EZ221-CO CANopen Slave Interface

User Manual

May 2005

Warning!
Dangerous electrical voltage!

Before commencing the installation

- Disconnect the power supply of the device.
- Ensure that devices cannot be accidentally restarted.
- Verify isolation from the supply.
- Short circuit to earth.
- Cover or enclose neighboring units that are live.
- Follow the engineering instructions (AWA) of the device concerned.
- Only suitably qualified personnel in accordance with EN 50110-1/-2 (VDE 0105 Part 100) may work on this device/system.
- Before installation and before touching the device ensure that you are free of electrostatic charge.
- The functional earth (FE) must be connected to the protective earth (PE) or to the potential equalization. The system installer is responsible for implementing this connection.
- Connecting cables and signal lines should be installed so that inductive or capacitive interference does not impair the automation functions.
- Install automation devices and related operating elements in such a way that they are well protected against unintentional operation.
- Suitable safety hardware and software measures should be implemented for the I/O interface so that a line or wire breakage on the signal side does not result in undefined states in the automation devices.
- Ensure a reliable electrical isolation of the low voltage for the 24 volt supply. Only use power supply units complying with IEC 60364-4-41 (VDE 0100 Part 410) or HD 384.4.41 S2.
- Deviations of the mains voltage from the rated value must not exceed the tolerance limits given in the specifications, otherwise this may cause malfunction and dangerous operation.
- Emergency stop devices complying with IEC/EN 60204-1 must be effective in all operating modes of the automation devices.Unlatching the emergency-stop devices must not cause restart.

- Devices that are designed for mounting in housings or control cabinets must only be operated and controlled after they have been installed with the housing closed. Desktop or portable units must only be operated and controlled in enclosed housings.

- Measures should be taken to ensure the proper restart of programs interrupted after a voltage dip or failure. This should not cause dangerous operating states even for a short time. If necessary, emergency-stop devices should be implemented.

- Wherever faults in the automation system may cause damage to persons or property, external measures must be implemented to ensure a safe operating state in the event of a fault or malfunction (for example, by means of separate limit switches, mechanical interlocks etc.).

Contents

About This Manual

Target group

This manual has been produced for automation technicians and engineers. A thorough knowledge of the CANopen fieldbus and the programming of a CANopen master is required. You should also be familiar with the operation of the EZ control relay or the EZD multi-function display.

Further manuals for this device

The following operating manuals should be followed:

- EZ500/700 Series (MN05013003E)
- EZ800 Series (MN05013004E)
- EZD Series (MN05013005E)

All manuals are available on the Internet for download as PDF files. For a fast search enter the documentation number as the search criterion at www.EatonElectrical.com.

References

[1] CANopen – Application Layer and Communication Profile
CiA DS 301

[2] CANopen – Cabling and Connector Pin Assignment
CiA DR 303-1

[3] CANopen – Indicator Specification
CiA DR 303-3

[4] CANopen – Layer Setting Services and Protocol (LSS)
CiA DSP 305

References are available on the Internet at www.can-cia.org.

Data types

The CANopen specifies its own data types. Reference CiA document [1] for details. The data types listed in the following table are used for the CANopen protocol handler of the EZ221-CO.

Name	Description	Range	
		Minimum	Maximum
UNSIGNED8	8-bit unsigned integer (b7 to b0)	0	255
UNSIGNED16	16-bit unsigned integer (b15 to b0)	0	65535
UNSIGNED32	32-bit unsigned integer (b31 to b0)	0	4294967295
VISIBLE_STRINGlen	Character string of the length len. The character string does not have to be delimited with 0_{hex}!	All ASCII characters from 20_{hex} to $7E_{hex}$ and 0_{hex} are permissible	
DOMAIN	User-specific data format		

Device designation

This manual uses the following short names for equipment types, as far as the description applies to all of these types:

- EZ512-..-..., EZ7..-..-...

Type designation of the control relay, the point represents a placeholder for all characters used

- EZ500 for
  - EZ512-AB...
  - EZ512-AC
  - EZ521-DA…
  - EZ512-DC

- EZ700 for
  - EZ719-AB…
  - EZ719-AC…
  - EZ719-DA…
  - EZ719-DC…
  - EZ721-DC…

- EZ800 for
  - EZ819-...
  - EZ820-...
  - EZ821-...
  - EZ822-...

