13.4 Object 3302_{hex}: Special function IN data (1 byte) 133
A 13.5 Object 3303_{hex}: Special function OUT data (1 byte) 134
A 13.6 Object 3304_{hex}: Special function IN data (2 bytes) 135
A 13.7 Object 3305_{hex}: Special function OUT data (2 bytes) 136
A 13.8 Object 3306_{hex}: Special function IN data (3 bytes) 137
A 13.9 Object 3307_{hex}: Special function OUT data (3 bytes) 138
A 13.10 Object 3308_{hex}: Special function IN data (4 bytes) 139
A 13.11 Object 3309_{hex}: Special function OUT data (4 bytes) 140
A 13.12 Object $330A_{hex}$: Special function IN data (6 bytes) 141
A 13.13 Object $330B_{hex}$: Special function OUT data (6 bytes) 142
A 13.14 Object $330C_{hex}$: Special function IN data (8 bytes) 143
A 13.15 Object $330D_{hex}$: Special function OUT data (8 bytes) 144
A 13.16 Object $330E_{hex}$: Special function IN data (> 8 bytes) 145
A 13.17 Object $330F_{hex}$: Special function OUT data (> 8 bytes) 146
A 14 Objects for serial communication 147
A 14.1 Object 3500_{hex}: Serial module type 147
A 14.2 Object 3501_{hex}: Serial module status 148
A 14.3 Object 3502_{hex}: Serial status word 149
A 14.4 Object 3503_{hex}: Serial control word 151
A 14.5 Object 3504_{hex}: Serial receive data 153
A 14.6 Object 3505_{hex}: Serial transmit data 154
A 14.7 Object 3506_{hex}: Serial receive data fragment 155
A 14.8 Object 3507_{hex}: Serial transmit data fragment 156
A 14.9 Object 3508_{hex}: Serial protocol 157
A 14.10 Object 3509_{hex}: Serial baud rate 158
A 14.11 Object $350A_{hex}$: Serial data width 159
A 14.12 Object $350D_{hex}$: Serial error pattern 161
A 14.13 Object $350E_{hex}$: Serial first delimiter 162
A 14.14 Object $350F_{hex}$: Serial second delimiter 163
A 14.15 Object 3510_{hex}: 3964R priority 164
A 14.16 Object 3511_{hex}: Serial output type 165
A 14.17 Object 3512_{hex}: Serial DTR control 166
A 14.18 Object 3513_{hex}: Serial rotation switch 167

A 14.19 Object 3514_{hex}: Serial XON pattern 168
A 14.20 Object 3515_{hex}: Serial XOFF pattern 169
A 14.21 Object $351C_{hex}$: Serial module enable 170
A 15 Digital input objects 171
A 15.1 Object 6000_{hex}: Read input 8 bits 171
A 15.2 Object 6005_{hex}: Global interrupt enable digital 8 bits 172
A 15.3 Object 6006_{hex}: Interrupt mask any change 8 bits 173
A 15.4 Object 6100_{hex}: Read input 16 bits 174
A 15.5 Object 6106_{hex}: Interrupt mask any change 16 bits 175
A 15.6 Object 6120_{hex}: Read input 32 bits 176
A 15.7 Object 6126_{hex}: Interrupt mask any change 32 bits 177
A 16 Digital output objects 178
A 16.1 Object 6200_{hex}: Write output 8 bits 178
A 16.2 Object 6206_{hex}: Error mode output 8 bits 180
A 16.3 Object 6207_{hex}: Error value output 8 bits 181
A 16.4 Object 6300_{hex}: Write output 16 bits 182
A 16.5 Object 6306_{hex}: Error mode output 16 bits 183
A 16.6 Object 6307_{hex}: Error value output 16 bits 184
A 16.7 Object 6320_{hex}: Write output 32 bits 185
A 16.8 Object 6326_{hex}: Error mode output 32 bits 186
A 16.9 Object 6327_{hex}: Error value output 32 bits 187
A 17 Analog input objects 188
A 17.1 Object 6400_{hex}: Read analog input 8 bits 188
A 17.2 Object 6401_{hex}: Read analog input 16 bits 190
A 18 Analog output objects 191
A 18.1 Object 6410_{hex}: Write analog output 8 bits 191
A 18.2 Object 6411_{hex}: Write analog output 16 bits 193
A 19 Analog input configuration objects 195
A 19.1 Object 6421_{hex}: Analog input interrupt trigger selection 195
A 19.2 Object 6423_{hex}: Analog input global interrupt enable 197
A 19.3 Object 6424_{hex}: Analog input interrupt upper limit integer 198
A 19.4 Object 6425_{hex}: Analog input interrupt lower limit integer 199
A 19.5 Object 6426_{hex}: Analog input interrupt delta unsigned 200
A 20 Analog output configuration objects 201
A 20.1 Object 6443_{hex}: Analog output error mode 201
A 20.2 Object 6444_{hex}: Analog output error value integer 203
A 21 Objects that are stored retentively 204

B Appendix B: PCP implementation 207

B 1 General overview 207

B 2 Manufacturer-specific data types 207

B 2.1 PCP_FRAG_REQUEST record data types 207

B 2.2 PCP_FRAG_RESPONSE record specification 208

B 3 PCP interface objects 208

B 3.1 Object 3400_{hex}: PCP PDU size 209

B 3.2 Object 3401_{hex}: PCP module channel size 210

B 3.3 Object 3402_{hex}: PCP module status 211

B 3.4 Object 3403_{hex}: PCP request 213

B 3.5 Object 3404_{hex}: PCP response 215

B 3.6 Object 3405_{hex}: PCP module number 218

B 3.7 Object 3406_{hex}: PCP write index 220

B 3.8 Object 3407_{hex}: PCP write subindex 221

B 3.9 Object 3408_{hex}: PCP write data 222

B 3.10 Object 3409_{hex}: PCP read index 223

B 3.11 Object $340A_{hex}$: PCP read subindex 224

B 3.12 Object $340B_{hex}$: PCP read data 225

B 3.13 Object $340C_{hex}$: PCP request fragment 226

B 3.14 Object $340D_{hex}$: PCP response fragment 227

B 3.15 Object $340E_{hex}$: PCP write invoke ID 228

B 3.16 Object $340F_{hex}$: PCP read invoke ID 229

B 3.17 Object 3410_{hex}: PCP write data confirmation 230

B 3.18 Object 3411_{hex}: PCP read data confirmation 232

B 4 PCP access using SDOs 234

B 4.1 PCP implementation of read/write data 234

B 4.2 PCP implementation of the request/response 234

B 5 PCP fragmentation implementation 238

B 5.1 PCP communication via CANopen® process data 238

B 5.2 Fragmented services 238

C Default PDO mapping 251

C 1 Introduction 251

C 2 PDO mapping 251

C 2.1 First RPDO mapping (digital outputs) 251

C 2.2 First TPDO mapping (digital inputs) 252

C 2.3 Second RPDO mapping (analog outputs) 252

C 2.4 Second TPDO mapping (analog inputs) 253

C 2.5 Third RPDO mapping (analog outputs) 253

C 2.6 Third TPDO mapping (analog inputs) 253

C 2.7 Fourth RPDO mapping (analog outputs) 254

C 2.8 Fourth TPDO mapping (analog inputs) 254

1 Inline 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 power supply required for the connected station components.

The bus coupler supports all basic digital, analog and function terminals of the Inline product groups.

The hardware and firmware of the bus coupler complies with the CANopen® specifications of the CiA (CAN in Automation) and the technical specifications of the Inline product family.

For the technical data of the bus coupler, please refer to the terminal-specific data sheet. It can be downloaded at: phoenixcontact.net/products.

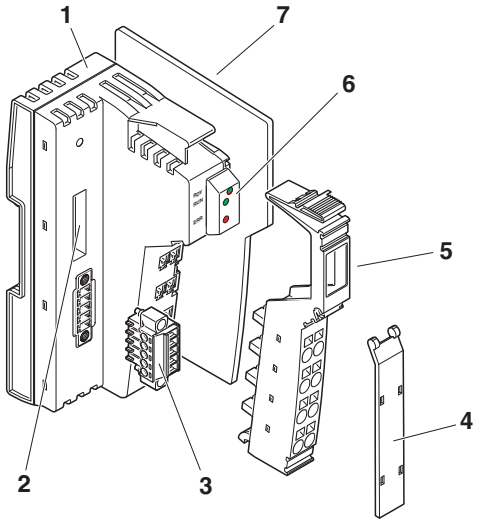


Figure 1-1 Components of the Inline CANopen® bus coupler

Key:

1 Electronics base

2 DIP switch for setting the device address, the transmission speed, and the termination resistor

3 Network connector

4 Marking field

5 Power connector

6 Diagnostics and status indicators

7 End plate

The electronic device data sheet (EDS) can be downloaded at: phoenixcontact.net/products.

Features

– General features

- Connection to CANopen® with 5-pos. MINI COMBICON connector
- Programmable fault response modes
- I/O configuration can be stored
- Auto configuration
- Extended diagnostics
- Diagnostics and status indicators
- IL CO BK-XC-PAC: Extreme-condition version. The terminal can be used under extreme ambient conditions (see data sheet).

– Inline features

- Up to 63 additional Inline devices can be connected
- Up to 16 PCP devices can be connected
- Automatic detection of the transmission speed in the local bus (500 kbps or 2 Mbps)

– CANopen® features

- Direct peer-to-peer communication
- Adjustable transmission speeds:
  Via DIP switches: 125 kbps, 500 kbps, 1 Mbps
  Via “auto baud rate”: 10 kbps, 20 kbps, 50 kbps, 125 kbps, 250 kbps, 500 kbps, 800 kbps, 1 Mbps
- Alarm messages
- Error states and substitute values
- Trigger modes: event, time, remote request
- Node and life guarding
- Heartbeat
- Analog interrupt triggers:
  - Upper limit
  - Lower limit
  - Delta, (unsigned delta)
- Supports two SDO servers simultaneously
- Integrated termination resistor can be switched on or off via DIP switch

2 Installing a CANopen® system

2.1 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 the areas of application for CANopen® have been extended to include other onboard control applications such as those for public transport, all-terrain vehicles, medical equipment, electronic applications in shipping, and building automation.

CANopen® has a linear structure, see Figure 2-1 on page 14.

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 “CAN in Automation" Group (CiA) is an international group of users and manufacturers, which ensures that the CANopen® protocol specifications are met and kept up-to-date.

The transmit/receive model used for the transmission of process data object (PDOs) enables broadcast transmission of information. In addition, every network device to be configured is allowed to listen to and to influence the PDO transmission of other devices. 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 enables powered network devices to be removed and replaced without interrupting the rest of the network.

2.2 Example topology of a CANopen® system

CANopen® has a linear bus topology, see Figure 2-1. 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).

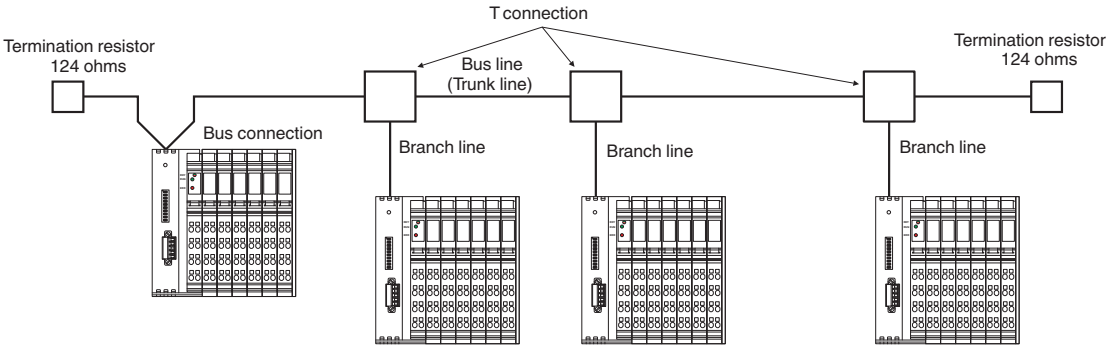


Figure 2-1 CANopen® example topology

The total length of trunk and branch lines depends on the data rate and the type of cable. Use Table 2-1 and Table 2-2 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.

Depending on the EMC requirements, trunk lines can be routed in different ways:

– In parallel
– Twisted
– In parallel and shielded
– Twisted and shielded

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 CO BK(-XC)-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 feed-in terminal must be used. The modules are supplied using a separate 24 V power supply.

Table 2-1 Maximum trunk line length vs number of devices

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

Table 2-2 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 a bus coupler stops communication to the connected modules 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 line

If there are no branch lines in the network, the trunk line is used to connect devices. If branch lines are present, T connectors will be linked via the trunk line (see Figure 2-1 on page 14). The maximum number of devices in the CANopen® network is limited to 127. The maximum length of the trunk line is between 40 m and 1 km, depending on the baud rate. The total cable length can be extended by using repeaters.

Branch line

The branch line connects the devices to the trunk line. They are connected using a T connector. Ensure that the branch lines are as short as possible, especially at higher baud rates. At 1 Mbps, 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 ohms 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.

2.3 Network considerations

2.3.1 Wiring

Figure 2-2 shows an example network structure with trunk and branch lines using T connections.

Termination resistor

A termination resistor must be installed at the extreme ends of the CANopen® system. Resistor properties: 124 Ω, 1% metal film and 0.25 W
(see CANopen® specification CiA 303-1), see Figure 2-2, detail D.

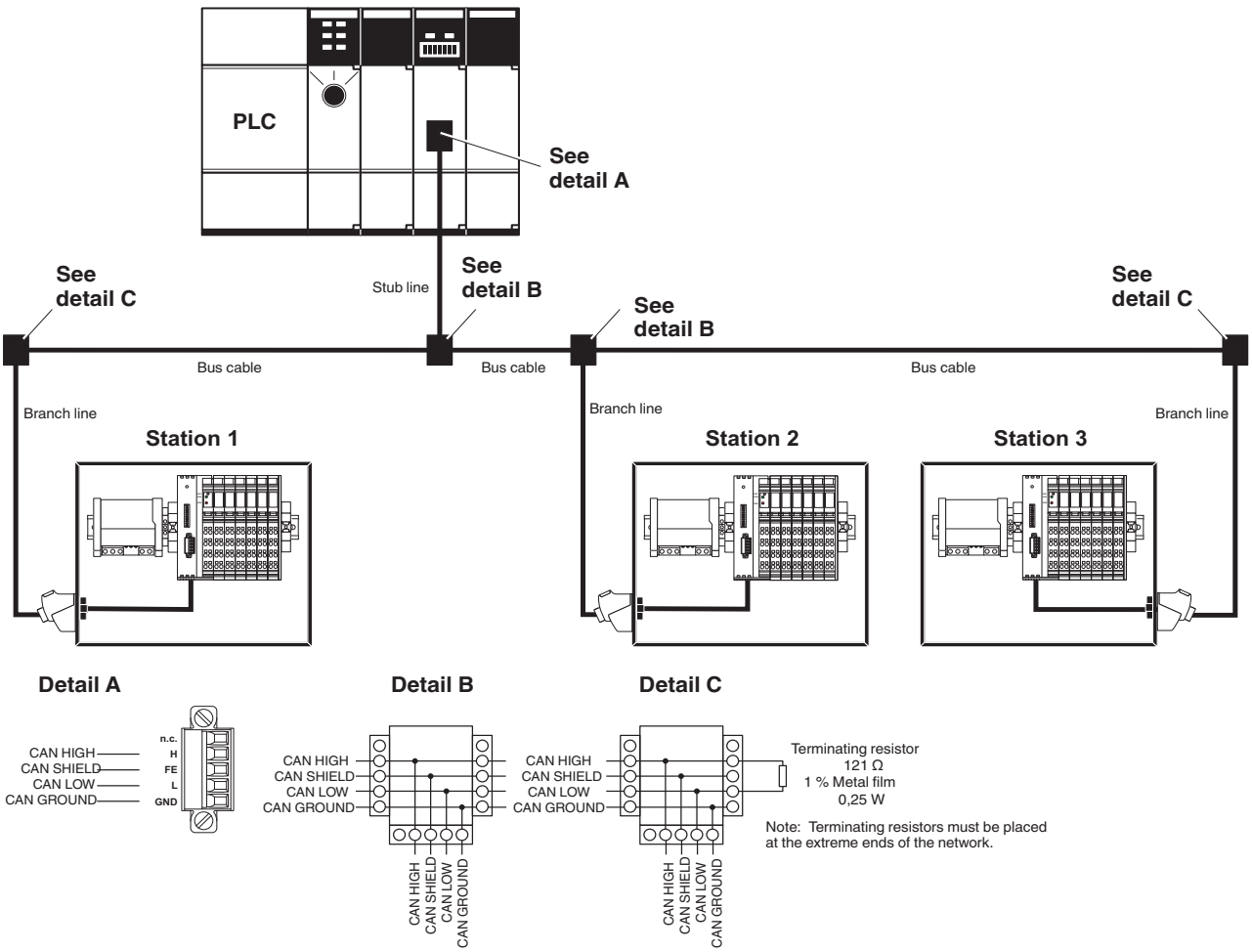


Figure 2-2 Example of setting up network communications and network power in an Inline CANopen® system

2.3.2 Central grounding for conductors and cables

WARNING:
Grounding protects people and machines against hazardous voltages. To avoid dangers, 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. Locate the power supply of the CANopen® network for all devices as close as possible to the physical center of the network. This will reduce the current in the ground line and decrease the overall ground voltage potential. Care must be taken that the ground potential in the network does not exceed 2 V DC. If this cannot be achieved, electrical isolation will be required.

Figure 2-3 on page 17 shows an example of a system grounding scheme using a single power supply. Please note that the power supply housing is directly connected to earth ground.

NOTE: Malfunction caused by interference to data telegrams
All CANopen® components must be grounded to avoid possible interference to data telegrams.

The Inline bus coupler is both an isolated physical layer device and an I/O device. Make sure to exactly observe the CANopen® grounding considerations during shielding (CAN shield). Connect the shield to the network connector labelled FE (see Figure 2-2 on page 16 and data sheet).

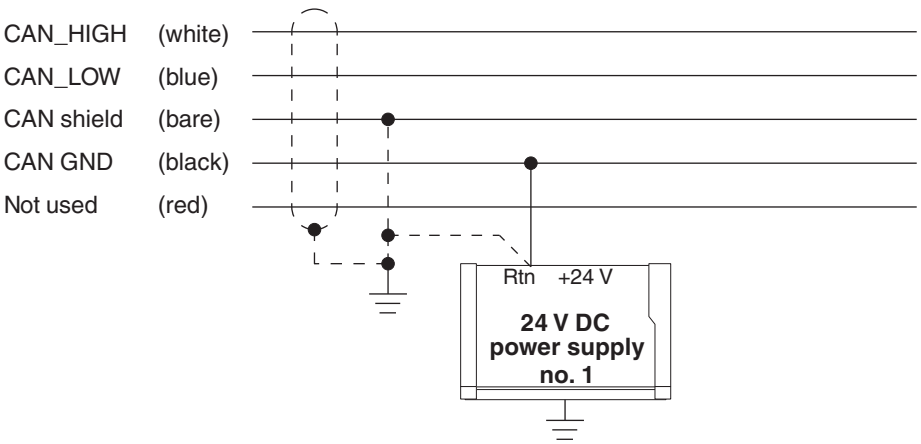


Figure 2-3 CANopen® system grounding scheme

3 Startup

3.1 Introduction

In this user manual, the terms “module” and “terminal” are used. They are basically the same.
Module is the general term.
In the Inline product group, modules are referred to as **terminals**.
When describing an Inline terminal, the term **terminal** is used.
When describing the CAN objects in general, the generic term **module** is used.

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

Knowledge of CANopen® the software used is required.
By default, data exchange via the bus coupler can be started as soon as the bus coupler receives the “Start remote node” command.

All DIP switches on the bus coupler are set to 0 upon delivery. This means that the following features are preset:

– Device address 0
  Device address 0 is reserved and is used to auto configure the I/O connected to the terminal.
– Transmission speed: auto baud rate
– Termination resistor: switched off
– Fault response mode 1
  (auto restart mode, Inline fault mode object, index 3002_{hex}).
  In auto restart mode all outputs will be turned off in the event of any local I/O error. An exception is a peripheral fault.

The PF bit will be set if an output is short circuited or if the supply to a power or segment terminal with fuse fails. The auto restart mode is described in more detail later in this section.

After the error has been resolved, the outputs are set to their predefined error state/substitute value (“0” by default) again.

By default, Inline station diagnostics includes the emergency object that is produced by the bus coupler during any error. The emergency object contains the PF bit. The PF bit needs to be evaluated by the application to determine the required action for the local station outputs.

3.2 Determining the maximum number of I/O terminals that can be connected

The bus coupler provides 32 IN and 32 OUT process data transmit and receive objects (PDOs). Independent of this, the bus coupler is capable to internally process 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 terminals 16
- Serial terminals 8
- Function terminals 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 0.8 A of communications power (U_L generated from U_{BK}).

If U_L requires a current higher than 0.8 A, the IB IL 24 PWR IN/R terminal can be used to reinject the U_L supply voltage.

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

3 Analog terminals cannot draw more than 0.5 A from the analog supply (U_{ANA}). Note that analog terminals also require current from the 0.8 A communications power (U_L).

The Inline communications power (U_L) provides a current of 0.8 A ant the analog supply (U_{ANA}) a current of 0.5 A. For further information on the current consumption of the I/O terminals connected to the bus coupler, please refer to the terminal-specific data sheet. The total current consumption of all terminals must not exceed the ratings stated above.

3.3 Configuring an Inline station

3.3.1 Bus coupler

Configuration of the bus coupler allows it to read in specific information about the I/O terminals connected on the local bus. Specific local bus information includes:

- Number of I/O terminals on the local bus
- Position of I/O terminals in the local bus
- Number of points or channels (subindexes) on each terminal
- I/O terminal types:
  - Digital input
  - Digital output
  - Analog Input
  - Analog output
  - PCP-capable (AS-i, RS-232, RS-485, other)
  - Serial (RS-232 or RS-485)
  - Function terminal (incremental encoder, absolute encoder, high-speed counter, etc.)

If a terminal cannot be identified as a digital, analog, or PCP terminal, it is assigned to the function terminals.

3.3.1.1 Configuration methods

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

After setting up or modifying an Inline CANopen® station, update the I/O configuration in the bus coupler memory.

Configure the bus coupler using one of the following auto configuration methods:

- Auto configuration by setting the DIP switches (no software required)
- Auto configuration by sending a service data object (SDO) message

Auto configuration by setting the DIP switches

When you use DIP switches 1 to 7 to set the device address 0, auto configuration is active. Auto configuration allows for the configuration of the bus coupler without any software support.

It features the following default settings:

Table 3-1 Default settings for auto configuration

PDO (process data object)		Contents
RPDO 1	Receive PDO 1	Digital outputs 1 ... 64
TPDO 1	Transmit PDO 1	Digital inputs 1 ... 64
RPDO 2	Receive PDO 2	Analog outputs 1 - 4
TPDO 2	Transmit PDO 2	Analog inputs 1 ... 4
RPDO 3	Receive PDO 3	Analog outputs 5 ... 8
TPDO 3	Transmit PDO 3	Analog inputs 5 ... 8
RPDO 4	Receive PDO 4	Analog outputs 9 ... 12
TPDO 4	Transmit PDO 4	Analog inputs 9 ... 12

RPDO	Receive process data object	Receive PDO
TPDO	Transmit process data object	Transmit PDO

All other I/O modules will not be mapped during configuration. This also includes function modules. Manually map all unmapped I/O modules to PDO 5 to PDO 32.

In CAN objects 3202hex (IN data COP index) and 3205hex (OUT data COP index), you can read out which index a module is mapped to.

For auto configuration of an installed station, proceed as follows:

- Disconnect power to the bus coupler.
- Set DIP switches 1 to 7 on the bus coupler, which are used to set the device address, to 0.
- Switch the supply voltages of the bus coupler (U_{BK}, U_M, and U_S) and of the connected I/O terminals on. If the RUN LED on the bus coupler is permanently on (green), the I/O configuration is stored in the station flash memory.
- Disconnect power to the bus coupler.

To switch to normal mode, proceed as follows:

- Set the desired switch setting for the device address, transmission speed, and termination resistor.
- Switch the supply voltages on.

The DIP switches are described in the data sheet.

Auto configuration by sending a service data object (SDO)

This auto configuration method is carried out by sending an SDO to the Reconfigure I/O object, index 3000_{hex}. If the Reconfigure I/O object, index 3000_{hex}, is set to “1”, the bus coupler requests the local bus and resets all the temporary variables of the active configuration. This configuration can be transferred to the non-volatile flash memory using 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.

SDO messages can be sent using software.

3.3.2 Analog input terminals (AI)

3.3.2.1 General configuration

To change the settings of an analog input channel, the corresponding process data output word must be accessed through the AIP 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 terminal.

Example configuration

This example explains reconfiguration of the first AI 2 terminal (channel 2) to the 10 V range as well as switching to the “IB ST” format (12 bits).

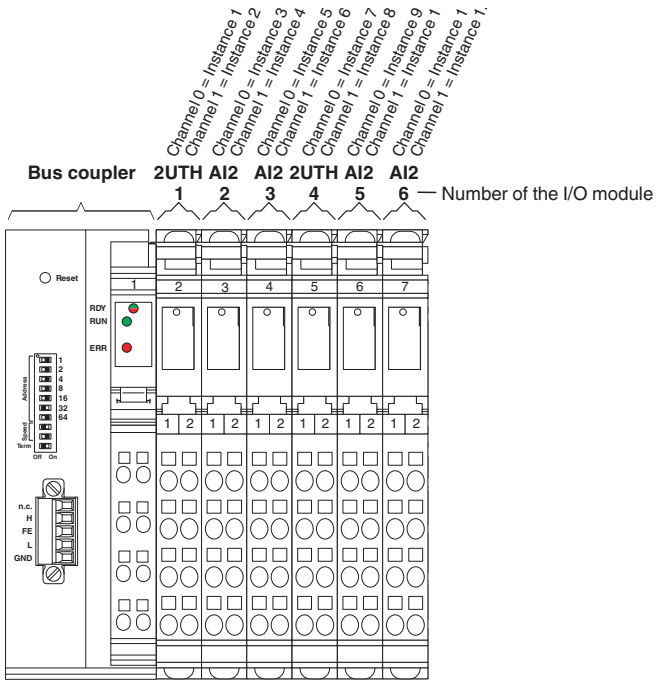


Figure 3-1 I/O station with analog input terminals and thermocouple terminals

The shown AI 2, UTH and RTD terminal are all 2-channel devices consisting of 16 bits for each channel. Channels for these terminals will occupy subindexes in the Read analog input ... object (index 6400_{hex}, 6401_{hex}).
Each channel has a corresponding output word. It is available in the AIP range object, index 2400_{hex}. In addition, the relevant channel can be configured.

1 Determine which subindex channel 2 of the first AI 2 terminal (second I/O module) resides in.
In the example, the second channel of the first AI 2 terminal will occupy subindex 4 of the Read analog input 16 bits object (6401_{hex}).

The UTH terminal occupies subindexes 1 and 2.
Channel 1 of the first AI 2 terminal occupies subindex 3.
If there are any difficulties locating the specific index for an object or subindex for the output word, an SDO can be sent to the following object with the following indexes:

Module number:

The position on the local bus is determined by counting the terminals from left to right starting with the bus coupler. It has module number 0.

(a) **OUT data COP index object, index 3205_{hex}**

The sent SDO is read to the subindex which is identified by the module number. This index will provide the number of the object index that the module's output data is mapped to.

(b) **OUT data COP first subindex object, index 3206_{hex}**

The sent SDO is read to the subindex which is identified by the module number. This index will provide the number of the first subindex that the module's output data is mapped to.

(c) **OUT data COP last subindex object, index 3207_{hex}**

The sent SDO is read to the subindex which is identified by the module number. This index will provide the number of the last subindex that a module's output data is mapped to.

If there are any difficulties locating the specific index or subindex for a specific input channel, an SDO can be sent to the following object with the following indexes:

(a) **IN data COP index object, index 3202_{hex}**

The sent SDO is read to the subindex which is identified by the module number. This index will provide the number of the object index that the module's data is mapped to.

(b) **IN data COP first subindex object, index 3203_{hex}**

The sent SDO is read to the subindex which is identified by the module number. 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}**

The sent SDO is read to the subindex which is identified by the module number. This index will provide the number of the last subindex that a module's input data is mapped to.

2 Refer to the terminal-specific data sheet to determine the value that need to be sent to the AIP object, index 2400_{hex}.
The AI terminal has the following default settings:

Range	(Bits 3 ... 0)	0 V ... 10 V	Bit code 0000
Format	(Bits 5 ... 4)	Inline, 15 bits	Bit code 00
Filter	(Bits 9 ... 8)	16-sample mean value	Bit code 00

Set the following:

Range	(Bits 3 ... 0)	Bipolar 10 V range	Bit code 0001
Format	(Bits 5 ... 4)	IB ST format	Bit code 01
	(Bits 7 ... 6)	Reserved	Bit code 00
Filter	(Bits 9 ... 8)	Default = 16-sample mean value	Bit code 00
	(Bits 14 ... 10)	Reserved	Bit code 00000
	(Bit 15)	Data for reconfiguration	Bit code 1

The value to be sent is 8011_{hex}.

3 Send value 8011_{hex} to the AIP range object, index 2400_{hex}, subindex 4 using a service data object (SDO) message. The message can also be sent using a configuration software. Once the new value range has been configured, the settings are stored in the non-volatile flash memory of the bus coupler.

For additional information on the AIP range object, index 2400hex, please refer to “Appendix: CANopen® objects and indexes” on page 51.

3.3.3 Thermocouple and RTD terminals

3.3.3.1 General configuration

This section describes how to change the default settings for a 2-channel Inline thermocouple terminal. Changing the default settings for the 2-channel RTD terminal is accomplished in the same manner.

There is no “Thermocouple” or “RTD” object. The default settings are changed using the AIP object, index 2400_{hex}. For more information refer to “Appendix: CANopen® objects and indexes” on page 51 and the information in this section.

Default settings for the thermocouple are:

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

Refer to the terminal-specific data sheet for information to appropriate attribute settings for the AIP object, index 2400_{hex}. It can be used to change the default settings.

Keep in mind that thermocouple or RTD subindexes will appear as analog subindexes. There are two subindexes for every thermocouple or RTD terminal.

Make sure to know which subindexes are analog inputs and which subindexes are thermocouple terminals or RTD terminals (not shown in Figure 3-1 on page 24).
The values are mapped to the Read analog input 16 bits object, index 6401_{hex}.
The subindexes are assigned as follows:

Table 3-2 Read analog input 16 bits object, index 6401_{hex}

Subindex	Terminal in Figure 3-1 on page 24	
	Number of the Inline I/O terminal	Type
1, 2	1	2UTH
3, 4	2	AI2
5, 6	3	AI2
7, 8	4	2UTH
9, 10	5	AI2
11, 12	6	AI2

To change the standard settings, send a service data object (SDO) message to a specific subindex.

Once the new thermocouple terminal or RTD terminal settings are made, the new configuration is stored in flash memory of the bus coupler.

For a description of the AIP range object, index 2400hex, please refer to “Appendix: CANopen® objects and indexes” on page 51.

3.3.4 Function terminals

3.3.4.1 General configuration

Function terminals include incremental encoders, absolute encoders, high-speed counters, etc. Function terminals can be configured using the service data object (SDO) data channel by default. The terminals are parameterized using output words which are assigned to the function terminals. Refer to the terminal-specific data sheet or user manual to determine the codes that need to be written to the relevant output word.

The input and output data of the function terminals can be mapped to any process data object (PDO). By default, this data is not mapped.

Parameterization of a function terminal cannot be stored in the flash memory of the bus coupler. A corresponding subroutine must therefore be implemented in the application.

3.3.5 Analog output terminals (AO)

3.3.5.1 General configuration

Some analog output terminals support parameters that can be edited. This includes the IB IL AO 2/U/BP-PAC terminal, for example. Each command that is sent to the analog output terminals for parameterization reasons must be verified. To do so, the AOP response data object, index 2410_{hex}, needs to be accessed.

The IB IL AO 2/U/BP-PAC terminal requires a sequence of commands to be sent to the relevant subindex containing the output word for the AO 2. These output words reside in the 16-bit Write analog output 16 bits object, index 6411_{hex}.

After each command in the sequence is sent, it must be checked whether a positive or negative response to the command is returned in the relevant subindex of the AOP response data object. Once the sequence is completed and all responses have been verified positive, the terminal is ready to receive normal analog output data via the 16-bit Write analog output object.

Only analog output terminals that can be configured and have associated input words, e.g., IB IL AO 2/U/BP-PAC, will occupy subindexes in the AOP response data object, index 2410_{hex}. Each channel of the terminal occupies one subindex in the object. The subindexes are assigned according to the position on the local bus. The terminal located most closely to the bus coupler is assigned the first subindex.
Analog output terminals that do not require configuration will not occupy subindexes in the object. For information on parameterization, please refer to the terminal-specific data sheet.

3.4 Mapping the modules to the process data objects (PDOs)

For a detailed explanation of objects, please refer to “Appendix: CANopen® objects and indexes” on page 51.

3.4.1 Bus coupler

The bus coupler automatically maps the I/O modules to the process data objects (PDOs) according to the implemented CANopen® specification. This procedure is explained in Section “Configuring an Inline station” on page 21. The PDOs can be mapped to a PLC or another device using Visual Object Linker.

If the configuration software uses the EDS file, this file is the basis for the memory mapping of the master to match the automatic mapping of the bus coupler.

3.4.2 Electronic Data Sheet (EDS) file

The EDS (Electronic Data Sheet) file is the software interface between the bus coupler and a CANopen® configuration software. The EDS file describes the object dictionary entries supported by the module and makes them available to the user in a readable format using the software.

If the I/O modules are to be included in the process data objects, the Reconfigure I/O object, index 3000_{hex}, can be used for newly configured or changed configurations.

If the Reconfigure I/O object, index 3000_{hex}, is set to “1”, the bus coupler requests the local bus and resets all the temporary variables of the active configuration. This configuration can be transferred to the non-volatile flash memory using a save command to the Store parameters object, index 1010_{hex}. The configuration will remain valid until the next Reconfigure I/O command is sent or the bus coupler is reconfigured in another way.

3.5 I/O data transfer

I/O data transfer can be accomplished by setting the bus coupler to the operational state. This is done by sending “Start Remote Node” command to the bus coupler. Once this is accomplished, the bus coupler 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.

3.5.1 Communication

The bus coupler communicates via CANopen® using different communication objects. The following types are available:

– 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 via the following CANopen® communication objects:

– 32 receive PDOs
– 32 transmit PDOs
– 2 server SDOs

3.5.1.1 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.

The PDOs can be activated in three ways:

– Event driven
– Time controlled
– Remote request

By default, the transmit PDOs of the bus coupler are asynchronous and are activated by changing the inputs.

When a digital input changes, PDO1 is transmitted.

The analog input interrupts are disabled by default. Therefore, the user will either have to enable the interrupt method or set the event timer to allow the default analog input TPDOs (2 to 4) to be transmitted. Analog input interrupts or the event timer will trigger any of the 1 to 32 TPDOs that have analog inputs mapped.

3.5.1.2 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 devices using SDOs.

3.5.1.3 Synchronization objects

The bus coupler provides one synchronization object. This object enables data synchronization when data is sampled and transmitted from each device on the network. A time interval can be programmed for each device. The device waits for this time interval after receiving the SYNC object and before sending the PDO.

3.5.1.4 Emergency objects

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

3.5.2 Standard objects for digital, analog, and special function modules

For a detailed explanation of objects, please refer to “Appendix: CANopen® objects and indexes” on page 51.

3.5.2.1 Objects for digital input modules, index 6000_{hex} ... $61FF_{hex}$

Objects 6000_{hex} ... $61FF_{hex}$ provide access to digital input data and the 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 digital input point available on the device.

Object 6000_{hex}, for example, 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. Subindex 1 is used to read the states of inputs 1 to 8, subindex 2 is used to read the states of inputs 9 to 16 and so on. For additional information see Appendix A.

Setting inputs to latch

Each digital input point can be independently parameterized to latch on a desired state. This is done using the DIP latch enable, index 2000_{hex}, and DIP latch state, index 2001_{hex}, objects. Figure 3-2 shows how “actual input” or “latched” data is selected. Enabling the latch and the desired latch state must be done by sending an SDO.

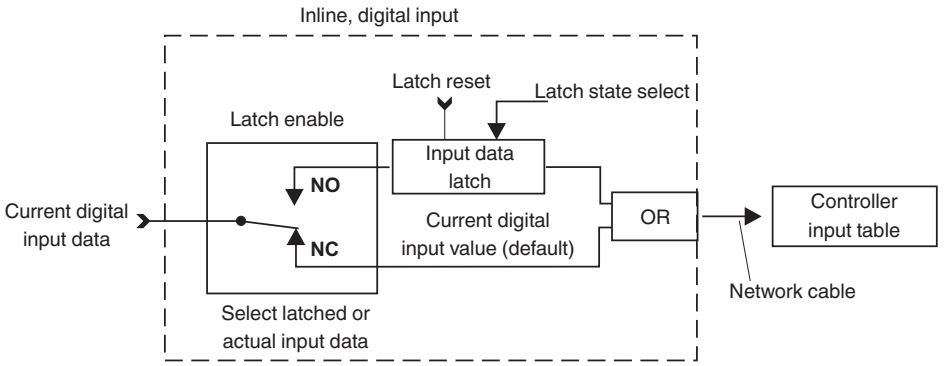


Figure 3-2 Latched or actual digital input data

The DIP latch enable object, index 2000_{hex}, accesses groups of 8 inputs.

The following applies to each bit in the relevant subindex:

– If the bit is set to logic “1”, the latch function is enabled.
– If the bit is set to logic “0”, the latch function is disabled.

Index 2000_{hex} is set to “0” by default and thus disables the latch function.

If the DIP latch enable object has been enabled by setting it to “1”, the desired latch level can be selected. This level can be set in the Digital latch State object, index 2001_{hex}, of the specific subindex. If DIP Latch State is set to “0”, the subindex also latches logic “0”. If DIP Latch State is set to “1” (by default), the input instance also latches logic “1”.

Resetting latched inputs

Setting bit 1 in the Inline interface control byte object, index 3111_{hex}, resets the latched inputs. 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 0 to allow for the next latched condition to occur when the control byte has been mapped to a PDO.

By default the Inline control byte is not included in a PDO. The control byte can be mapped to a PDO if required.

Sending an SDO message to the Inline interface control byte object, index 3111_{hex}, prevents that the latch is cleared by the PDOs. Setting the Inline Interface Control Byte to value “2” will reset all latches and enable the next input latch. It will also reset the object value automatically to “0”.

The actual value of the “latch” is stored in the bus coupler memory. The latch state is “1” by default. This means that the default latch value in the bus coupler memory is “0”. If the latch state is set to “0”, the internal latch value must be re-initialized using one of the following two methods.
Both of these methods will initialize the internal latch value required for the specific latch state (determined by the latch value) and prepare the latching function. It becomes active as soon as latching has been enabled.

1 Store the configuration settings. Restart the device.

2 Reset the latches using index 3111_{hex}, control byte bit 1.

Objects for digital output modules, index 6200_{hex} ... $63FF_{hex}$

Objects 6200_{hex} ... $63FF_{hex}$ provide access to digital output data and the 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 digital output point available on the device.

Object 6200_{hex}, for example, 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 to subindex 1 will set the output states of outputs 1 to 8. Writing to subindex 2 will set the output states of outputs 9 to 16. For further information see Appendix A.

Objects for analog input modules, index 6400_{hex} ... 6401_{hex}

The objects for analog input modules are used to map digital inputs on the CANopen® bus coupler. There is a separate subindex for each analog input point 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 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 and configuring a trigger interrupt, then setting the Analog input global interrupt object, index 6423_{hex}. Keep in mind that an adjustment of the TPDO inhibit time may require the network traffic to be reduced.

(1) Setting a trigger interrupt type

Select the trigger interrupt type. To do so, use 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 following options are available to do so:

– 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}

Objects for analog output modules, index 6410_{hex} ... 6411_{hex}

The objects for analog output modules are used to map analog outputs on the CANopen® bus coupler. There is a separate subindex for each analog output point 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 substitute values. The error states and substitute values can be used to control the analog output during a network or station fault. Each channel can be parameterized separately to react in one of the two ways listed below. For information on how to select the fault response modes, please refer to Section “Behavior in the event of an error” on page 35. For additional information on the error states and substitute values of the analog outputs see Section “Analog output configuration objects” on page 201.

1 Hold the last value.

2 Set to the value which is defined by the Analog output error value integer object (object 6444_{hex}).

Inline special function object, index 3300_{hex} ... $330D_{hex}$

The special function objects allow for controlling and monitoring the below listed terminals and any other terminal that cannot be mapped to a standard CANopen® object.

– Incremental encoder
– Absolute encoder
– High-speed counter
– ...

Objects for PCP modules, index 3400_{hex} ... 3411_{hex}

The PCP module objects allow for sending data to and receiving data from modules that have PCP capability. There is a separate subindex for each module connected to the bus coupler. The CANopen® bus coupler supports a total of 16 PCP modules.

Objects for serial modules, index 3400_{hex} ... 3411_{hex}

The objects for serial modules provide special access to serial modules such as the RS-232 and RS-485 modules. There is a separate instance for each serial module connected to the bus coupler.

3.6 Behavior in the event of an error

3.6.1 Objects for controlling the output behavior in the event an error

Objects 1029_{hex} and 3002_{hex} ... $300A_{hex}$ can be used to control the output behavior in the event of different errors.

Table 3-3 Objects controlling the output behavior in the event of an error

Object	Name	Description
1029	Error behavior	The object controls what mode the module will be put in during a communication error. Some communication errors, for example, are “bus off” conditions: – PDO reception with less data that expected – CAN buffer overflow – Timeout of a node guard/heartbeat
3003 ... 300A	... fault ... mode	The objects determine the CANopen® state. This depends on the specific error types.
3002	Inline fault mode	The object is used to determine how the bus coupler will control the outputs when specific errors occur. There are four modes available for controlling the outputs. Each mode is defined below (see Section 3.6.2.1 to Section 3.6.2.4).

3.6.2 Configuring the behavior in the event of an error

The behavior in the event of an error is a configurable setting in the Inline fault mode object, index 3002_{hex}. This setting will control the behavior of the Inline station in the event that an error occurs that is stored in the three following objects.

All the error bits of the Inline Stations Status are described in Section 4, “Diagnostics”.

1 **Inline Module Change Fault Mode, index 3006_{hex}.**
Represented in bit 3 of the low byte of the Station status object, index 3101_{hex}.
The status of this error is also be represented in byte 3, bit 3 of the emergency telegram.

2 **Inactive local bus fault mode object, index 3007_{hex}.**
Represented in bit 4 of the low byte of the Station status object, index 3101_{hex}.
The status of this error is also be represented in byte 3, bit 4 of the emergency telegram.

3 **Inline connection fault mode object, index 3008_{hex}**
Represented in bit 5 of the low byte of the Station status object, index 3101_{hex}.
The status of this error is also be represented in byte 3, bit 5 of the emergency telegram.

3.6.2.1 Inline fault mode 0: Stop running data cycles on fault

If a local error occurs, the outputs are automatically switched off.

Once the fault has been resolved, one of the following measures must be taken to switch on the device again:

– Restart the local bus (index 3111_{hex}, subindex 0, bit 2)
– Send the Reset Node command
– Set CANopen® to the operational/pre-operational state

3.6.2.2 Inline fault mode 1: Auto restart (default)

If a local error occurs, the outputs are automatically switched off.

The bus coupler is in the mode defined by the ... fault ... mode (3003_{hex} to 3009_{hex}) objects. Once the fault has been removed the outputs can be controlled.

If the bus coupler is not able to establish local bus communication again, it can be established by the user. The following options are available:

– Change the state from Stop to Pre-Operational and from Pre-Operation to Operational.
  Or
– Send a Restart Bus command (object 3111_{hex}, bit 2).
  Or
– Send a Reset Node command.

If local bus communication can still not be established, switch the module off and check all the module connections.

3.6.2.3 Inline fault mode 2: Go to fault state

The outputs are turned off automatically for up to two seconds when a local error 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. Reconfiguration (object 3000_{hex}) or a restart (power up of UBK) must be performed. The bus coupler continues to transmit predefined substitute values even if the state is returned to operational. The module continues to transmit predefined substitute values (the Fault Cycles bit is set in the Station status object, index 3101_{hex}) until a command is sent to restart the local bus. Once the local bus is restarted, new I/O data will update the outputs.

3.6.2.4 Inline fault mode 3: Continue on fault

The outputs are turned off automatically for up to two seconds when a local error occurs. After two seconds, the station will perform an update for all remaining active I/O modules.

Output data used is defined by Fault mode objects, indexes 3002_{hex} to $300A_{hex}$. Output data can either be substitute values or process data. The bus coupler continues to run in this mode until a command is sent to restart the local bus. The station will not recover its lost I/O even after the fault condition has been cleared. Reconfiguration or a restart (power up of U_{BK}) must be performed.

3.6.3 Changing the fault response mode

Fault mode is a setting in the flash memory of the bus coupler. Fault mode cannot be changed by means of reconfiguration of the I/O station or a restart (power up of U_{BK}). In contrast to mode 1, fault mode can be changed by sending an SDO to the Fault mode object, index 3002_{hex} (see “Appendix: CANopen® objects and indexes” on page 51).

3.6.4 ... fault ... mode objects, indexes 3003_{hex} to $300A_{hex}$

Objects 3003_{hex} to 3009_{hex} allow the user to control the state of the bus coupler in the event of a local bus fault. Fault control is activated when the corresponding bit is set in the Inline Status. Inline Status is part of the Station status object, index 3101_{hex}. If any fault mode causes the state of the bus coupler to change, the predefined substitute values are output and the Fault Cycles bit is set in the Station status object, index 3101_{hex}.

Fault modes can only be activated in the operational state. 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 Inline peripheral fault mode object, index 3004_{hex}) will set the module to the pre-operational state. Peripheral fault mode specifies “no change of state”.

To return to the operational state from a state change, as defined by object indexes 3003_{hex} to 3009_{hex}, a “start remote node” command can be sent. 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 transmitted instead of substitute values.

If any module causes the bus coupler to go to the stop state and that fault needs special measures to be cleared (such as a module that has a latched PF), it may not be possible to set the bus coupler back to the operational state. In this case, proceed as follows:

- Remove the cause of the latched peripheral fault.
- Delete the latched peripheral fault in the pre-operational state. Use the Interface control byte object (index 3111_{hex}) for this.

3.6.4.1 Object 3003_{hex}: Inline CRC fault mode

This object specifies which state the bus coupler is set to when an Inline CRC error occurs. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3003_{hex:\ Inline\ CRC\ fault\ mode}$” on page 103.

3.6.4.2 Object 3004_{hex}: Inline peripheral fault mode

This object specifies which state the module is set to when a peripheral Inline fault (PF) occurs. The bus coupler must be in the operational state. An example of a PF is a short circuited output.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3005_{hex:\ Inline\ power\ fault\ mode}$” on page 105.

3.6.4.3 Object 3005_{hex}: Inline power fault mode

This object specifies which state the bus coupler is set to when an Inline power fault occurs. The bus coupler must be in the operational state. The fault mode becomes active at a voltage of less than approximately 18 V. Power faults may occur with U_{BK}, U_S, U_M or the processor supply.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3005_{hex:\ Inline\ power\ fault\ mode}$” on page 105.

3.6.4.4 Object 3006_{hex}: Module change fault mode

This object determines whether the bus coupler is set to the operational state in the following cases:

– The active local bus configuration differs from the saved configuration (referred to as “module change” in the following)
– The bus coupler receives the “Start Remote Node” command.

0 = Bus coupler in the pre-operational state

1 = Allow the operational state

For detailed information see “Object $3006_{hex:\ Inline\ module\ change\ fault\ mode}$” on page 106.

3.6.4.5 Object 3007_{hex}: Inactive local bus fault mode

This object specifies which state the bus coupler is set to when the local bus is inactive and not running data cycles. The bus coupler must be in the operational state.

Object 3007_{hex} is used in the following cases:

– A connection of module change fault occurs.
– The bus coupler initializes the local bus.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3007_{hex:\ Inactive\ local\ bus\ fault\ mode}$” on page 107.

3.6.4.6 Object 3008_{hex}: Inline connection fault mode

This object specifies which state the bus coupler is set to when an Inline connection fault occurs. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3008_{hex:\ Inline\ connection\ fault\ mode}$” on page 108.

3.6.4.7 Object 3009_{hex}: Inline faulted cycles mode

This object specifies which state the bus coupler is set to when the bus coupler is running data cycles with substitute values instead of process data. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

For detailed information see “Object $3009_{hex:\ Inline\ faulted\ cycles\ mode}$” on page 109.

3.6.4.8 Object $300A_{hex}$: Processor power fault mode

This object defines whether the bus coupler communicates with the modules on the local bus or not if the U_{BK} supply falls below 11 V.

The communications power (U_L), for example, is generated from the U_{BK} supply. Therefore, U_{BK} is required for the entire communications supply on the local bus.

0 = Keep running cycles (outputs remain in current state)

1 = Reset local bus (default): This sets all the modules to their default output state. For the default values, please refer to the terminal-specific data sheets.

For detailed information see “Object $300A_{hex:\ Processor\ power\ fault\ mode}$” on page 110.

4 Diagnostics

4.1 Diagnostics and status indicators

The bus coupler and the InlineI/O terminals are provided with diagnostics and status indicators which are used for quick local error diagnostics.

A detailed description of the diagnostics and status indicators can be found in the data sheet for the bus coupler and in the terminal-specific data sheets.

4.2 Alarm message and diagnostic objects

For a detailed explanation of objects, please refer to “Appendix: CANopen® objects and indexes” on page 51.

Diagnostic information is provided via alarm messages and different diagnostic objects. An alarm message (emergency message) is generated automatically. An SDO, however, must be sent by the user to the object indexes described in this section.

4.2.1 Alarm message

The CANopen® bus coupler generates an emergency message in the event of an error. This alarm message contains the Inline status and the number of the faulty module along with other useful information.

The alarm message consists of the 8 bytes.

Emergency object telegram structure:

Byte number	0	1	2	3	4	5	6	7
Meaning	Alarm error code		Error register object, index 1001_{hex}	Station status object, index 3101_{hex}, low byte	Station status object, index 3101_{hex}, high byte	Inline faulted module object, index 3102_{hex}	Not relevant	Not relevant

The alarm message is additionally explained in the following sections:

– “Emergency error codes” on page 53
– “Emergency object data” on page 53
– “Appendix: CANopen® objects and indexes” on page 51

4.2.2 Diagnostic information

In addition to the alarm messages diagnostic information is provided via different service data objects (SDOs). They allow for status analysis of the Inline station. The corresponding SDOs are listed in following table.

Table 4-1 Objects

Index (hex)	Name	Function
1001	Error register	This object index can be used to receive the manufacturer-specific bit 7 using an SDO. This bit indicates an error in the Inline station. You can use this bit or an alarm message to retrieve further diagnostic information.
1002	Manufacturer status register	Manufacturer-specific status of the Inline station and the number of the faulty Inline module. This object index is further defined in Appendix A.
1003	Pre-defined error field	Up to the last ten errors (subindexes). Subindex 0: total number of available errors in the stack. This object index is further defined in Appendix A.
3101	Station status	Current status of the Inline station.
3102	Inline faulted module	Number of the currently faulty Inline module.
310C	Inline latched fault	Latches and indicates the last Inline station status (low byte). The same bit meaning as with Station status object, index 3101_{hex}, except that the bits are latched. This object index is further defined in Appendix A.
310D	Inline latched faulted module	Latches and indicates the number of the last faulty Inline module. The same bit meaning as with Station status object, index 3101_{hex}, except that the bits are latched. This object index is further defined in “Appendix: CANopen® objects and indexes” on page 51.

In the operational state, a local error in the Inline station always generates an alarm message. By default, due to the settings of the error bits (explained in this section), predefined substitute values (zero by default) are sent to the local outputs. It is possible that error information may be overwritten due to intermittent error conditions. In this case, it is recommended to access the Pre-defined error field object, index 1003_{hex}. Each subindex in the object, index 1003_{hex}, represents a stored fault sequence.

4.2.3 Station status object, index 3101_{hex}

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

Bit meanings for the 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 error 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 Retry max parameter. If there is no CRC transmission of data even after Retry max has expired, the CRC error bit is set. By default in CANopen®, the bus coupler is set to the pre-operational state in the event of a CRC error. The Inline CRC fault mode object, index 3003_{hex}, is used to control this default setting.

Low Byte, bit 1: Peripheral fault

The Peripheral fault bit will be set if an output is short circuited or if the supply to a power or segment terminal with diagnostics fails. By default in CANopen®, there is no change of state in the event of a peripheral fault. The Inline peripheral fault mode object, index 3004_{hex}, is used to control this default setting.

Low byte, bit 2: Power fault

The Power fault bit will be set when any of the power supplies (U_L, U_S, U_M) is in an overvoltage or undervoltage condition. By default in CANopen®, the bus coupler is set to the pre-operational state in the event of a supply error. The Inline power fault mode object, index 3005_{hex}, is used to control this default setting.

Low byte, bit 3: Module change fault

The Module change fault bit will be set when the active configuration on the Inline local bus does not match the configuration that was stored in the flash memory during the last configuration cycle. By default in CANopen®, the bus coupler is set to the pre-operational state or remains in this state in the event of a module change error. The Module change fault mode object, index 3006_{hex}, is used to control this default setting.

Low byte, bit 4: Inactive local bus fault

The Inactive local bus fault bit will be set when no data cycles are run on the local bus. When this condition occurs the bus coupler is in the turned off state. Possible causes include:

– The bus coupler is not able to communicate with the first I/O module connected to it. Possible errors include a bus coupler error or an error of the first I/O module.
– The bus coupler will try to auto-recover the local bus. An error occurred and the bus coupler is in the waiting state until the error has been removed and the local bus can be restarted again.

The default CANopen® action in the event of an inactive local bus is controlled by the Inactive local bus fault mode object, index 3007_{hex}. By default, the bus coupler is set to the pre-operational 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 communicate with the connected modules and to determine the faulty position. This connection fault occurs due to a broken data path on the local bus.
Two objects indicate the position of the modules between which the fault occurred. Read the Inline first faulted module object, index $310A_{hex}$, and the Inline last faulted module object, index $310B_{hex}$, to locate the fault. The CANopen® state action generated by bit 5 is determined by the Inline connection fault mode object, index 3008_{hex}. By default, the bus coupler is set to the pre-operational state.

Low byte, bit 6:
Inline faulted cycles mode

The Inline faulted cycles mode bit informs the application that the local outputs were set to the preprogrammed CANopen® error state and will no longer communicate with the controller.

Faulty data cycles are run in the following cases:

– An Inline station status error causes a change of state of the CANopen® interface.

Or

– The Inline fault mode object, index 3002_{hex}, is set to mode 2, and an error occurs.

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 deleted by means of the following:

– Write the “SAVE” ASCII string to the Store parameters object, index 1010_{hex}.

Or

– Perform these steps:
  – Set all DIP address switches to “0”.
  – Restart bus coupler.
  – Set the DIP address switches to the desired address.
  – Restart bus coupler.

4.2.4 Inline faulted module object, index 3102_{hex}

This object contains the number of the module that has faulted. The numbering of the module starts at the bus coupler with 0 and ends with 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 alarm message.

By default, these faults are considered major faults (except for a peripheral fault) and the red RUN LED on the bus coupler is affected, see Figure 4-1.

4.2.5 Examples of requesting diagnostic information

The following describes examples of requesting diagnostic information. 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).

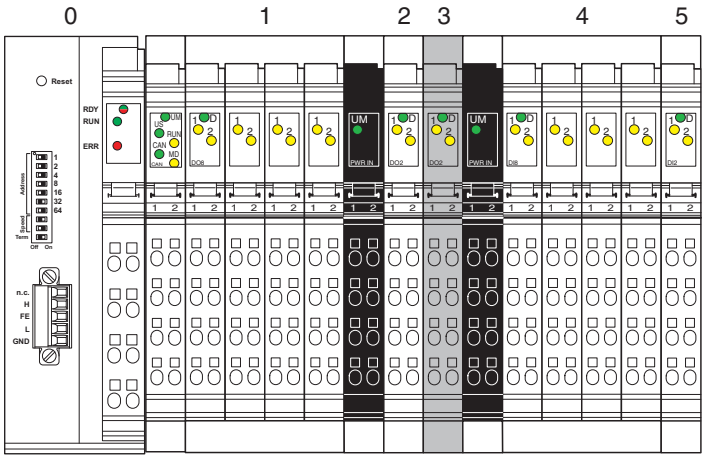


Figure 4-1 Example station

4.2.5.1 Example A: Peripheral fault, short circuit at module no. 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 the fault code and location of errors using the emergency object

The user can evaluate byte 3 and byte 5 of the emergency object to request the fault codes and the location of errors on the Inline station.

1 Read byte 3, Inline station status. This value represents the fault code. In this example the output short circuit will cause the Peripheral fault bit (bit 1) to be set and value “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 this example, byte 5 reads value 3. This indicates that an error occurred on module number 3. Keep in mind that numbering on a CANopen® Inline station starts with 0 (bus coupler) and continues right to number 63.

B. Receiving the fault code and location of errors using SDOs

Errors can also be evaluated by sending a service data object (SDO) message to the Station status object, index 3101_{hex}, and to the 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 Station status object, index 3101_{hex}. This value represents the fault code. In this 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. In this example, the value is 3. Keep in mind that numbering on a CANopen® Inline station starts with 0 (bus coupler) and continues right to number 63.

4.2.5.2 Example B: Bus error, communications break between modules 2 and 3

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

A. Receiving the fault code and location of errors using the emergency object

The user can evaluate byte 3 and byte 5 of the emergency object to request the fault code and the location of errors on the Inline station.

1 Read byte 3, Inline station status. This value represents the fault code. In this 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 communication path (DI or DO) is broken.

2 Read byte 5, Inline faulted module number This value represents the location of the break between modules 2 and 3. There are two possible fault conditions in this example: module change fault or Inline connection fault.

Module change fault

In the event of a module change fault, byte 5 contains value 3. The bus coupler may consider this fault a missing module number 3.

Inline connection fault

In the event of a connection fault, byte 5 contains value 3. The bus coupler may consider this fault a connection fault between modules 2 and 3. In this example, the value of the Inline faulted module byte is 0 (representing 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 faulty connection. The location would be a value of either 2 or 3 depending on which communications line is broken.

Send an SDO message to the Inline last faulted module object, index $310B_{hex}$, to receive a value indicating the location of the other end of the faulty connection. The location would be a value of either 3 or 2 depending on which communications line is broken. Keep in mind that numbering on a CANopen® Inline station starts with 0 (bus coupler) and continues right to number 63.

B. Receiving the fault code and location of errors using SDOs

Errors can also be evaluated by sending a service data object (SDO) message to the Station status object, index 3101_{hex}, and to the 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 Station status object, index 3101_{hex}. This value represents the fault code. In this 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 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. There are two possible fault conditions in this example: module change fault or Inline connection fault.

Module change fault

In the event of a module change fault, value 3 is shown. The bus coupler may consider this fault a missing module number 3.

Inline connection fault

In the event of an Inline connection fault, the location of the fault is between modules 2 and 3. In this case, the Inline faulted module object, index 3102_{hex}, indicates the number of the module that is located directly before the location of the fault. 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 faulty connection. The location would be a value of either 2 or 3 depending on which communications line is broken.

Send an SDO message to the Inline last faulted module object, index $310B_{hex}$, to receive a value indicating the location of the other end of the faulty connection. The location would be a value of either 3 or 2 depending on which communications line is broken. Keep in mind that numbering on a CANopen® Inline station starts with 0 (bus coupler) and continues right to number 63.

4.2.6 Fault modes

By default, all the fault bits of the Inline station status byte (byte 0, bit 0, bits 2 to 6) except for bit 1 are considered major faults. By default, the fault bus coupler are set to the pre-operational state. This will stop all PDO activity and an emergency message will be sent to the master. The default values can be changed by using an EDS file or sending an SDO to the corresponding ... fault ... mode object, indexes 3003_{hex} ... 3009_{hex}.
To do so, the bus coupler must be in the operational state.

Possible fault settings are:

0 = Pre-operational (default)

1 = No state change

2 = Stopped

4.2.7 Latched diagnostic data

The bus coupler will latch the last occurring Inline status fault, the module number and errors at the connection points. In this way, every intermittent fault that occurs too quickly to be updated by the CANopen® master will be detected. Writing value 0 to the Inline latched fault object, index $310C_{hex}$, subindex 0, deletes the latched values. The following latched diagnostic data is available through the EDS file or by sending an SDO to the corresponding Inline interface object, indexes $310C_{hex}$ to $310F_{hex}$.

– **Inline latched fault**, index $310C_{hex}$
  This object contains the last reported fault of an Inline station. The individual bits indicate the same faults as described in the Inline station status byte, see “Station status object, index 3101_{hex}” on page 43.
– **Inline latched faulted module**, index $310D_{hex}$
  This object contains the position of the first faulty module that was reported to the Inline station during the last fault.
– **Inline latched first faulted module**, index $310E_{hex}$
  This object contains the latched number of the first module before the fault location that was latched during the last connection fault.
– **Inline latched last faulted module**, index $310F_{hex}$
  This object contains the number of the last module after the fault location that was latched during the last connection fault.

4.2.8 Inline interface control byte

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

For information on how to latch digital input states, please refer to Section “I/O data transfer” on page 29.

By default the Inline interface control byte is not included in a PDO. It can be added to the receive PDO by mapping the Inline control byte object, index 3111_{hex}, subindex 0. This byte can also be accessed via a service data object. To do so, send an upload or download to the Inline control byte object, index 3111_{hex}, subindex 0. If accessed using an SDO, this bit will be deleted automatically.

– Bit 0 = 1: Delete all latched peripheral faults.
– Bit 1 = 1: Delete all input states.
– Bit 2 = 1: Start the local bus.

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

4.2.9 Error states

In the case of digital and analog outputs, the bus coupler supports standard CANopen® error modes and substitute values. The error modes and substitute values can be set and read by means of service data objects (SDOs). Error modes will only occur during a network error They will not occur in the event of a local Inline error unless the Inline fault mode object, index 3002_{hex} is set to a 2. The default setting for the substitute value is 0.

Digital output support

– Hold last state
– Turn off during a faulted condition (default)
– Turn on during a faulted condition

Analog output support

– Hold last value
– Set to the value which is defined by object 6444_{hex} (analog output error value)

"Appendix: CANopen® objects and indexes" on page 51 will detail the indexes and objects of the digital outputs and analog error modes and substitute values.

4.2.10 Error codes of analog inputs, thermocouples and RTD modules

Analog inputs, thermocouples and RTD modules are able to report diagnostic codes. These codes must be read from the PDO data or through an SDO message sent to the 16-bit Read analog input object, index 6401_{hex}, see Appendix A. A list of these codes in the "IL" Inline format can be found in Table 4-2.

Error codes are dependent on the type of module and the configuration of the module format. By default, error codes are received in the "IL" Inline format and can be listed as shown in Table 4-2. In the event of format changes, the error code can be determined using the terminal-specific data sheet.

Table 4-2 Error messages of analog input modules

Code (hex)	Error
8001	Underrange
8002	Open circuit
8004	Measured value invalid
8008	Cold junction faulty
8020	I/O supply voltage faulty
8010	Configuration invalid
8040	Module faulty
8080	Overrange

A Appendix: CANopen® objects and indexes

A 1 General overview

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

Inline configuration object, index 3000_{hex}, can be used to read the Inline local bus again.

Inline configuration objects, indexes 3001_{hex} ... $300F_{hex}$, can be used to configure the bus coupler for specific Inline error modes.

Inline Interface objects, indexes 3100_{hex} ... 3111_{hex}, can be used to configure and monitor the Inline side of the bus coupler. These objects provide the following information, for example:

– Number of connected Inline devices
– Number of bits and bytes
– Status of the Inline interface

The Inline module objects (indexes 3200_{hex} ... 3207_{hex}) can be used to access the ID codes of the connected I/O modules, for example. These objects also provide information on the CANopen® index and subindex to which an Inline module is mapped to.

The Inline function module objects (indexes 3300_{hex} ... $330F_{hex}$) allows for access and control of modules that are currently not mapped to a CANopen®-approved object. These objects transparently transmit data to and from the Inline modules. It is therefore essential that the user understands how to format the Inline data for these modules.

A 2 Supported codes

A 2.1 SDO abort codes

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

Aborts codes that are not listed are reserved for future use.

A 2.2 Emergency error codes

Error code (hex)	Description
00xx	Error reset or no error
10xx	Generic error
3120	Input voltage to low
3220	Internal voltage to low
3320	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})

A 3 Emergency object data

The emergency telegram consists of 8 bytes with the data represented in the emergency object.

Byte number	Contents
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	Inline faulted module (object 3102_{hex})
Byte 6	Reserved
Byte 7	Reserved

A 4 Predefined connection set

Bit number: COB identifier

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

Identifier allocation scheme for the predefined connection set

A 4.1 Broadcast objects

Object	Function code	Resulting COB ID		Communication parameter for index
	bin	dec	hex	
NMT	0000	0	0	-
SYNC	0001	128	80	1005_{hex}
TIME STAMP	0010	256	100	1012_{hex}, 1013_{hex}

A 4.2 Peer-to-peer objects

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

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

Assigning COB IDs

Pay particular attention when assigning COB IDs to SDOs and PDOs. If you select a COB ID for a PDO that is already assigned to an SDO by the predefined connection set, the bus coupler will accept the COB ID. However, the module may not function as desired. Acceptance of the COB ID provides greater flexibility for the advanced user.

A 5 Data types in the object dictionary

Index	Object	Name
0001	Deftype	Boolean
0002	Deftype	Integer 8
0003	Deftype	Integer 16
0004	Deftype	Integer 32
0005	Deftype	Unsigned 8
0006	Deftype	Unsigned 16
0007	Deftype	Unsigned 32
0008	Deftype	Real 32
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	Integer 24
0011	Deftype	Real 64
0012	Deftype	Integer 40
0013	Deftype	Integer 48
0014	Deftype	Integer 56
0015	Deftype	Integer 64
0016	Deftype	Unsigned 24
0017	Reserved	
0018	Deftype	Unsigned 40
0019	Deftype	Unsigned 48
001A	Deftype	Unsigned 56
001B	Deftype	Unsigned 64
001C ... 001F	Reserved	-
0020	Defstruct	PDO_Communicaton_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 complex 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

A 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

A 6 Object dictionary overview

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

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

A 6.1 Objects specific to the communication profile

Index (hex)	Object	Name	Data type	Access	M/O
Standard objects					
1000	Var	Device type	Unsigned 32	RO	M
1001	Var	Error register	Unsigned 8	RO	M
1002	Var	Manufacturer status register	Unsigned 32	RO	O
1003	Var	Pre-defined error field	Unsigned 32	RO	O
1005	Var	COB-ID SYNC message	Unsigned 32	RW	O
1008	Var	Manufacturer device name	Visible String	Const	O
1009	Var	Manufacturer hardware version	Visible String	Const	O
100A	Var	Manufacturer software version	Visible String	Const	O
100C	Var	Guard time	Unsigned 16	RW	O
100D	Var	Life time factor	Unsigned 8	RW	O

Index (hex)	Object	Name	Data type	Access	M/O
1010	Array	Store parameters	Unsigned 32	RW	O
1011	Array	Restore default parameters	Unsigned 32	RW	O
1012	Var	COB-ID time stamp	Unsigned 32	RW	O
1014	Var	COB-ID emergency message	Unsigned 32	RW	O
1015	Var	Inhibit time EMCY	Unsigned 16	RW	O
1016	Array	Consumer heartbeat time	Unsigned 32	RW	O
1017	Var	Producer heartbeat time	Unsigned 16	RW	O
1018	Record	Identity object	Identity (23_{hex})	RO	M
1020	Array	Verify configuration	Unsigned 32	RW	O
1027	Array	Module list	Unsigned 16	RO	O
1029	Array	Error behaviour	Unsigned 8	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					
RPDO communication parameter					
1400	Record	1st RPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
1401	Record	2nd RPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
...	...	...	...	...	...
141F	Record	32nd RPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
RPDO mapping parameter					
1600	Record	1st RPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O
1601	Record	2nd RPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O
...	...	...	...	...	...
161F	Record	32nd RPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O
TPDO communication parameter					
1800	Record	1st TPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
1801	Record	2nd TPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
...	...	...	...	...	...
181F	Record	32nd TPDO communication parameter	PDO CommPar (20_{hex})	RW	M/O
TPDO mapping parameter					
1A00	Record	1st TPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O
1A01	Record	2nd TPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O
...	...	...	...	...	...
1A1F	Record	32nd TPDO mapping parameter	PDO Mapping (21_{hex})	RW	M/O

Ranges 1600_{hex} to $161F_{hex}$ and $1A00_{hex}$ to $1A1F_{hex}$ are also used to map multiplexed PDOs, see manufacturer specifications in the following table:

A 6.2 Manufacturer-specific I/O objects

Index (hex)	Object	Name	Data type	Access
2000	Array	Digital input latch enable (8 bits)	Unsigned 8	RW
2001	Array	Digital input latch state (8 bits)	Unsigned 8	RW
2400	Array	Analog input range, remanent	Unsigned 16	RW
2401	Array	Analog input range, mapping	Unsigned 16	RW
2410	Array	Analog output response	Unsigned 16	RW

A 6.3 Inline configuration objects

Index (hex)	Object	Name	Data type	Access
3000	Var	Reconfigure I/O	Boolean	RW
3002	Var	Inline fault mode	Unsigned 8	RW
3003	Var	Inline CRC fault mode	Unsigned 8	RW
3004	Var	Inline peripheral fault mode	Unsigned 8	RW
3005	Var	Inline power fault mode	Unsigned 8	RW
3006	Var	Inline module change fault mode	Unsigned 8	RW
3007	Var	Inactive local bus fault mode	Unsigned 8	RW
3008	Var	Inline connection fault mode	Unsigned 8	RW
3009	Var	Inline faulted cycles mode	Unsigned 8	RW
300A	Var	Processor power fault mode	Unsigned 8	RW
300F	Var	Erase configuration	Boolean	RW

A 6.4 Inline interface objects

Index (hex)	Object	Name	Data type	Access
3101	Var	Station status	Unsigned 16	RO
3102	Var	Inline faulted module	Unsigned 8	RO
3103	Var	Inline error max retry	Unsigned 8	RW
3104	Var	Inline number of modules	Unsigned 8	RO
3105	Var	Inline count of bits	Unsigned 16	RO
3106	Var	Inline count of bytes	Unsigned 16	RO
3109	Var	Inline Loop diagnostic count	Unsigned 16	RO
310A	Var	Inline first faulted module	Unsigned 8	RO
310B	Var	Inline last faulted module	Unsigned 8	RO
310C	Var	Inline fault (latched)	Unsigned 8	RO
310D	Var	Inline faulted module (latched)	Unsigned 8	RW
310E	Var	Inline first faulted module (latched)	Unsigned 8	RO
310F	Var	Inline last faulted module (latched)	Unsigned 8	RO
3110	Var	Inline power status	Unsigned 8	RO
3111	Var	Inline interface control byte	Unsigned 8	RW

A 6.5 Inline module objects

Index (hex)	Object	Name	Data type	Access
3200	Array	Inline module ID (stored)	Unsigned 16	RO
3201	Array	Inline module ID (current)	Unsigned 16	RO
3202	Array	Inline module IN data index	Unsigned 16	RO
3203	Array	Inline module IN data first subindex	Unsigned 8	RO
3204	Array	Inline module IN data last subindex	Unsigned 8	RO
3205	Array	Inline module OUT data index	Unsigned 16	RO
3206	Array	Inline module OUT data first subindex	Unsigned 8	RO
3207	Array	Inline module OUT data last subindex	Unsigned 8	RO

A 6.6 Inline special function module objects

Index (hex)	Object	Name	Data type	Access
3300	Array	Special function data size	Unsigned 8	RO
3301	Array	Special function status	Boolean	RO
3302	Array	Special function IN data (1 byte)	Unsigned 8	RO
3303	Array	Special function OUT data (1 byte)	Unsigned 8	RWW
3304	Array	Special function IN data (2 bytes)	Unsigned 16	RO
3305	Array	Special function OUT data (2 bytes)	Unsigned 16	RWW
3306	Array	Special function IN data (3 bytes)	Unsigned 24	RO
3307	Array	Special function OUT data (3 bytes)	Unsigned 24	RWW
3308	Array	Special function IN data (4 bytes)	Unsigned 32	RO
3309	Array	Special function OUT data (4 bytes)	Unsigned 32	RWW
330A	Array	Special function IN data (6 bytes)	Unsigned 48	RO
330B	Array	Special function OUT data (6 bytes)	Unsigned 48	RWW
330C	Array	Special function IN data (8 bytes)	Unsigned 64	RO
330D	Array	Special function OUT data (8 bytes)	Unsigned 64	RWW
330E	Array	Special function IN data (>8 bytes)	Unsigned 8	RO
330F	Array	Special function OUT data (>8 bytes)	Unsigned 8	RWW

A 6.7 Objects for serial communication

Index (hex)	Object	Name	Type	Access	NV
3500	Array	Serial module type	Unsigned 8	RO	V
3501	Array	Serial module status	Boolean	RO	
3502	Array	Serial status word	Unsigned 16	RO	
3503	Array	Serial control word	Unsigned 16	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	Serial protocol	Unsigned 8	RW	NV
3509	Array	Serial baud rate	Unsigned 8	RW	NV

Index (hex)	Object	Name	Type	Access	NV
350A	Array	Serial data width	Unsigned 8	RW	NV
350D	Array	Serial error pattern	Unsigned 8	RW	NV
350E	Array	Serial first delimiter	Unsigned 8	RW	NV
350F	Array	Serial second delimiter	Unsigned 8	RW	NV
3510	Array	3964R priority	Unsigned 8	RW	NV
3511	Array	Serial output type	Unsigned 8	RW	NV
3512	Array	Serial DTR control	Unsigned 8	RW	NV
3513	Array	Serial rotation switch	Unsigned 8	RW	NV
3514	Array	Serial XON pattern	Unsigned 8	RW	NV
3515	Array	Serial XOFF pattern	Unsigned 8	RW	NV
351C	Array	Serial module enable	Boolean	RW	NV

A 6.8 Digital input objects

Index (hex)	Object	Name	Data type	Category
6000	Array	Read input 8 bits	Unsigned 8	C: DI
6005	Var	Global interrupt enable digital 8 bits	Boolean	O
6006	Array	Interrupt mask any change 8 bits	Unsigned 8	O
6100	Array	Read input 16 bits	Unsigned 16	O
6106	Array	Interrupt mask any change 16 bits	Unsigned 16	O
6120	Array	Read input 32 bits	Unsigned 32	O
6126	Array	Interrupt mask any change 32 bits	Unsigned 32	O

A 6.9 Digital output objects

Index (hex)	Object	Name	Data type	Category
6200	Array	Write output 8 bits	Unsigned 8	C: DO
6206	Array	Error mode output 8 bits	Unsigned 8	O
6207	Array	Error value output 8 bits	Unsigned 8	O
6300	Array	Write output 16 bits	Unsigned 16	O
6306	Array	Error mode output 8 bits	Unsigned 16	O
6307	Array	Error value output 8 bits	Unsigned 16	O
6320	Array	Write output 32 bits	Unsigned 32	O
6326	Array	Error mode output 32 bits	Unsigned 32	O
6327	Array	Error state output 32 bits	Unsigned 32	O

A 6.10 Analog input objects

Index (hex)	Object	Name	Data type	Category
6400	Array	Read analog input 8 bits	Integer 8	O
6401	Array	Read analog input 16 bits	Integer 16	C: AI

A 6.11 Analog output objects

Index (hex)	Object	Name	Data type	Category
6410	Array	Write analog output 8 bits	Integer 8	O
6411	Array	Write analog output 16 bits	Integer 16	C:AO

A 6.12 Analog input configuration objects

Index (hex)	Object	Name	Data type	Category
6421	Array	Analog input interrupt trigger selection	Unsigned 8	O
6423	Var	Analog input global interrupt enable	Boolean	C: AI
6424	Array	Analog input interrupt upper limit integer	Integer 32	O
6425	Array	Analog input interrupt lower limit integer	Integer 32	O
6426	Array	Analog input interrupt delta unsigned	Unsigned 32	O

A 6.13 Analog output configuration objects

Index (hex)	Object	Name	Data type	Category
6443	Array	Analog output error mode	Unsigned 8	O
6444	Array	Analog output error value integer	Integer 32	O

A 6.14 PCP interface objects

The PCP interface objects are listed in B “Appendix B: PCP implementation”.

A 7 Objects specific to the communication profile

A 7.1 Object 1000_{hex}: Device type

This object contains information about the device type. It describes the type of device and its function. It is composed of a 16-bit field which describes the device profile and a second 16-bit field which provides additional information about the device functions. The Additional Information parameter is specific to the device profile.

a. Object description

Index	1000_{hex}
Name	Device type
Object	Var
Data type	Unsigned 32
Category	Mandatory

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	00000191_{hex}

c. Byte description

Additional MSB information

Special function

	Bits 7 ... 4	Not defined

I/O function

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

General information of LSB 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

A 7.2 Object 1001_{hex}: Error register

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

a. Object description

Index	1001_{hex}
Name	Error register
Object	Var
Data type	Unsigned 8
Category	Mandatory

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

c. Structure of the error register

Bit	M/O	Support	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 indicated at every error situation.

A 7.3 Object 1002_{hex}: Manufacturer status register

This object is a common status register for manufacturer-specific purposes. This user manual only defines the size and the location of this object.

a. Object description

Index	1002_{hex}
Name	Manufacturer status register
Object	Var
Data type	Unsigned 32
Category	Optional

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 32
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}	

A 7.4 Object 1003_{hex}: Pre-defined error field

This object is used to store the errors that have occurred on the device and the errors indicated by the emergency object. This provides a history of errors related to a device.

– The entry at subindex 0 contains the number of actual errors that are registered in the array starting at subindex 1.
– New errors are stored in subindex 1 and older errors move down the list.
– Writing value “0” to subindex 0 deletes the entire error history (the array is cleared). Values higher than 0 are not allowed to be written. This results in an abort message (error code: 0609 0030_{hex}).
– The error numbers are of type Unsigned 32 and are composed of a 16-bit error code and an additional 16-bit error information field which is manufacturer-specific. The error code is part of the two least significant bytes (LSB) while the additional information is part of the two most significant bytes (MSB. The additional information consists of the information stored in the Station status object (3101_{hex}). If this object is supported, it must consist of at least two entries: the length entry on subindex 0_{hex} and at least one error entry on subindex 1_{hex}.

a. Byte description of the MSB and LSB

Additional information	Error code
Structure of the predefined error field	

b. Object description

Index	1003_{hex}
Name	Pre-defined error field
Object	Array
Data type	Unsigned 32
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	Unsigned 32
Default value	No

Subindex	$\mathbf{2}_{hex}$ **-** $\mathbf{10}_{hex}$
Description	Standard error field
Entry category	Optional
Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	No

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

This object defines the COB ID of the Synchronization object (SYNC). In addition, it defines whether the device generates a SYNC.

a. Structure of the SYNC COB ID entry (Unsigned 32)

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 the 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}). First transmission of the SYNC object starts within a sync cycle after bit 30 has been set to 1. Bits 0 ... 29 must not be changed as long as the object is available (bit 30 = 1).

c. Object description

Index	1005_{hex}
Name	COB-ID SYNC
Object	Var
Data type	Unsigned 32
Category	Conditional; mandatory, if PDO communication on a synchronous base is supported

d. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	00000080_{hex}

A 7.6 Object 1008_{hex}: Manufacturer device name

This object provides 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 CO BK(-XC)-PAC

A 7.7 Object 1009_{hex}: Manufacturer hardware version

This object provides the manufacturer description of the hardware version.

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

A 7.8 Object $100A_{hex}$: Manufacturer software version

This object provides the manufacturer description of the software version.

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

A 7.9 Object $100C_{hex}$: Guard time

This object contains the guard time in milliseconds.

Multiplying the life time factor (Guard time factor object, index $100D_{hex}$) with the guard time results in the life time for the life guarding protocol. The value is 0 when not used.

a. Object description

Index	$100C_{hex}$
Name	Guard time
Object	Var
Data type	Unsigned 16
Category	Conditional; mandatory, if heartbeat is not supported

b.Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 16
Default value	0

A 7.10 Object $100D_{hex}$: Life time factor

This object contains the life time factor.

Multiplying the life time factor with the guard time (Guard time object, index $100C_{hex}$) results in the life time for the node guarding protocol. The value is 0 when not used.

a. Object description

Index	$100D_{hex}$
Name	Life time factor
Object	Var
Data type	Unsigned 8
Category	Conditional; mandatory, if heartbeat is not supported

b. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 8
Default value	0

A 7.11 Object 1010_{hex}: Store parameters

This object supports storing of parameters in a non-volatile memory. By means of read access the device provides information about its storing capabilities. The parameter groups are distinguished as follows:

Subindex 0:	Largest subindex supported.
Subindex 1:	All parameters that can be stored on the device.
Subindex 2:	Communication-related parameters (indexes 1000_{hex} ... $1FFF_{hex}$ manufacturer-specific communication parameters).
Subindex 3:	Application-related parameters (indexes 6000_{hex} ... $9FFF_{hex}$ manufacturer-specific application parameters).
Subindexes 4 ... 127:	Manufacturers may store their choice of parameters.
Subindexes 128 ... 254:	Reserved for future use.

In order to avoid accidental storage of parameters, storage can only be executed when a specific signature is written to the relevant subindex. The signature is “save”.

a. Storage and signature for write access 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 relevant subindex the device stores the parameter and then confirms SDO transmission (initiate download response). If the storing fails, the device responds with an abort of the SDO transfer (abort code: 0606 0000_{hex}).

If a wrong signature is written to the subindex, the device refuses to store the parameter and stops the SDO transfer (abort code: 0800 $002x_{hex}$). On read access to the relevant subindex the device provides information about its storage functions using the following format:

b. Storage and signature for read access (Unsigned 32)

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 non-volatile memory without user request.

d. Object description

Index	1010_{hex}
Name	Store parameters
Object	Array
Data type	Unsigned 32
Category	Optional

e. Entry description

Subindex	**0**$_{hex}$
Description	Largest supported subindex
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	Unsigned 32
Default value	01_{hex}

Subindex	**2**$_{hex}$
Description	Save communication parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	01_{hex}

Subindex	**3**$_{hex}$
Description	Save application parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	02_{hex}

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

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

This object is used to restore the default parameter values according to the communication or device profile. By read access the device provides information about its capabilities to restore these values. The parameter groups are distinguished as follows:

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

In order to avoid accidental restoring of default parameters, restoring can only be executed when a specific signature is written to the relevant subindex. The signature is “Load”.

a. Restoring, signature for write access (ASCII)

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

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

If the storing fails, the device responds with an abort of the SDO transfer (abort code: $0606\ 0000_{hex}$). If a wrong signature is written to the subindex, the device refuses to restore the default parameters and stops the SDO transfer (abort code: $0800\ 002x_{hex}$).

The default values become valid after the device is reset (device reset for subindexes 1_{hex} ... $7F_{hex}$, communication reset for subindex 2_{hex}) or restarted.

On read access to the relevant subindex the device provides information about its capabilities for restoring default parameters using the following format:

b. Read access structure for storing default values (Unsigned 32)

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

c. Read access structure for restoring

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	Unsigned 32
Category	Optional

e. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Largest supported subindex
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	Unsigned 32
Default value	01_{hex}

Subindex	$\mathbf{2}_{hex}$
Description	Restore communication default parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	01_{hex}

Subindex	**3_{hex}**
Description	Restore application default parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	01_{hex}

Subindex	**4_{hex} (see note)**
Description	Restore manufacturer specific parameters
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	01_{hex}

The number and type of physically connected modules control most of the manufacturer-specific parameters. Therefore, subindex 4 is reserved for future use.

A 7.13 Object 1012_{hex}: COB-ID time stamp

This object defines the COB ID of the Time stamp object (TIME). In addition, it defines whether the device receives or generates a TIME.

a. Structure of the TIME COB ID entry (Unsigned 32)

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 0			11-bit identifier	
29bitID	0/1	0/1	1	29-bit identifier				

b. Description of the 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: $0609\ 0030_{hex}$). Bits 0 ... 29 must not be changed as long as the object is available (bit 30 = 1).

c. Object description

Index	1012_{hex}
Name	COB-ID time stamp message
Object	Var
Data type	Unsigned 32
Category	Optional

d. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	100_{hex}

A 7.14 Object 1014_{hex}: COB-ID emergency message

This object defines the COB ID of the emergency object (EMCY).

a. Structure of the EMCY identifier (Unsigned 32)

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 the 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: $0609\ 0030_{hex}$). Bits 0 ... 29 must not be changed as long as the object is available (bit 31 = 1).

c. Object description

Index	1014_{hex}
Name	COB-ID emergency message
Object	Var
Data type	Unsigned 32
Category	Conditional; mandatory, if emergency is supported

d. Entry description

Access	RO; RW as an option
PDO mapping	No
Value range	Unsigned 32
Default value	80h + node ID

A 7.15 Object 1015_{hex}: Inhibit time EMCY

This object can be used to adjust the inhibit time for the EMCY message. If this entry is available, it must be writable in the object dictionary and a multiple of 100 ms.

a. Object description

Index	1015_{hex}
Name	Inhibit time EMCY
Object	Var
Data type	Unsigned 16
Category	Optional

b. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 16
Default value	0

A 7.16 Object 1016_{hex}: Consumer heartbeat time

This object defines the expected heartbeat cycle time and must therefore be larger than the Producer heartbeat time object. The Producer heartbeat time is configured on the device that generates the heartbeat. Monitoring starts after reception of the first heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The producer heartbeat time has to be a multiple of 1 ms.

a. Structure of the consumer heartbeat time entry (Unsigned 32)

MSB	←	←	←	←	←	←	LSB
Bits		31-24			23-16		15-0
Value		Reserved (value: 00_{hex})			Node-ID		Heartbeat time
Encoded as		-			Unsigned 8		Unsigned 16

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

b. Object description

Index	1016_{hex}
Name	Consumer heartbeat time
Object	Array
Data type	Unsigned 32
Category	Optional

c. Entry description

Subindex	**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	Unsigned 32
Default value	0

Subindex	**2_{hex} ... 4_{hex}**
Description	Consumer heartbeat time
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 32
Default value	0

A 7.17 Object 1017_{hex}: Producer heartbeat time

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

a. Object description

Index	1017_{hex}
Name	Producer heartbeat time
Object	Var
Data type	Unsigned 16
Category	Conditional; mandatory, if guarding is not supported

b.Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 16
Default value	0

A 7.18 Object 1018_{hex}: Identity object

This object contains general information about a device.

The vendor ID (subindex 1_{hex}) contains a unique value allocated to the 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® functions are expanded, the major revision has to be incremented. The minor revision number identifies different versions of the same CANopen® behavior.

a. Structure of the 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	$\mathbf{0}_{hex}$
Description	Number of entries
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	1 ... 4
Default value	4

Subindex	$\mathbf{1}_{hex}$
Description	Vendor ID
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	00000084_{hex}

Subindex	2_{hex}
Description	Product code
Entry category	Optional
Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	01000001_{hex}

Subindex	3_{hex}
Description	Revision number
Entry category	Optional
Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	00010001_{hex}

Subindex	4_{hex}
Description	Serial number
Entry category	Optional
Access	RO
PDO mapping	No
Value range	Unsigned 32
Default value	No

A 7.19 Object 1020_{hex}: Verify configuration

If a device supports the storing of parameters in a non-volatile memory, a network configuration tool or the CANopen® manager can use this object to verify the configuration after a device reset and to check if reconfiguration is necessary. The configuration tool stores the date and time in this object and in DCF. The device uses the configuration tool to save its configuration by writing “save” to index 1010_{hex}, subindex 1_{hex}. After a reset the device restores the last configuration and the signature automatically or on request.

If any other command changes the boot-up configuration values, the device resets the verify configuration object to 0. The configuration manager compares the signature and the 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	Unsigned 32	RW	O

The subobjects of the Verify configuration object include:

Index	Subindex	Area for configuration check	Data type
1020_{hex}	0_{hex}	Number of entries	Unsigned 8
	1_{hex}	Configuration date	Unsigned 32
	2_{hex}	Configuration time	Unsigned 32

Since January 1, 1984, the configuration date contains the number of days. The configuration time is expressed ms after midnight.

Application note: Using this object speeds up the boot process significantly. However, the system integrator must be aware that the user could activate the “Store configuration” command, 1010_{hex}, after changing the configuration value without changing the value for 1020_{hex}. The system integrator must therefore ensure absolute correct use of this function.

A 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 for the connection of several module combinations. The 1027_{hex}, Module list object describes the actually connected 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 subobjects for the Module list object include:

c. Entry description

Index	Subindex	Area for configuration check	Data type
1027_{hex}	0_{hex}	Number of connected modules	Unsigned 8
1027_{hex}	1_{hex}	Module 1	Unsigned 16
1027_{hex}	…	…	…
1027_{hex}	N	Module N	Unsigned 16

The continuous subindexes ($1 \leq N \leq 254$) describe the modules according to their order of connection. Each entry contains a number that identifies the module. This number must be unique within all module types that can be connected to this bus coupler device type.

d. Module list

High byte	Low byte
Data length code	Module ID

A 7.21 Object 1029_{hex}: Error behavior

If a serious error occurs in the operational state, the bus coupler automatically changes to the pre-operational state. If object 1029_{hex} is used, the bus coupler can be programmed to alternatively enter the stopped state or remain in the current state in the case of a bus coupler failure.
For this, the bus coupler must be in the operational state.
The errors on the bus coupler include the following communication errors:

– Bus OFF conditions of the CAN interface
– Life guarding event with “occurred” state
– Heartbeat event with “occurred” state

Serious bus coupler errors can also be caused by internal errors: The value of the error class can be represented as follows:

0 = Pre-operational (default)
1 = No state change
2 = Stopped
3 ... 127 = Reserved

a. Object description

Index	1029_{hex}
Name	Error behavior
Object	Array
Data type	Unsigned 8
Category	Optional

b. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number 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	Unsigned 8
Default value	0

A 7.22 Objects 1200_{hex} to 1201_{hex}: Server SDO parameter

The SDO parameter data type is used to describe the SDOs used on a device. The data type has index 22_{hex} in the object dictionary.

a. Structure of the SDO COB ID entry (Unsigned 32)

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 0				11-bit identifier
29-bit ID	0/1	0	1	29-bit identifier				

b. Description of the 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 uses more than one server SDO, the default SDO must be located as the first server SDO at index 1200_{hex}. Read-only entry 2. All additional server SDOs are invalid by default (invalid bit, see description of SDO COB ID entry), their description is located in the subsequent indexes. The COB ID must not be changed as long as the SDO is available.

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	Unsigned 32 (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	Unsigned 32 (Table 53)
Default value	Index 1200_{hex}: 580_{hex} + node ID, Index 1201_{hex} ... $127F_{hex}$: No

A 7.23 Objects 1400_{hex} to $141F_{hex}$: RPDO communication parameter

These objects contain the communication parameters for the PDOs that the device is able to receive. Type of the PDO communication parameter: 20_{hex}.

The entry is interpreted as defined in the “Structure of the PDO COB ID entry” and “Description of the POD COB ID entry” tables.

a. Structure of the PDO COB ID entry (Unsigned 32)

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 0				11-bit identifier
29bitID	0/1	0/1	1	29-bit identifier				

b. Description of the 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 bits allow for the selection of the PDOs that are used in the operational state. If PDOs are available that are completely configured but not used, they can be set to “invalid” (deleted). This function is necessary for devices using more than four RPDOs or four TPDOs, because each device only has 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}). Bits 0 ... 29 must not be changed as long as the PDO is available (bit 31 = 1).

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

c. Transmission methods (PDO transmission)

	Cyclic	Acyclic	Synchro-nous	Asynchro-nous	RTR only
0	-	X	X	-	-
1 - 240	X	-	X	-	-
241 - 251	Reserved				
252	-	-	X	-	X
253	-	-	-	X	X
254	-	-	-	X	-
255	-	-	-	X	-

Synchronous (transmission methods 0 ... 240 and 252) means that the PDO transmission should 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 or to update the transmitted data on the following synchronous transmit PDO. Details of this method 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 method of zero means that the message is transmitted synchronously but not periodically with the SYNC object.

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

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

Transmission methods 252 and 253 mean that the PDO is only transmitted on remote transmission request. For transmission method 252, the data is updated (but not sent) immediately after reception of the SYNC object. For transmission method 253, the data is updated after reception of the remote transmission request (hardware and software restrictions must be taken into account). These values only apply to TPDOs.

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

Subindex 3_{hex} contains the inhibit time. This time is a minimum interval for PDO transmission. The value is defined as multiple of 100 ms. The value must not be changed as long as the PDO is available (bit 31 of subindex 1 = 0).

Subindex 4_{hex} is reserved. If it is not implemented, read or write access results to an SDO transfer abort (abort code: 0609 0011_{hex}).

For types 254/255, an event timer can be used for TPDOs additionally. If an event timer exists for a TPDO (value not equal to 0), the elapsed time is considered to be an event. The elapsed event timer is a multiple of 1 ms of the entry in subindex 5_{hex} of the TPDO. This event will cause this TPDO to be transmitted in addition to otherwise defined events. The timer settings are defined by the occurring events.

Independent of the transmission method, the RPDO event timer is used to detect the expiration of the RPDO.

d. Object description

Index	1400_{hex} ... $141F_{hex}$
Name	RPDO parameter
Object	Record
Data type	PDO CommPar
Category	Conditional; mandatory for each supported PDO

e. Entry description

Subindex	$\mathbf{0}_{hex}$
Description	Number of supported entries
Entry category	Mandatory
Access	RO
PDO mapping	No
Value range	2 – 5

Subindex	$\mathbf{1}_{hex}$
Description	COB ID used by PDO
Entry category	Mandatory
Access	RO; RW if variable COB ID is supported
PDO mapping	No
Value range	Unsigned 32 (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	$\mathbf{2}_{hex}$
Description	Transmission method
Entry category	Mandatory
Access	RO; RW if variable transmission method is supported
PDO mapping	No
Value range	Unsigned 8 (Table 55)
Default value	(Device profile dependent)

Subindex	$\mathbf{3}_{hex}$
Description	Inhibit time (not used for RPDO)
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 16
Default value	No

Subindex	**4_{hex}**
Description	Compatibility entry
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 8
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 Unsigned 16
Default value	No

A 7.24 Objects 1600_{hex} to $161F_{hex}$: RPDO mapping parameter

These objects contain the mapping for the PDOs (Parameter 21_{hex}) that can be received by the device. Subindex 0_{hex} contains the number of valid entries within the mapping record. The number of entries also corresponds to the number of application variables that can be transmitted/received with the corresponding PDO.

Subindexes 1_{hex} to the number of entries contain information about the mapped application variables. The index, subindex, and length of these entries are used to describe the PDO content. The code of all the three values is represented in hexadecimal format. The entry length contains the length of the object in bits (1_{hex} ... 40_{hex}). This parameter is used to verify the overall mapping length. It is mandatory.

The entry structure of subindexes 1_{hex} ... 40_{hex} is as follows:

a. Structure of the PDO mapping entry

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

If PDO mapping cannot be changed (e.g., the PDO length is exceeded or the SDO client attempts to map an object that cannot be mapped) the device stops the SDO transfer service.

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

After all objects have been mapped subindex 0 is set to the valid number of mapped objects. Finally, the PDO is created by writing to the COB ID communication parameter. If subindex 0 is set to 0, the device accepts the new PDO map only before transmitting the response of

the SDO service. If an error is detected, the devices transmits the abort SDO transfer service using abort codes 0602 0000_{hex}, 0604 0041_{hex} or 0604 0042_{hex}. Reading subindex 0 returns the actual number of valid objects mapped.

In the event type (index 1_{hex} ... 7_{hex}) data is mapped, they are used as “dummy entries”. The corresponding data in the PDO is not evaluated by the device. This optional function is useful, for example, to transmit data to several devices using only one PDO, and each device only using one part of the PDO. It is not possible to create dummy mapping for a TPDO.

b. Object description

Index	1600_{hex} ... $161F_{hex}$
Name	RPDO mapping parameter
Object	Record
Data type	PDO mapping
Category	Conditional; mandatory for each supported PDO

c. Entry description

Subindex	**0_{hex}**
Description	Number of mapped objects
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	Unsigned 32
Default value	(Device profile dependent)

A 7.25 Objects 1800_{hex} to $181F_{hex}$: TPDO communication parameter

These objects contain the communication parameters for the PDOs that the device is able to send. A detailed description of the entries can be found in Section “Objects 1400_{hex} to $141F_{hex}$: RPDO communication parameter” on page 87.

a. Object description

Index	1800_{hex} ... $181F_{hex}$
Name	TPDO communication parameter
Object	Record
Data type	PDO CommPar
Category	Conditional; mandatory for each supported PDO

b. Entry description

Subindex	**0**$_{hex}$
Description	Number of supported entries
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	Unsigned 32 (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 method
Entry category	Mandatory
Access	RO; RW if transmission method can be changed
PDO mapping	No
Value range	Unsigned 8 (Table 54)
Default value	(Device profile dependent)

Subindex	**3**$_{hex}$
Description	Inhibit time
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 16
Default value	No

Subindex	**4_{hex}**
Description	Reserved
Entry category	Optional
Access	RW
PDO mapping	No
Value range	Unsigned 8
Default value	No

Subindex	**5_{hex}**
Description	Event timer
Entry category	Optional
Access	RW
PDO mapping	No
Value range	0 = Not used Unsigned 16
Default value	No

A 7.26 Objects $1A00_{hex}$ to $1A1F_{hex}$: TPDO mapping parameter

These objects contain the map for the PDOs that the device is able to send. A detailed description of the entries can be found in Section “Objects 1600_{hex} to $161F_{hex}$: RPDO mapping parameter” on page 90.

a. Object description

Index	$1A00_{hex}$ ... $1A1F_{hex}$
Name	TPDO 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 objects
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	Unsigned 32
Default value	(Device profile dependent)

c. Destination Address Mode (DAM)

The *addr* and *m* areas of the MPDO refer to the consumer. This enables access to the consumer's object dictionary similar to that of an SDO. Setting addr = 0 allows for multicasting and broadcasting and writing to the object dictionary of more than one device simultaneously without needing a PDO for each single object. As with DDOs, initialization of a DAM MPDO depends on the application.

d. Source Address Mode (SAM)

The addr and m areas of the MPDO refer to the producer. Only one producer MPDO of this type is allowed for each node. A 254 or 255 transmission method must be selected.

The producer uses an object scanner list to find out which objects are to be sent. The consumer uses an object dispatcher list as “reference” 3. The restriction of using 32-bit transfers only is not a problem because all the devices know the data types (and sizes) and the associated objects.

e. Entries in the object dictionary

PDO mapping record

Meaning of subindex 0: Increased number of mapped objects. The valid range for non-multiplexed PDOs is 0 to 64. Value 255 indicates a DAM MPDO, and a value of 254 indicates a SAM MPDO.

In source address mode (SAM), further entries in the mapping record (MR) are ignored.

In destination address mode (DAM), the first object describes the local object (only one object can be mapped to 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.	Unsigned 8

A 8 Manufacturer-specific I/O objects

A 8.1 Object 2000_{hex}: Digital input latch enable (8 bits)

If the bits of this object are set to 1, the corresponding digital input is latched when being switched to the defined state (see object 2001_{hex}).

a. Object description

Index	2000_{hex}
Name	Digital input latch enable (8 bits)
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of elements
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 64
Default value	0

Subindex	**1_{hex}**
Description	Digital input latch enable (8 bits) 1
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = number of entries (8 bits)**
Description	Digital input latch enable (8 bits) (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

Reset the memories after changing objects 2000_{hex} and 2001_{hex}. This will delete all the values stored before during a change of the memory configuration. The following options are available:

– Send the “Clear DIP latches” command
(Bit 1 of the Inline interface control byte object, index 3111_{hex}). Or:
– Restart the device.

An error during sending the command or restarting the bus coupler may lead to faulty latched values.

A 8.2 Object 2001_{hex}: Digital input latch state (8 bits)

If the bits of this object are set to 1, the corresponding digital input is latched upon reception of a high logic level. If the bits of this object are set to 0, the corresponding digital input is latched upon reception of a low logic level. The latching function must be enabled (see object 2000_{hex}).

a. Object description

Index	2000_{hex}
Name	Digital input latch state (8 bits)
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of elements
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 64
Default value	0

Subindex	**1_{hex}**
Description	Number of elements (8 bits) 1
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	255
...	...
Subindex	**(N) = number of entries (8 bits)**
Description	Number of elements (8 bits) N
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	255

Reset the memories after changing objects 2000_{hex} and 2001_{hex}. This will delete all the values stored before during a change of the memory configuration. The following options are available:

– Send the “Clear DIP latches” command
(Bit 1 of the Inline interface control byte object, index 3111_{hex}).

Or:

– Restart the device.

An error during sending the command or restarting the bus coupler may lead to faulty latched values.

A 8.3 Object 2400_{hex}: Analog input range, remanent

This object determines the input range for the corresponding analog input, see terminal-specific data sheets.

Table A-1 Difference between objects 2400_{hex} and 2401_{hex}

	2400_{hex}: AIP range, remanent	**2401_{hex}: AIP range, mapping**
Storage	Retentive	Volatile
Mapping to PDO	Not possible	Possible

a. Object description

Index	2400_{hex}
Name	Analog input range
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of elements
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 254
Default value	0

Subindex	**1_{hex}**
Description	Analog input range 1
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = number of entries**
Description	Analog input range (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 8.4 Object 2401_{hex}: Analog input range, mapping

This object determines the input range for the corresponding analog input, see terminal-specific data sheets.

Table A-2 Difference between objects 2400_{hex} and 2401_{hex}

	2400_{hex}: AIP range, remanent	**2401_{hex}: AIP range, mapping**
Storage	Retentive	Volatile
Mapping to process data	Not possible	Possible

a. Object description

Index	2400_{hex}
Name	Analog input range
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of elements
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	0 ... 254
Default value	0

Subindex	**1_{hex}**
Description	Analog input range 1
Access	RW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = number of entries**
Description	Analog input range (N)
Access	RW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No

A 8.5 Object 2410_{hex}: Analog output response

This object contains the value which is transmitted as the response by analog output module (e.g., AO 2 modules) which can be parameterized via process data.

a. Object description

Index	2410_{hex}
Name	Analog output response
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	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 response 1
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = number of analog outputs with response data**
Description	Analog output response (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No

A 9 Inline configuration objects for reading the Inline local bus again

A 9.1 Object 3000_{hex}: Reconfigure I/O

Setting this value to 1 causes all connected modules to be added to the I/O scan and all PDOs defined in CiA 401 to be remapped automatically. This object can only be written to when in the pre-operational state. The configuration is only temporary until the user executes 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

A 10 Inline configuration objects for setting the bus coupler behavior in the event of an error

A 10.1 Object 3002_{hex}: Inline fault mode

This object determines the way the bus coupler controls the outputs in the of an event that hinders the bus coupler from communicating with the I/O modules.

The Inline fault mode objects is used for the following errors:

– Configuration errors
– Module change errors
– Connection errors

Once the bus coupler detects an error, it will respond in one of the following error modes.

Fault modes

0 = Stop running data cycles on the local bus until the error is removed and a command is sent to restart the bus.

1 = Stop running data cycles on the local bus; continuously attempt to remove the error, see note.

2 = Run data cycles with substitute values to the modules that the bus coupler can still communicate with. The data used still correspond to the substitute values defined by the user. The module continues to run in this mode until a command is sent to restart the local bus. The Fault cycles bit is set in the Inline status byte (see below) one of these errors occurs.

3 = Run data cycles to the modules that the bus coupler can still communicate with. The output data used is defined by fault modes 3002_{hex} to $300A_{hex}$. The output data can either be substitute values or process data. The module continues to run in this mode until a command is sent to restart the local bus.

a. Object description

Index	3002_{hex}
Name	Inline fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 3
Default value	1

If the bus coupler is not able to automatically re-establish local bus communication, it can be established by the user. The following options are available:

– Change the state from stop to pre-operational and then to operational.
  Or:
– Send a “Restart bus” command, object 3111_{hex}, bit 2.
  Or:
– Send a Reset node command.

If local bus communication can still not be re-established, switch the module off and check all the module connections.

A 10.2 Object 3003_{hex}: Inline CRC fault mode

This object specifies which state the bus coupler is set to when an Inline CRC error occurs. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3003_{hex}
Name	Inline CRC fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	1

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.3 Object 3004_{hex}: Inline peripheral fault mode

This object specifies which state the bus coupler is set to when an Inline peripheral fault (PF) occurs. The bus coupler must be in the operational state. An example of a peripheral fault is a short circuited output.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3004_{hex}
Name	Inline peripheral fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	1

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stopped state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.4 Object 3005_{hex}: Inline power fault mode

This object specifies which state the bus coupler is set to when an Inline power fault occurs. The bus coupler must be in the operational state. The fault mode becomes active at a voltage of less than approximately 18 V. Power faults may occur with U_{BK}, U_S, U_M or the processor supply.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3005_{hex}
Name	Inline power fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	1

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.5 Object 3006_{hex}: Inline module change fault mode

This object determines whether the bus coupler is set to the operational state in the following cases:

– The active local bus configuration differs from the saved configuration (referred to as “module change” in the following)
– The bus coupler receives the “Start remote node” command.

0 = Hold bus coupler in the pre-operational state

1 = Allow the operational state

a. Object description

Index	3006_{hex}
Name	Inline module change fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 1
Default value	1

1. If the following situation occurs:
   – An attempt is made to set the bus coupler to run mode.
   And:
   – A module change is detected.
   And:
   – The “Inline module change fault mode” object is set to 0.
   This results in the following reaction:
   – The Start error bit in the Station status object (index 3101_{hex}) is set.
   And:
   – An alarm message is generated.
   The alarm message provides information that the bus coupler is not set to run mode.
2. The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.6 Object 3007_{hex}: Inactive local bus fault mode

This object specifies which state the bus coupler is set to when the local bus is inactive and not running data cycles. The bus coupler must be in the operational state.

Object 3007_{hex} is used in the following cases:

– A connection or module change error occurs.
– The bus coupler initializes the local bus.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3007_{hex}
Name	Inactive local bus fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	0

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.7 Object 3008_{hex}: Inline connection fault mode

This object specifies which state the bus coupler is set to when an Inline connection fault occurs. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3008_{hex}
Name	Connection fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	0

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.8 Object 3009_{hex}: Inline faulted cycles mode

This object specifies which state the bus coupler is set to when the bus coupler is running data cycles with substitute values instead of process data. The bus coupler must be in the operational state.

0 = Pre-operational (default)

1 = No state change

2 = Stopped

a. Object description

Index	3009_{hex}
Name	Inline faulted cycles mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	0 ... 2
Default value	1

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.9 Object $300A_{hex}$: Processor power fault mode

This object defines whether the bus coupler communicates with the modules on the local bus or not if the U_{BK} supply falls below 11 V.

The communications power (U_L), for example, is generated from the U_{BK} supply. Therefore, U_{BK} is required for the entire communications supply on the local bus.

A processor blowout condition occurs when the processor power (U_{BK}) falls below 11 V but does not drop low enough to shut down all the I/O modules.

To restart the bus, a “Restart bus” command must be sent. Bit 1 of object 3111_{hex} is used for this.

0 = Keep running cycles (outputs remain in current state)

1 = Reset local bus (default): This sets all the modules to their default output state. For the default values, please refer to the terminal-specific data sheets.

a. Object description

Index	$300A_{hex}$
Name	Processor power fault mode
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 8
Default value	1

The “Start remote node” command can be sent to reset the object. This will correct the problem. If the reason for the error has been removed, the Fault cycles bit will be cleared in the Inline status byte. Furthermore, process data will again be transmitted instead of substitute values. If any module causes the bus coupler to go to the stop state and that error needs special measures to be removed (e.g., a module with a latched PF), it may not be possible to set the bus coupler to the operational state again. In this case, the user can send the “Reset node” command to restart the local bus. This in turn should clear any latched errors, as long as the reason for the error has been removed. It is also possible to change to the pre-operational state and then to acknowledge the peripheral fault. This will ensure that the temporary configuration settings are saved.

A 10.10 Object $300F_{hex}$: Erase configuration

When set to 1, this bit deletes the configuration. In this way, the bus coupler can be auto configured upon first power up. After writing to this object, the bus coupler must be immediately be restarted or reset. In the event of an error, the bus coupler might need to be configured manually. To do so, set all DIP switches to zero, then perform a restart.

a. Object description

Index	$300F_{hex}$
Name	Erase configuration
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	Boolean
Default value	0

A 11 Inline interface objects

A 11.1 Object 3101_{hex}: Station status

This object indicates the Inline station status.
See also “Station status object, index 3101_{hex}” on page 43.

a. Object description

Index	3101_{hex}
Name	Station status
Object	Var
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 16
Default value	0

c. Module status

Inline status (low byte)		
Bit	**Name**	**Description**
0	CRC error	An Inline CRC error has been detected.
1	Peripheral fault (PF)	An Inline peripheral fault has been detected.
2	Power fault	One or more supply inputs is below the required voltage value or not connected. Read the supply status to determine which supply is faulty.
3	Module change	The configuration of the 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 an interruption between two or more I/O modules.
6	Fault cycles	After an error, the bus coupler is using defined substitute values instead of process data for the output modules.
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, if the user changes the device address. Delete this bit by writing the “Save” command to object 1010_{hex} or setting the device ID switch to zero, performing a restart, then setting the desired ID and performing a re-start again.
2. Start error: The module configuration was changed and the user sends the “Start Remote Node” service to request run mode. The configuration of the bus coupler is set not to allow run mode upon a module change (see object 3006_{hex}). Therefore, this bit is set and an emergency message is generated.

A 11.2 Object 3102_{hex}: Inline faulted module

This object contains the number of the first module that has faulted.

0 = IL CO BK(-XC)-PAC

1 = First I/O module connected

2 = Second I/O module connected

... ...

n = N^{th} I/O module connected

a. Object description

Index	3102_{hex}
Name	Inline faulted module
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.3 Object 3103_{hex}: Inline error max retry

This object contains the number of faulty responses the bus coupler accepts before indicating a CRC error. Default = 32

a. Object description

Index	3103_{hex}
Name	Inline error max retry
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RW
PDO mapping	No
Value range	Unsigned 8
Default value	32

A 11.4 Object 3104_{hex}: Inline number of modules

This object indicates the number of Inline modules the bus coupler has detected.

a. Object description

Index	3104_{hex}
Name	Inline number of modules
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	0 ... 63
Default value	No

A 11.5 Object 3105_{hex}: Inline count of bits

This object represents the data size of the Inline modules in bits. This value indicates the total number of bits, process data and PCP.

a. Object description

Index	3105_{hex}
Name	Inline count of bits
Object	Var
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 11.6 Object 3106_{hex}: Inline count of bytes

This object represents the data size of the Inline modules in bytes. This value indicates the total number of bytes, process data and PCP.

a. Object description

Index	3106_{hex}
Name	Inline count of bytes
Object	Var
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 11.7 Object 3109_{hex}: Inline Loop diagnostic count

This object indicates the loop diagnostic count during a connection error. This is useful during first power up where the configuration is unknown.

a. Object description

Index	3109_{hex}
Name	Inline Loop diagnostic count
Object	Var
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 11.8 Object $310A_{hex}$: Inline first faulted module

In the event of a connection error, this value indicates the first module before the error location (break) where local bus communication was interrupted.

a. Object description

Index	$310A_{hex}$
Name	Inline first faulted module (before break)
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.9 Object $310B_{hex}$: Inline last faulted module

In the event of a connection error, this value indicates the last module after the error location (break) where local bus communication was interrupted.

a. Object description

Index	$310B_{hex}$
Name	Inline last faulted module (after break)
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.10 Object $310C_{hex}$: Inline fault (latched)

This object contains the errors latched by the module. See Object 3101_{hex} for a further description.

a. Object description

Index	$310C_{hex}$
Name	Inline fault (latched)
Objects	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

If any fault mode causes the outputs to use substitute values, the original cause of the fault, however, has been removed, the value in this object will indicate the Fault cycles bit only.

A 11.11 Object $310D_{hex}$: Inline faulted module (latched)

This object contains the number of the last module that has faulted.

a. Object description

Index	$310D_{hex}$
Name	Inline faulted module (latched)
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

If any fault mode causes the outputs to use substitute values, the original cause of the fault, however, has been removed, the value in this object is 0.

A 11.12 Object $310E_{hex}$: Inline first faulted module (latched)

This object contains the number of the first module before the error location that was latched during the last connection error.

a. Object description

Index	$310E_{hex}$
Name	Inline first faulted module (latched)
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.13 Object $310F_{hex}$: Inline last faulted module (latched)

This object contains the number of the last module after the error location that was latched during the last connection error.

a. Object description

Index	$310F_{hex}$
Name	Inline last faulted module (latched)
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.14 Object 3110_{hex}: Inline power status

This object indicates the status of the power supplies connected to the bus coupler.

0 = Power supply is OK

1 = Power supply is 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	3110_{hex}
Name	Inline power status
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RO
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 11.15 Object 3111_{hex}: Inline interface control byte

This object contains a byte that can be used to activate the following:

– Acknowledge a peripheral fault
– Delete the latch (memory) of the digital inputs
– Restart the local bus

a. Object description

Index	3111_{hex}
Name	Inline interface control byte
Object	Var
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Access	RWW
PDO mapping	Yes
Value range	Unsigned 8
Default value	0

Bit	Name	Description
0	Acknowledge peripheral faults	This bit is 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 been removed. After the fault has been removed, this value should be set back to a 0 to enable normal operation.
1	Clear all latched digital inputs	When set to 1, this bit 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, this bit will restart the Inline local bus. This method is used to recover from faults that caused the Inline cycles to be stopped.
3	Reserved	
4	Reserved	
5	Reserved	
6	Reserved	
7	Reserved	

A 12 Inline module objects

A 12.1 Object 3200_{hex}: Inline module ID (stored)

This module indicates the Inline ID of the original module configuration.

a. Object description

Index	3200_{hex}
Name	Inline module ID (stored)
Object	Array
Data type	Unsigned 16
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 ID (stored) 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	Inline module ID (stored) (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 12.2 Object 3201_{hex}: Inline module ID (current)

This object indicates the current ID of the Inline module.

a. Object description

Index	3201_{hex}
Name	Inline module ID (current)
Object	Var
Data type	Unsigned 16
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 ID (current) 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	Inline module ID (current) (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 12.3 Object 3202_{hex}: Inline module IN data index

This object indicates the number of the CANopen® index to which the module input data is mapped to (i.e., DIP module = 6000_{hex}, AIP module = 6401_{hex} etc.).

a. Object description

Index	3202_{hex}
Name	Inline module IN data index
Object	Var
Data type	Unsigned 16
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 index module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	IN data index module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 12.4 Object 3203_{hex}: Inline module IN data first subindex

This object indicates the first subindex of the CANopen® index to which the module input data is mapped to.

a. Object description

Index	3203_{hex}
Name	Inline module IN data first subindex
Object	Var
Data type	Unsigned 8
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 first subindex module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	IN data first subindex module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 12.5 Object 3204_{hex}: Inline module IN data last subindex

This object indicates the last subindex of the CANopen® index to which the module input data is mapped to.

a. Object description

Index	3204_{hex}
Name	Inline module IN data last subindex
Object	Var
Data type	Unsigned 8
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 last subindex module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	IN data last subindex module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 12.6 Object 3205_{hex}: Inline module OUT data index

This object indicates the number of the CANopen® index to which the module output data is mapped to (i.e., DOP module = 6200_{hex}, AOP module = 6411_{hex} etc.).

a. Object description

Index	3205_{hex}
Name	Inline module OUT data index
Object	Var
Data type	Unsigned 16
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 index module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	OUT data index module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	No

A 12.7 Object 3206_{hex}: Inline module OUT data first subindex

This object indicates the first subindex of the CANopen® index to which the module output data is mapped to.

a. Object description

Index	3206_{hex}
Name	Inline module OUT data first subindex
Object	Var
Data type	Unsigned 8
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 first subindex module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	OUT data first subindex module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 12.8 Object 3207_{hex}: Inline module OUT data last subindex

This object indicates the last subindex of the CANopen® index to which the module output data is mapped to.

a. Object description

Index	3207_{hex}
Name	Inline module OUT data last subindex
Object	Var
Data type	Unsigned 8
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 last subindex module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected I/O modules**
Description	OUT data last subindex module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 13 Inline special function module objects

A 13.1 Object 3300_{hex}: Special function data size

Object 3300_{hex}, subindex 0, indicates the number of special function modules.
Object 3300_{hex}, subindex > 0 (subindex = module) indicates the module data width in bytes.

a. Object description

Index	3300_{hex}
Name	Special function data size
Object	Var
Data type	Unsigned 8
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 data size 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function module data size (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

A 13.2 Object 3301_{hex}: Special function status

This object indicates the status of the special function modules.

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 status 1
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 status (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	0 ... 1
Default value	No

c. Special function module status

0 = OK

1 = Faulty

A 13.3 Objects 3302_{hex} to $330F_{hex}$: Access to special function modules

Objects 3302_{hex} ... $330F_{hex}$ are used to access the special function modules.

Objects 3302_{hex} ... $330D_{hex}$ enable access to special function modules with a maximum of eight bytes of data.
Objects $330E_{hex}$ and $330F_{hex}$ enable access to special function modules with more than eight bytes of data.

The data length determines the object that is used for access to the special function module data.

Example:

Module IB IL 400 MLR 1-8A uses one byte of input data and one byte of output data.
Objects 3302_{hex} and 3303_{hex} are used for access.

Module IB IL INC-PAC uses four bytes of input data and four bytes of output data.
Objects 3308_{hex} and 3309_{hex} are used for access.

Partial access to particular special function modules is possible by using objects with a shorter data length. (Objects 3302_{hex} and 3303_{hex} can also be used to access the lowest byte of objects 3308_{hex} and 3309_{hex}.)

A 13.4 Object 3302_{hex}: Special function IN data (1 byte)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	3302_{hex}
Name	Special function IN data (1 byte)
Object	Array
Data type	Unsigned 8
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 IN data (1 byte) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function IN data (1 byte) (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 8
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.5 Object 3303_{hex}: Special function OUT data (1 byte)

This object contains the output data from the bus coupler to the Inline module. The Inline module determines the data size.

a. Object description

Index	3303_{hex}
Name	Special function OUT data (1 byte)
Object	Array
Data type	Unsigned 8
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 OUT data (1 byte) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (1 byte) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 8

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.6 Object 3304_{hex}: Special function IN data (2 bytes)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	3304_{hex}
Name	Special function IN data (2 bytes)
Object	Array
Data type	Unsigned 16
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 data IN (2 bytes) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function data IN (2 bytes) module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.7 Object 3305_{hex}: Special function OUT data (2 bytes)

This object contains the output data from the bus coupler to the Inline module. The Inline module determines the data size.

a. Object description

Index	3305_{hex}
Name	Special function OUT data (2 bytes)
Object	Array
Data type	Unsigned 16
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 OUT data (2 bytes) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (2 bytes) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.8 Object 3306_{hex}: Special function IN data (3 bytes)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	3306_{hex}
Name	Special function IN data (3 bytes)
Object	Array
Data type	Unsigned 24
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 IN data (3 bytes) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 24
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function IN data (3 bytes) module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 24
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.9 Object 3307_{hex}: Special function OUT data (3 bytes)

This object contains the output data from the bus coupler to the Inline module. The Inline module determines the data size.

a. Object description

Index	3307_{hex}
Name	Special function OUT data (3 bytes)
Object	Array
Data type	Unsigned 24
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 OUT data (3 bytes) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 24
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (3 bytes) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 24
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.10 Object 3308_{hex}: Special function IN data (4 bytes)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	3308_{hex}
Name	Special function IN data (4 bytes)
Object	Array
Data type	Unsigned 32
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 IN data (4 bytes) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 32
Default value	No
..	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function IN data (4 bytes) module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 32
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.11 Object 3309_{hex}: Special function OUT data (4 bytes)

This object contains the output data from the bus coupler to the Inline module.
The Inline module determines the data size.

a. Object description

Index	3309_{hex}
Name	Special function data OUT (4 bytes)
Object	Array
Data type	Unsigned 32
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 OUT data (4 bytes) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 32
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (4 bytes) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 32
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.12 Object $330A_{hex}$: Special function IN data (6 bytes)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	$330A_{hex}$
Name	Special function IN data (6 bytes)
Object	Array
Data type	Unsigned 48
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 IN data (6 bytes) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 48
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function IN data (6 bytes) module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 48
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.13 Object $330B_{hex}$: Special function OUT data (6 bytes)

This object contains the output data from the bus coupler to the Inline module.
The Inline module determines the data size.

a. Object description

Index	$330B_{hex}$
Name	Special function OUT data (6 bytes)
Object	Array
Data type	Unsigned 48
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 OUT data (6 bytes) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 48
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (6 bytes) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 48
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.14 Object $330C_{hex}$: Special function IN data (8 bytes)

This object contains the input data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	$330C_{hex}$
Name	Special function IN data (8 bytes)
Object	Array
Data type	Unsigned 64
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 IN data (8 bytes) module 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 64
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function IN data (8 bytes) module (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 64
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.15 Object $330D_{hex}$: Special function OUT data (8 bytes)

This object contains the output data from the bus coupler to the Inline module. The Inline module determines the data size.

a. Object description

Index	$330D_{hex}$
Name	Special function OUT data (8 bytes)
Object	Array
Data type	Unsigned 64
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 OUT data (8 bytes) module 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 64
Default value	No
...	...
Subindex	**(N) = Number of special function modules (data length > 9 bytes)**
Description	Special function OUT data (8 bytes) module (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 64
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.16 Object $330E_{hex}$: Special function IN data (> 8 bytes)

This object is used to access modules with a data size of more than eight bytes. The data can be mapped to more than one PDO.
Input data: data from the Inline module to the bus coupler.
The Inline module determines the data size.

a. Object description

Index	$330E_{hex}$
Name	Special function IN data (> 8 bytes)
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of special function IN data bytes (> 8 bytes)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 254
Default value	63

Subindex	**1_{hex}**
Description	Special function IN data (large) byte 1
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of data bytes**
Description	Special function IN data (large) byte (N)
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 8
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 13.17 Object $330F_{hex}$: Special function OUT data (> 8 bytes)

This object is used to access modules with a data size of more than eight bytes. The data can be mapped to more than one PDO. It is the task of the user to ensure data consistency.
Output data from the bus coupler to the Inline module.
The Inline module determines the data size.

a. Object description

Index	$330F_{hex}$
Name	Special function OUT data (> 8 bytes)
Object	Array
Data type	Unsigned 64
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of special function OUT data bytes (> 8 bytes)
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 254
Default value	63

Subindex	**1_{hex}**
Description	Special function OUT data (large) byte 1
Access	RWW
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of data bytes**
Description	Special function OUT data (large) byte (N)
Access	RWW
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 8
Default value	No

Please note Section “Objects 3302_{hex} to $330F_{hex}$: Access to special function modules” on page 132.

A 14 Objects for serial communication

A 14.1 Object 3500_{hex}: Serial module type

Object 3500_{hex}, subindex 0, indicates the number of serial modules.
Object 3500_{hex}, subindex > 0 (subindex = module) indicates the type of the serial module.

a. Object description

Index	3500_{hex}
Name	Serial module type
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial module type module number 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial module type module number (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
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

A 14.2 Object 3501_{hex}: Serial module status

This object indicates the status of the serial modules.

a. Object description

Index	3501_{hex}
Name	Serial module status
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial module status module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Boolean
Default value	No
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial module status module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Boolean
Default value	No

c. Serial module status

0 = OK

1 = Faulty

A 14.3 Object 3502_{hex}: Serial status word

This object contains the status words of the serial modules.

a. Object description

Index	3502_{hex}
Name	Serial status word
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial status word module 1
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial status word module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
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

A 14.4 Object 3503_{hex}: Serial control word

This object contains the control words of the serial modules.

a. Object description

Index	3503_{hex}
Name	Serial control word
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial control word module 1
Access	RWW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	No
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial control word module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
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

A 14.5 Object 3504_{hex}: Serial receive data

This object contains the receive data of the serial modules.

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 ... 16
Default value	16

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 serial modules connected**
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

A 14.6 Object 3505_{hex}: Serial transmit data

This object contains the transmit data of the serial modules.

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 ... 16
Default value	16

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 serial modules connected**
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

A 14.7 Object 3506_{hex}: Serial receive data fragment

This object contains fragmented receive data of the serial modules.

a. Object description

Index	3506_{hex}
Name	Serial receive data 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 ... 16
Default value	16

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 serial modules connected**
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 238.

A 14.8 Object 3507_{hex}: Serial transmit data fragment

This object contains fragmented transmit data of the serial modules.

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 ... 16
Default value	16

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 serial modules connected**
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 238.

A 14.9 Object 3508_{hex}: Serial protocol

This object contains the protocol of the serial modules.

a. Object description

Index	3508_{hex}
Name	Serial protocol
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial protocol module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial protocol module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
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

A 14.10 Object 3509_{hex}: Serial baud rate

This object contains the baud rate of the serial modules.

a. Object description

Index	3509_{hex}
Name	Serial Baud Rate
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial baud rate module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	07_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial baud rate module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
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

A 14.11 Object $350A_{hex}$: Serial data width

This object contains the data width of the serial modules.

a. Object description

Index	$350A_{hex}$
Name	Serial data width
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial data width module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	02_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial data width module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
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

A 14.12 Object $350D_{hex}$: Serial error pattern

This object contains the character for the error pattern of the serial modules.

a. Object description

Index	$350D_{hex}$
Name	Serial error pattern
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial error pattern module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	24_{hex} (‘$’)
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial error pattern module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	24_{hex} (‘$’)

c. Error pattern

Code	Meaning
24_{hex}	**$ (default)**
xx_{hex}	Any character

A 14.13 Object $350E_{hex}$: Serial first delimiter

This object contains the first delimiter of the serial modules.

a. Object description

Index	$350E_{hex}$
Name	Serial first delimiter
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial first delimiter module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	$0D_{hex}$ (CR) Carriage Return
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial first delimiter module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	$0D_{hex}$ (CR) Carriage Return

c. First delimiter

Code	**Meaning**
$0D_{hex}$	**Carriage Return (CR), (default)**
xx_{hex}	Any character

A 14.14 Object $350F_{hex}$: Serial second delimiter

This object contains the second delimiter of the serial modules.

a. Object description

Index	$350F_{hex}$
Name	Serial second delimiter
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial second delimiter module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	$0A_{hex}$ (LF) Line Feed
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial first delimiter module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	$0A_{hex}$ (LF) Line feed

c. Second delimiter

Code	**Meaning**
$0A_{hex}$	**Line feed (LF), (default)**
xx_{hex}	Any character

A 14.15 Object 3510_{hex}: 3964R priority

This object contains the priority of the serial modules.

a. Object description

Index	3510_{hex}
Name	3964R priority
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	3964R priority module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of serial modules connected**
Description	3964R priority module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	0

c. 3964R priority

Code	Meaning
00_{hex}	**Low priority (default)**
01_{hex}	High priority

A 14.16 Object 3511_{hex}: Serial output type

This object contains the output type of the serial modules.

a. Object description

Index	3511_{hex}
Name	Serial output type
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial output type module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0 for 232; 1 for 485; 2 for 422
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial output type module (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
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

A 14.17 Object 3512_{hex}: Serial DTR control

This object contains DTR control of the serial modules.

a. Object description

Index	3512_{hex}
Name	Serial DTR control
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial DTR control module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	00_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial DTR control module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	00_{hex}

c. DTR control (only valid for RS-232 type)

Code	Meaning
00_{hex}	**Automatic (default)**
01_{hex}	Via process data

A 14.18 Object 3513_{hex}: Serial rotation switch

This object contains the rotation switch of the serial modules.

a. Object description

Index	3513_{hex}
Name	Serial rotation switch
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of serial modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	16

Subindex	**1_{hex}**
Description	Serial rotation switch module 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	00_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial rotation switch module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	00_{hex}

c. Rotation switch

Code	Meaning
00_{hex}	**No rotation (default)**
01_{hex}	Rotation

A 14.19 Object 3514_{hex}: Serial XON pattern

This object contains the character for the XON pattern of the serial modules.

a. Object description

Index	3514_{hex}
Name	Serial XON pattern
Object	Array
Data type	Unsigned 8
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	Unsigned 8
Default value	11_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial XON pattern module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	11_{hex}

c. XON pattern

Code	**Meaning**
11_{hex}	**(Default)**
xx_{hex}	Any character (not the same as XOFF pattern)

A 14.20 Object 3515_{hex}: Serial XOFF pattern

This object contains the character for the XOFF pattern of the serial modules.

a. Object description

Index	3515_{hex}
Name	Serial XOFF pattern
Object	Array
Data type	Unsigned 8
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	Unsigned 8
Default value	13_{hex}
...	...
Subindex	**(N) = Number of serial modules connected**
Description	Serial XOFF pattern module (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	13_{hex}

c. XOFF pattern

Code	**Meaning**
13_{hex}	**(Default)**
xx_{hex}	Any character (not the same as XON pattern)

A 14.21 Object $351C_{hex}$: Serial module enable

This object can be used to disable the serial module in order for PCP objects to be used for communication with the module.

0 = Disabled

1 = Enabled

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 ... 16
Default value	16

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 serial modules connected**
Description	Serial enable module (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Boolean
Default value	1

A 15 Digital input objects

A 15.1 Object 6000_{hex}: Read input 8 bits

This object reads groups of 8 input lines as 8-bit information. A maximum of 254 x 8-bit inputs can be addressed (2032 inputs). This object is mandatory for digital input modules and supports all the implemented input lines.

a. Object description

Index	6000_{hex}
Name	Read input 8 bits
Object	Array
Data type	Unsigned 8
Category	Conditional: Device with digital inputs

b. Entry description

Subindex	**0_{hex}**
Description	Number of inputs 8 bits
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	1_{hex} to FE_{hex}
Default value	32

Subindex	**1_{hex}**
Description	Read input 1_{hex} to 8_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Default
Value range	Unsigned 8
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	Unsigned 8
Default value	No
...	...

Subindex	**9_{hex} ... FE_{hex}**
Description	Read input 41_{hex} to 48_{hex} ... Read input $7E8_{hex}$ to $7F0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Optional
Value range	Unsigned 8
Default value	No

CANopen® uses multi-byte accesses to represent the data as the low byte to the high byte. The same data can be accessed using the 8-bit access or multi-byte accesses (e.g., 6000_{hex} and 6100_{hex}). The 8-bit access on the odd subindexes represents the eight lowest bits of the 16-bit access. The even subindexes represent the upper bits of the 16-bit access.

A 15.2 Object 6005_{hex}: Global interrupt enable digital 8 bits

This object is used to globally enable and disable 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} (or 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 method.

TRUE = Global interrupt enabled

FALSE = Global interrupt disabled

a. Object description

Index	6005_{hex}
Name	Global interrupt enable digital 8 bits
Object	Variable
Data type	Boolean
Category	Optional

b. Entry description

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

A 15.3 Object 6006_{hex}: Interrupt mask any change 8 bits

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

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6006_{hex}
Name	Interrupt mask any change 8 bits
Object	Array
Data type	Unsigned 8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of inputs 8 bits
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	Unsigned 8
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	Unsigned 8
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	Unsigned 8
Default value	FF_{hex}

A 15.4 Object 6100_{hex}: Read input 16 bits

This object reads groups of 16 input lines as 16-bit information. A maximum of 254 x 16-bit inputs can be addressed (4064 inputs).

a. Object description

Index	6100_{hex}
Name	Read input 16 bits
Object	Array
Data type	Unsigned 16
Category	Conditional: optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of inputs 16 bits
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	Unsigned 16
Default value	No

Subindex	**2_{hex}**
Description	Read input 11_{hex} to 20_{hex}
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No
...	...
Subindex	**FE_{hex}**
Description	Read input $FD0_{hex}$ to $FE0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 16
Default value	No

A 15.5 Object 6106_{hex}: Interrupt mask any change 16 bits

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

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6106_{hex}
Name	Interrupt mask any change 16 bits
Object	Array
Data type	Unsigned 16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of inputs 16 bits
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	Unsigned 16
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	Unsigned 16
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	Unsigned 16
Default value	$FFFF_{hex}$

A 15.6 Object 6120_{hex}: Read input 32 bits

This object reads groups of 16 input lines as 32-bit information. A maximum of 254 x 32-bit inputs can be addressed (8128 inputs).

a. Object description

Index	6120_{hex}
Name	Read input 32 bits
Object	Array
Data type	Unsigned 32
Category	Conditional: optional

b. Entry description

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

Subindex	**1_{hex}**
Description	Read inputs 1_{hex} to 20_{hex}
Access	RO
Entry category	Mandatory
PDO mapping	Yes
Value range	Unsigned 32
Default value	No

Subindex	**2_{hex}**
Description	Read inputs 21_{hex} to 40_{hex}
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 32
Default value	No
...	...
Subindex	**FE_{hex}**
Description	Read inputs $1FA0_{hex}$ to $1FC0_{hex}$
Access	RO
Entry category	Optional
PDO mapping	Yes
Value range	Unsigned 32
Default value	No

A 15.7 Object 6126_{hex}: Interrupt mask any change 32 bits

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

1 = Interrupt enabled

0 = Interrupt disabled

a. Object description

Index	6126_{hex}
Name	Interrupt mask any change 32 bits
Object	Array
Data type	Unsigned 32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of inputs 32 bits
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 input 1_{hex} to 20_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$

Subindex	**2_{hex}**
Description	Interrupt any change input 21_{hex} to 40_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$
...	...
Subindex	**FE_{hex}**
Description	Interrupt any change input $1FA1_{hex}$ to $1FC0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$

A 16 Digital output objects

A 16.1 Object 6200_{hex}: Write output 8 bits

This object compiles a group of eight output lines as one byte of information. A maximum of 254 x 8-bit output blocks can be addressed.

a. Object description

Index	6200_{hex}
Name	Write output 8 bits
Object	Array (8_{hex})
Data type	Unsigned 8
Category	Conditional: Device with digital outputs

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 8 bits
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	Unsigned 8
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	Unsigned 8
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	Unsigned 8
Default value	0_{hex}

CANopen® uses multi-byte accesses to represent the data as the low byte to the high byte. The same data can be accessed using the 8-bit access or multi-byte accesses (e.g., 6200_{hex} and 6300_{hex}). The 8-bit access on the odd subindexes represents the eight lowest bits of the 16-bit access. The even subindexes represent the upper bits of the 16-bit access.

A 16.2 Object 6206_{hex}: Error mode output 8 bits

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

1 = The output value is set to the substitute value defined in object 6207_{hex}.

0 = The output value is held if an error occurs.

a. Object description

Index	6206_{hex}
Name	Error mode output 8 bits
Object	Array
Data type	Unsigned 8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 8 bits
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	Unsigned 8
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	Unsigned 8
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	Unsigned 8
Default value	FF_{hex}

A 16.3 Object 6207_{hex}: Error value output 8 bits

If the relevant error mode is active, this object defines the substitute value the outputs are set to in the event of an error.

0 = In the event of an error, the output is set to “0”, provided that object 6206_{hex} is enabled.

1 = In the event of an error, the output is set to “1”, provided that object 6206_{hex} is enabled.

a. Object description

Index	6207_{hex}
Name	Error value output 8 bits
Object	Array
Data type	Unsigned 8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 8 bits
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	Unsigned 8
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	Unsigned 8
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	Unsigned 8
Default value	0_{hex}

A 16.4 Object 6300_{hex}: Write output 16 bits

This object compiles a group of 16 output lines as 2-byte information. A maximum of 254 x 16-bit output blocks can be addressed.

a. Object description

Index	6300_{hex}
Name	Write output 16 bits
Object	Array
Data type	Unsigned 16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 16 bits
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	Unsigned 16
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	Unsigned 16
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	Unsigned 16
Default value	0_{hex}

A 16.5 Object 6306_{hex}: Error mode output 16 bits

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

1 = The output value is set to the substitute value defined in object 6307_{hex}.

0 = The output value is held if an error occurs.

a. Object description

Index	6306_{hex}
Name	Error mode output 16 bits
Object	Array
Data type	Unsigned 16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 16 bits
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	Unsigned 16
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	Unsigned 16
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	Unsigned 16
Default value	$FFFF_{hex}$

A 16.6 Object 6307_{hex}: Error value output 16 bits

If the relevant error mode is active, this object defines the substitute value the outputs are set to in the event of an error.

0 = In the event of an error, the output is set to “0”, provided that object 6306_{hex} is enabled.

1 = In the event of an error, the output is set to “1”, provided that object 6306_{hex} is enabled.

a. Object description

Index	6307_{hex}
Name	Error value output 16 bits
Object	Array
Data type	Unsigned 16
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 16 bits
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	Unsigned 16
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	Unsigned 16
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	Unsigned 16
Default value	0_{hex}

A 16.7 Object 6320_{hex}: Write output 32 bits

This object compiles a group of 32 output lines as 4-byte information. A maximum of 254 x 32-bit output blocks can be addressed.

a. Object description

Index	6320_{hex}
Name	Write output 32 bits
Object	Array
Data type	Unsigned 32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 32 bits
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	Unsigned 32
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	Unsigned 32
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	Unsigned 32
Default value	0_{hex}

A 16.8 Object 6326_{hex}: Error mode output 32 bits

This object indicates whether an output is set to a predefined substitute value (see object 6327_{hex}) in the event of an internal device failure.

1 = The output value is set to the substitute value defined in object 6327_{hex}.

0 = The output value is held if an error occurs.

a. Object description

Index	6326_{hex}
Name	Error mode output 32 bits
Object	Array
Data type	Unsigned 32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 32 bits
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 20_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$

Subindex	**2_{hex}**
Description	Error mode output 21_{hex} to 40_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$
...	...
Subindex	**FE_{hex}**
Description	Error mode output $1FC0_{hex}$ to $1FE0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	FFFF $FFFF_{hex}$

A 16.9 Object 6327_{hex}: Error value output 32 bits

If the relevant error mode is active, this object defines the substitute value the outputs are set to in the event of an error.

0 = In the event of an error, the output is set to “0”, provided that object 6326_{hex} is enabled.

1 = In the event of an error, the output is set to “1”, provided that object 6326_{hex} is enabled.

a. Object description

Index	6327_{hex}
Name	Error value output 32 bits
Object	Array
Data type	Unsigned 32
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of outputs 32 bits
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 20_{hex}
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Error value output 21_{hex} to 40_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Error value output $1FC0_{hex}$ to $1FE0_{hex}$
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	0_{hex}

A 17 Analog input objects

A 17.1 Object 6400_{hex}: Read analog input 8 bits

This object reads the value of input channel “n”. The value has a width of 8 bits or less. The value is always aligned on the left. The remaining bits on the right side of the LSB are set to zero.

a. Object description

Index	6400_{hex}
Name	Read analog input 8 bits
Object	Array
Data type	Integer8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs 8 bits
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® uses multi-byte accesses to represent the data as the low byte to the high byte. The same data can be accessed using the 8-bit access or multi-byte accesses (e.g., 6400_{hex} and 6401_{hex}). The 8-bit access on the odd subindexes represents the eight lowest bits of the 16-bit access. The even subindexes represent the upper bits of the 16-bit access.

A 17.2 Object 6401_{hex}: Read analog input 16 bits

This object reads the value of input channel “n”. The value has a width of 16 bits or less. The value is always aligned on the left. The remaining bits on the right side of the LSB are set to zero.

a. Object description

Index	6401_{hex}
Name	Read analog input 16 bits
Object	Array
Data type	Integer16
Category	Conditional: Device with analog input

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog inputs 16 bits
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

A 18 Analog output objects

A 18.1 Object 6410_{hex}: Write analog output 8 bits

This object writes an Integer8 value to output channel “n”. The value is always aligned on the left.

a. Object description

Index	6410_{hex}
Name	Write analog output 8 bits
Object	Array
Data type	Integer8
Category	Optional

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog outputs 8 bits
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® uses multi-byte accesses to represent the data as the low byte to the high byte. The same data can be accessed using the 8-bit access or multi-byte accesses (e.g., 6410_{hex} and 6411_{hex}). The 8-bit access on the odd subindexes represents the eight lowest bits of the 16-bit access. The even subindexes represent the upper bits of the 16-bit access.

A 18.2 Object 6411_{hex}: Write analog output 16 bits

This object writes an Integer16 value to output channel “n”. The value is always aligned on the left.

a. Object description

Index	6411_{hex}
Name	Write analog output 16 bits
Object	Array
Data type	Integer16
Category	Conditional: Device with analog output

b. Entry description

Subindex	**0_{hex}**
Description	Number of analog outputs 16 bits
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}

Setting up the analog inputs

Use care when enabling the analog input interrupts. The analog interrupt deltas, the upper limits and lower limits should be configured first. Any errors during interrupt configuration prior to using the global enable object “Analog input interrupt global enable” will cause PDO messages to be generated in the event of bit changes of an analog input, and will lead to a CAN “bus saturation”. This means that the bus will be full of PDO messages from the analog inputs. It may therefore not be possible to place messages with a low-priority CAN ID onto the bus. Another method for controlling the number of generated interrupt messages is to set an inhibit time in the communication parameter of the associated PDO.

A 19 Analog input configuration objects

A 19.1 Object 6421_{hex}: Analog input interrupt trigger selection

This object determines which events trigger an interrupt for a specific channel. The bits listed below represent the options that can be used to trigger an interrupt.

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	Interrupt trigger selection
Object	Array
Data type	Unsigned 8
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	0 ... 31
Default value	7

Subindex	**2_{hex}**
Description	Analog input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	0 ... 31
Default value	7
...	...
Subindex	**FE_{hex}**
Description	Analog input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	0 ... 31
Default value	7

A 19.2 Object 6423_{hex}: Analog input global interrupt enable

This object can be used to globally disable or enable the interrupt behavior without changing the interrupt mask.

By default, analog input do not activate 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 input

b. Object description

Subindex	$\mathbf{0}_{hex}$
Access	RW
PDO mapping	Optional
Value range	Boolean
Default value	FALSE

A 19.3 Object 6424_{hex}: Analog input interrupt upper limit integer

When enabled (see object 6423_{hex}), an interrupt is triggered in case an analog input exceeds the given value. The value is always aligned on the left. As long as the trigger condition is met, every change of the analog input data generates a new interrupt, as long as there is no additional trigger condition, e.g., 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}

A 19.4 Object 6425_{hex}: Analog input interrupt lower limit integer

When enabled (see object 6423_{hex}), an interrupt is triggered in case an analog input falls below the given value. The value is always aligned on the left. As long as the trigger condition is met, every change of the analog input data generates a new interrupt, as long as there is no additional trigger condition, e.g., input interrupt delta (6426_{hex}).

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}

A 19.5 Object 6426_{hex}: Analog input interrupt delta unsigned

This value indicates the delta value (above or below the value transmitted last) for analog inputs for which an interrupt has been enabled (see object 6423_{hex}).

a. Object description

Index	6426_{hex}
Name	Analog input interrupt delta unsigned
Object	Array
Data type	Unsigned 32
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	Unsigned 32
Default value	0_{hex}

Subindex	**2_{hex}**
Description	Analog input 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	0_{hex}
...	...
Subindex	**FE_{hex}**
Description	Analog input FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 32
Default value	0_{hex}

A 20 Analog output configuration objects

A 20.1 Object 6443_{hex}: Analog output error mode

This object indicates whether an output is set to a predefined substitute value (see object 6444_{hex}) in the event of an internal device failure.

1 = The output value is set to the substitute value defined in object 6444_{hex}.

0 = The output value is held if an error occurs.

a. Object description

Index	6443_{hex}
Name	Analog output error mode
Object	Array
Data type	Unsigned 8
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	Unsigned 8
Default value	1_{hex}

Subindex	**2_{hex}**
Description	Error mode analog output 2_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	1_{hex}
...	...

Subindex	**FE_{hex}**
Description	Error mode analog output FE_{hex}
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	1_{hex}

A 20.2 Object 6444_{hex}: Analog output error value integer

If the relevant error mode is active, this object defines the substitute value the outputs are set to in the event of an error.

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}

A 21 Objects that are stored retentively

Table A-3 lists the objects that are stored retentively on the bus coupler.

In the Pre-Operational state, the bus coupler immediately stores data retentively.
In the Operational state, it stores the data non-retentively. The data is only stored retentively to the Flash when switching to a non-time-critical state (e.g., Pre-Operational).

Set the SDO timeout parameter to at least 1000 ms.

Table A-3 Objects that are stored retentively

Index (hex)	Name	Note
2000	Digital input latch enable (8 bits)	Manufacturer-specific I/O objects
2001	Digital input latch state (8 bits)	
2400	Analog input range, remanent	
3002	Inline fault mode	Inline configuration objects for setting the bus coupler behavior in the event of an error
3003	Inline CRC fault mode	
3004	Inline peripheral fault mode	
3005	Inline power fault mode	
3006	Inline module change fault mode	
3007	Inactive local bus fault mode	
3008	Inline connection fault mode	
3009	Inline faulted cycles mode	
300A	Processor power fault mode	
3103	Inline error max retry	Inline interface objects
3200	Inline module ID (stored)	Inline module objects
3201	Inline module ID (current)	
3500	Serial module type	Objects for serial communication
3508	Serial protocol	
3509	Serial baud rate	
350A	Serial data width	
350D	Serial error pattern	
350E	Serial first delimiter	
350F	Serial second delimiter	
3510	3964R priority	
3511	Serial output type	
3512	Serial DTR control	
3513	Serial rotation switch	
3514	Serial XON pattern	
3515	Serial XOFF pattern	
351C	Serial module enable	

Table A-3 Objects that are stored retentively [...]

Index (hex)	Name	Note
6005	Global interrupt enable digital 8 bits	Digital input objects
6006	Interrupt mask any change 8 bits	
6106	Interrupt mask any change 16 bits	
6126	Interrupt mask any change 32 bits	
6206	Error mode output 8 bits	Digital output objects
6207	Error value output 8 bits	
6306	Error mode output 8 bits	
6307	Error value output 8 bits	
6326	Error mode output 32 bits	
6327	Error state output 32 bits	
6421	Analog input interrupt trigger selection	Analog input configuration objects
6424	Analog input interrupt upper limit integer	
6425	Analog input interrupt lower limit integer	
6426	Analog input interrupt delta unsigned	
6443	Analog output error mode	Analog output configuration objects
6444	Analog output error value integer	

B Appendix B: PCP implementation

B 1 General overview

The IL CO BK(-XC)-PAC bus coupler allows for PCP information access using SDOs or PDOs. In the event of 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. In this case, however, the request needs to be fragmented because of the 8-bit length restriction for CANopen®. The response from the module is also fragmented for the same reason. The handshaking required for fragmentation is described in detail in the following sections.

B 2 Manufacturer-specific data types

B 2.1 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	Unsigned 8
	1_{hex}	Service	Unsigned 8
	2_{hex}	Request Data Byte 1	Unsigned 8
	3_{hex}	Request Data Byte 2	Unsigned 8
	4_{hex}	Request Data Byte 3	Unsigned 8
	5_{hex}	Request Data Byte 4	Unsigned 8
	6_{hex}	Request Data Byte 5	Unsigned 8
	7_{hex}	Request Data Byte 6	Unsigned 8
	8_{hex}	Request Data Byte 7	Unsigned 8

B 2.2 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	Unsigned 8
	1_{hex}	Service	Unsigned 8
	2_{hex}	Response Data Byte 1	Unsigned 8
	3_{hex}	Response Data Byte 2	Unsigned 8
	4_{hex}	Response Data Byte 3	Unsigned 8
	5_{hex}	Response Data Byte 4	Unsigned 8
	6_{hex}	Response Data Byte 5	Unsigned 8
	7_{hex}	Response Data Byte 6	Unsigned 8
	8_{hex}	Response Data Byte 7	Unsigned 8

B 3 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	Unsigned 8	RO
3401	Array	PCP module channel size	Unsigned 8	RO
3402	Array	PCP module status	Unsigned 8	RO
3403	Array	PCP request	DOMAIN	RW
3404	Array	PCP response	DOMAIN	RO
3405	Array	PCP module number	Unsigned 8	RW
3406	Array	PCP write index	Unsigned 16	RW
3407	Array	PCP write subindex	Unsigned 8	RW
3408	Array	PCP write data	DOMAIN	RW
3409	Array	PCP read index	Unsigned 16	RW
340A	Array	PCP read subindex	Unsigned 8	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	Unsigned 8	RW
340F	Array	PCP read invoke ID	Unsigned 8	RW
3410	Array	PCP write data confirmation	DOMAIN	RO
3411	Array	PCP read data confirmation	DOMAIN	RO

B 3.1 Object 3400_{hex}: PCP PDU size

This object contains the value of the PDU size used for the PCP channel of the module. Typically, this value is 64 bytes. This value is the maximum number of bytes that the PCP channel can transmit per request/response.

a. Object description

Index	3400_{hex}
Name	PCP PDU size
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP module 1 PDU size
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP module (N) PDU size
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

B 3.2 Object 3401_{hex}: PCP module channel size

This object contains the value of the PCP channel size. This indicates the number of bytes that are transferred during an Inline scan.

a. Object description

Index	3401_{hex}
Name	PCP module channel size
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP module 1 channel size
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP module (N) channel size
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

B 3.3 Object 3402_{hex}: PCP module status

a. Object description

Index	3402_{hex}
Name	PCP module status
Object	Array
Data type	Unsigned 8
Category	Mandatory

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0
...	...
Subindex	**1_{hex}**
Description	PCP module 1 status
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	No
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP module (N) status
Access	RO
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	No

c. Status: Indicates 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 service-specific PCP error has occurred. See the Error class and Error code bytes.
2	PCP channel busy	A PCP transaction is already in progress, e.g., 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 fragment received after last fragment, middle fragment 2 received before fragment 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 3.4 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 ... 16
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 connected PCP modules**
Description	PCP request module (N)
Access	RWW
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

c. Format

Services without 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 3.5 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 ... 16
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 connected 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 service-specific PCP error has occurred. See the Error class and Error code bytes.
2	PCP channel busy	A PCP transaction is already in progress, e.g., 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 fragment received after last fragment, middle fragment 2 received before fragment 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 without 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 without 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 without 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 without 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 3.6 Object 3405_{hex}: PCP module number

The attribute of the PCP module object allows for setting the number of PCP modules that are accessed by the read and write objects (3408_{hex} and $340C_{hex}$).

PCP abort response without 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	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP module number 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	0 ... number of connected PCP modules
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP module number (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	0 ... number of connected PCP modules
Default value	0

B 3.7 Object 3406_{hex}: PCP write index

The PCP write index object is used to set the PCP object dictionary index to which the data of the PCP write data object (3408_{hex}) is written.

a. Object description

Index	3406_{hex}
Name	PCP write index
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP write index 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP write index (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	0

B 3.8 Object 3407_{hex}: PCP write subindex

The PCP write subindex object sets the PCP object dictionary subindex that is written when writing to the PCP write data object (3408_{hex}).

a. Object description

Index	3407_{hex}
Name	PCP write subindex
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP write subindex 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP write subindex (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	0

B 3.9 Object 3408_{hex}: PCP write data

The PCP write data object is used for writing data to the PCP object dictionary that is referenced by the PCP module, PCP write index and PCP write subindex objects.

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 ... 16
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 connected PCP modules**
Description	PCP write data (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

c. Format for writing data

UNSIGN8	Size for writing PCP data
Array	[Size for writing PCP data] of the Unsigned 8 data

In the event of an SDO abort, the reasons for the abort can be read from object 3410_{hex}. For further details, see PCP write data confirmation object (3410_{hex}).

B 3.10 Object 3409_{hex}: PCP read index

The PCP read index object is used to set the PCP object dictionary index that is read when reading the PCP read data object ($340B_{hex}$).

a. Object description

Index	3409_{hex}
Name	PCP read index
Object	Array
Data type	Unsigned 16
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP read index 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 16
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP read index (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 16
Default value	0

B 3.11 Object $340A_{hex}$: PCP read subindex

The PCP read subindex object is used to set the PCP object dictionary subindex that is read when reading the PCP read data object ($340B_{hex}$).

a. Object description

Index	$340A_{hex}$
Name	PCP read subindex
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP read subindex 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP read subindex (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	0

B 3.12 Object $340B_{hex}$: PCP read data

The PCP read data object is used to read data from the PCP object dictionary that is referenced by the PCP module, PCP read index and PCP read subindex objects.

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 ... 16
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 connected PCP modules**
Description	PCP read data (N)
Access	RO
Entry category	Optional
PDO mapping	No
Value range	DOMAIN
Default value	No

c. Format for reading data

UNSIGN8	Size of the response data
Array	[Size of the response data] of Unsigned 8 data

In the event of an SDO abort, the reasons for the abort can be read from object 3411_{hex}. For further details, see PCP read data confirmation object.

B 3.13 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 ... 16
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 connected 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 238.

B 3.14 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 ... 16
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 connected PCP modules**
Description	PCP response fragment module (N)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 238.

B 3.15 Object $340E_{hex}$: PCP write invoke ID

a. Object description

Index	$340E_{hex}$
Name	PCP write invoke ID
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP write invoke ID 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP write invoke ID (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	0

The PCP write invoke ID object is used to se the PCP invoke ID that is accessed by the PCP write data object (3408_{hex}).

B 3.16 Object $340F_{hex}$: PCP read invoke ID

The PCP read invoke ID object is used to set the PCP invoke ID that is accessed by the PCP read data object ($340B_{hex}$).

a. Object description

Index	$340F_{hex}$
Name	PCP read invoke ID
Object	Array
Data type	Unsigned 8
Category	Manufacturer

b. Entry description

Subindex	**0_{hex}**
Description	Number of PCP modules
Access	RO
Entry category	Mandatory
PDO mapping	No
Value range	0 ... 16
Default value	0

Subindex	**1_{hex}**
Description	PCP read invoke ID 1
Access	RW
Entry category	Mandatory
PDO mapping	No
Value range	Unsigned 8
Default value	0
...	...
Subindex	**(N) = Number of connected PCP modules**
Description	PCP read invoke ID (N)
Access	RW
Entry category	Optional
PDO mapping	No
Value range	Unsigned 8
Default value	0

B 3.17 Object 3410_{hex}: PCP write data confirmation

This object can be read to determine the reason of the SDO abort when using the PCP write data object.

The status byte of object 3402_{hex} (PCP module status) must be checked to determine the type of response. The following format is used:

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 ... 16
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 connected PCP modules**
Description	PCP write data confirmation (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 service-specific PCP error has occurred. See the Error class and Error code bytes.
2	PCP channel busy	A PCP transaction is already in progress, e.g., 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 fragment received after last fragment, middle fragment 2 received before fragment 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 3.18 Object 3411_{hex}: PCP read data confirmation

This object can be read to determine the reason of the SDO abort when using the PCP read data object.

The status byte of object 3402_{hex} (PCP module status) must be checked to determine the type of response. The following format is used:

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 ... 16
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 connected PCP modules**
Description	PCP read data confirmation (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 (see error codes)	A service-specific PCP error has occurred. See the Error class and Error code bytes.
2	PCP channel busy	A PCP transaction is already in progress, e.g., 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 fragment received after last fragment, middle fragment 2 received before fragment 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 4 PCP access using SDOs

The user has two options for accessing PCP information. The two methods differ as follows: Method 1 configures the invoke ID, index, and subindex individually followed by writing or reading the data to be transferred. When using method 2 a request is sent to the PCP request object. The response is read via the PCP response object. The request must contain the service, index, subindex, invoke ID and the data. Method 1 is more efficient when reading from or writing to the same object repeatedly. A good example is reading received data from and writing transmit data to serial modules. Method 2 is more efficient for module configuration that requires multiple read and write operations from and to different objects. Whichever method is chosen, only one method should be used for each PCP module. After PCP access is complete the user can choose the same method or a different method.

B 4.1 PCP implementation of read/write data

Method 1 requires the PCP read object or PCP write object (invoke ID, index, subindex) to be set depending on whether a read or write service is needed. For actual data transmission the relevant data object must be accessed thereafter. Using the CANopen® subindex, every PCP access object (i.e., subindex 1 indicates PCP module 1, subindex 2 indicates PCP module 2 etc.) indicates which PCP module is to be accessed.
For writing, the user sets the PCP write invoke ID, PCP write index and PCP write subindex objects. Then the data is written to the PCP write data object. The first byte must contain the number of bytes to be written. If the write operation is successful, this SDO write operation is confirmed. If the write operation fails, the result can be read from the PCP write confirmation object.
For reading, the user sets the PCP read invoke ID, PCP read index and PCP read subindex objects. Then the data must be read from the PCP read data object. If the read operation is valid, the number of data bytes followed by the data read is returned. In the event of an error, the result can be read from the PCP read confirmation object.

An example of method 1 is writing serial data (“Hello”) to an RS-232 module.

Where x = PCP module number of the RS-232 module.

1. Set subindex x of the PCP write invoke ID object to 0.
2. Set subindex x of the PCP write index object to $5FE0_{hex}$.
3. Set subindex x of the PCP write subindex object to 0.
4. Send the data to subindex of the PCP write data object to 05_{hex}, 48_{hex}, 65_{hex}, $6C_{hex}$, $6C_{hex}$, $6F_{hex}$.

The bus coupler sends the PCP request to the specified PCP module, waits for a response, and returns the response as “successful” or “faulty”. In the event of an error message, subindex x of the PCP write data confirmation object should be read for details on the failed PCP write operation. Use care when using this method, as the CANopen® channel cannot process any further data when a response is pending. This should affect module configuration but should be taken into account when running process data.

B 4.2 PCP implementation of the request/response

Method 2 requires to send a PCP request to the module and then to read the PCP response to receive the status or data. The user fist creates a request and then uses an SDO to send the request to the bus coupler. The bus coupler checks the request for the correct length and a valid service. In the event of 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. If the request was a read request, the PCP response object must be read to receive the response. The bus coupler returns the “busy” status until the PCP module returns a 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 operation and the response read operation. The time of most read/write operations is in the range of a few milliseconds, again determined by the configuration, PCP channel size, and type of request.

a. Request format

Services without 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 response

Services without 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 without 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

Example of setting the RS-232 baud rate to 9600 using the request/response method:

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 corresponds to $5FFF_{hex}$)
Byte 4	Index high = $5F_{hex}$
Byte 5	Subindex = 02_{hex} (baud rate corresponds to 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 by 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

Example of reading the RS-232 data width using the request/response method:

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 corresponds to 5FFF-$_{hex}$)
Byte 4	Index high = $5F_{hex}$
Byte 5	Subindex = 03_{hex} (data width corresponds to 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}

B 5 PCP fragmentation implementation

B 5.1 PCP communication via CANopen® process data

PCP transfer can also be accomplished by using process data. This uses the same method as the request/response method previously described, with the exception that the request and response must be divided into fragments of eight bytes.

B 5.2 Fragmented services

There are four transfer fragments that are distinguished by the service byte:

– First fragment
– Middle fragment
– Last fragment
– Abort/error fragment

a. First fragment

Services without 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

Service byte 1, definition of the first 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 = First fragment	
6	0 (zero)	00 = First 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 request of the user. The processing time for this service depends on the PCP channel size, the number of Inline modules, and the actual service requested. This bit provides information on 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}- $01F_{hex}$)	Number = 1–$1F_{hex}$ fragment number; if more fragments are needed the fragment number should be switched to 0
5	1	Bits 5 and 6 combined, determine fragment type: 1 = Middle fragment
6	0	Bits 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 byte 1, 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 service-specific PCP error has occurred. See the Error class and Error code bytes.
2	PCP channel busy	A PCP transaction is already in progress, e.g., 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 fragment received after last fragment, middle fragment 2 received before fragment 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 response

PCP error response without 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 without 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 without 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**	**Subin-dex**	**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**	**Subin-dex**	**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 during read process.

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**	**Subin-dex**	**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 bus coupler

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. The user should change to the 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 = Communication error between PLC and bus coupler during PCP write process.

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**	**Subin-dex**	**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

Communication error: Communication re-established

Continue as normal since PCP request is buffered on the bus coupler 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}

C Default PDO mapping

C 1 Introduction

If a device supports a specific type of I/O function (analog/digital I/Os) it also supports the associated default PDOs. However, the module can support additional manufacturer-specific PDOs. If variable PDO mapping is supported the default PDO settings can be changed by means of configuration. For standard mapping, up to four approved TPDOs and four approved RPDOs are available. If a module does not support a specific I/O function, the associated default PDO also remains unused. If a device supports more than the default digital input or output channels, the associated analog default PDO remains unused. The additional digital I/O modules use additional PDOs. This also applies to additional analog channels.

All TPDOs with transmission method 255 are transmitted when entering the operational state.

Default mapping is implemented according to CiA 301.

C 2 PDO mapping

C 2.1 First RPDO mapping (digital outputs)

This RPDO asynchronously receives the values from a maximum of 64 digital outputs at one I/O module. The default transmission method is 255. The default values of the mapped outputs are described in the Default state objects.

These default objects are valid after power-up and after a reset.

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 objects mapped to the PDO depends on the hardware.

C 2.2 First TPDO mapping (digital inputs)

This TPDO transmits the values of a maximum of 64 digital inputs in an event-driven manner. The default transmission method is 255. The default values for the disable and event timers are 0. If the value of a digital output changes, this PDO is transmitted immediately. If an interrupt mask is enabled, the PDO is only transmitted 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 objects mapped to the PDO depends on the hardware.

C 2.3 Second RPDO mapping (analog outputs)

This RPDO asynchronously receives the 16-bit values of a maximum of four analog outputs at one I/O module. The default transmission method is 255. The default values of the mapped outputs are described in the Default state objects.

These default objects are valid after power-up and after a reset.

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 objects mapped to the PDO depends on the hardware.

C 2.4 Second TPDO mapping (analog inputs)

This TPDO transmits the 16-bit values of a maximum of four analog inputs in an event-driven manner. The default transmission method is 255. The default values for the disable and event timers are 0. By default, the interrupt source (object 6423_{hex}) is disabled. If the value of an analog input changes and object 6423_{hex} is enabled, the PDO is transmitted immediately. If the analog interrupt condition is enabled, the PDO is only transmitted if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO is transmitted if one of these interrupt 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 objects mapped to the PDO depends on the hardware.

C 2.5 Third RPDO mapping (analog outputs)

This RPDO asynchronously receives the 16-bit values of a maximum of four analog outputs at one I/O module. The default transmission method is 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 objects mapped to the PDO depends on the hardware.

C 2.6 Third TPDO mapping (analog inputs)

This TPDO transmits the 16-bit values of a maximum of four analog inputs in an event-driven manner. The default transmission method is 255. By default, the interrupt source (object 6423_{hex}) is disabled. If the value of an analog input changes and object 6423_{hex} is enabled, the PDO is transmitted immediately. If the analog interrupt condition is enabled, the PDO is only transmitted if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO is transmitted if one of these interrupt 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 objects mapped to the PDO depends on the hardware.

C 2.7 Fourth RPDO mapping (analog outputs)

This RPDO asynchronously receives the 16-bit values of a maximum of four analog outputs at one I/O module. The default transmission method is 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 objects mapped to the PDO depends on the hardware.

C 2.8 Fourth TPDO mapping (analog inputs)

This TPDO transmits the 16-bit values of a maximum of four analog inputs in an event-driven manner. The default transmission method is 255. By default, the interrupt source (object 6423_{hex}) is disabled. If the value of an analog input changes and object 6423_{hex} is enabled, the PDO is transmitted immediately. If the analog interrupt condition is enabled, the PDO is only transmitted if the interrupt condition is fulfilled. If more than one interrupt condition is enabled, the PDO is transmitted if one of these interrupt 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 objects mapped to the PDO depends on the hardware.

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

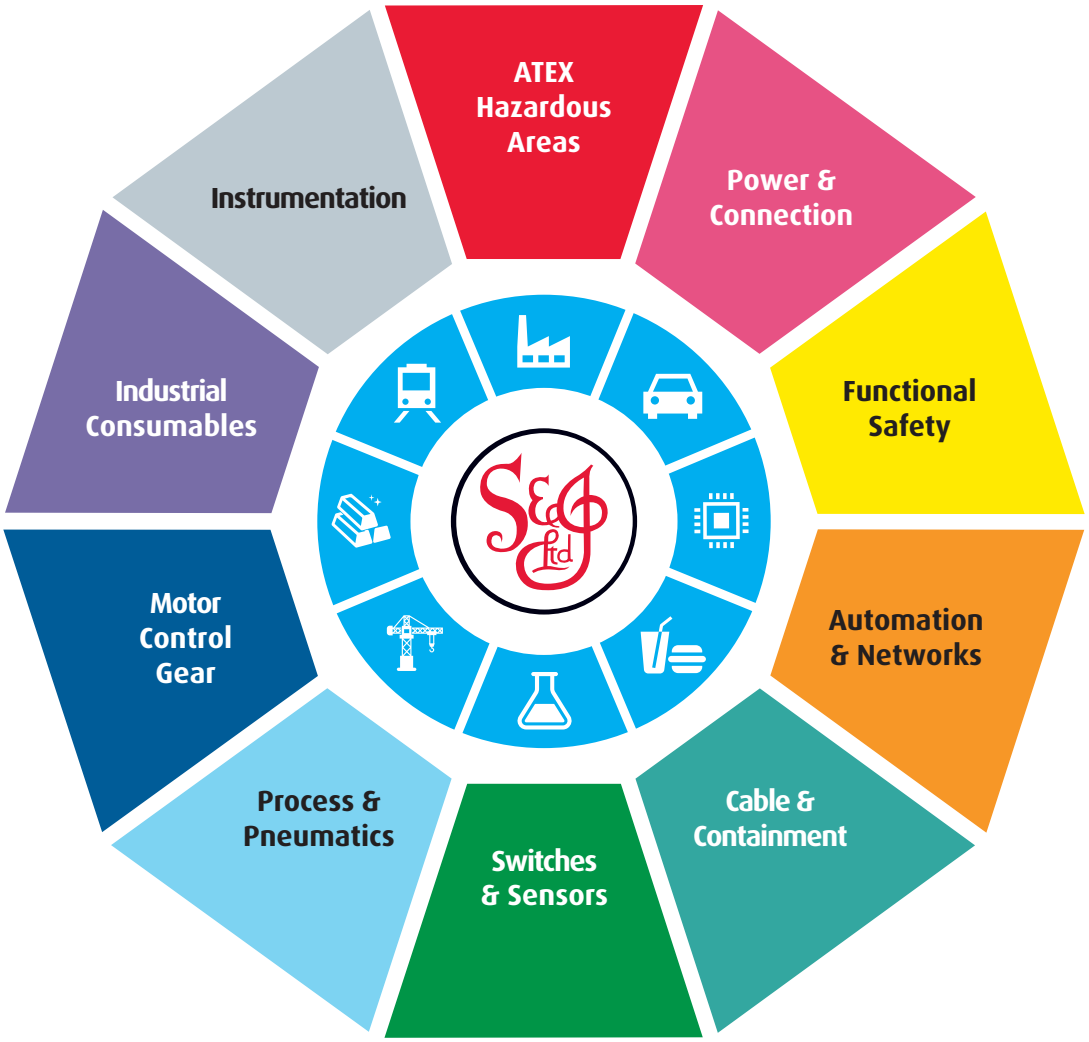
At Scattergood & Johnson Ltd, we pride ourselves on being a technical distributor to specialist industries.

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