- EZD-CP8.. for
  - EZD-CP8-ME
  - EZD-CP8-NT

- EZ-AB for
  - EZ512-AB...
  - EZ719-AB...
- EZ-AC for
  - EZ512-AC-..
  - EZ719-AC
  - EZ8..-AC-...
- EZ-DC for
  - EZ512-DC-..
  - EZ719-DC-..
  - EZ8..-.DC-...
- EZ-DA for
  - EZ512-DA...
  - EZ719-DA...

Abbreviations and symbols

Meaning of abbreviations and symbols used in this manual:

BCD	Binary Coded Decimal code
CAL	CAN Application Layer
CAN	Controller Area Network
COB	Communication Object
COB ID	Communication Object Identifier
COV	Change of Value
DEC	Decimal (number system with base 10)
EDS	Electronic Data Sheets
EMCY	Emergency Object
HEX	Hexadecimal (number system with base 16)
ID	Identifier
LSS	Layer Setting Service
NMT	Network Management
NVM	Non-Volatile Memory
NVM-PA	Non-Volatile Memory Parameter (load and save area)
NVM-RO	Non-Volatile Memory-Read Only (read-only memory area)
PC	Personal Computer
PDO	Process Data Object
ro	Read Only (read access only)
ROM	Read Only Memory
RTR	Remote Transmit Request
rw	Read/Write (read/write access)
SELV	Safety Extra Low Voltage
SDO	Service Data Object
FS	Factory Setting

Writing conventions

Except for the first page of chapters and empty pages at the end, the top left of the page shows the chapter title and the top right of the page shows the current section for greater clarity.

▶ indicates actions to be taken.

Attention!
Warns of a hazardous situation that could result in damage to the product or components.

Caution!
Warns of the possibility of serious damage and slight injury.

Warning
Warns of the possibility of a hazardous situation that could result in major damage and serious or fatal injury or even death.

→ Indicates interesting tips and additional information

1 The EZ221-CO

The EZ221-CO communication module was developed for automation tasks that use the CANopen fieldbus. The EZ221-CO is a gateway and can only be used in conjunction with the EZ700, EZ800 or EZD basic units. The system unit, consisting of the EZ/EZD control unit and the CANopen gateway functions in the fieldbus system exclusively as a slave station.

System overview

The EZ CANopen slaves are integrated into a CANopen.

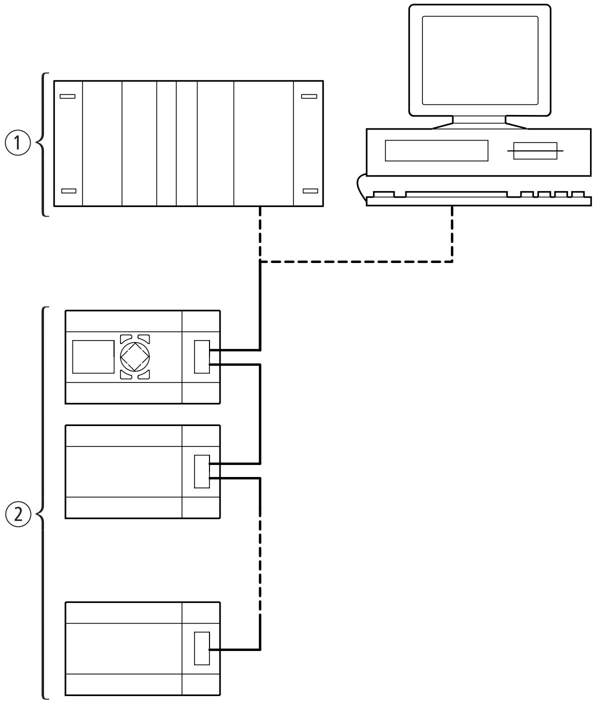


Figure 1: Integration of EZ221-CO in the CANopen network

① Master area, PLC or PC with CAN card

② Slave area, e.g.: EZ control relay with CANopen gateway

Setup of the unit

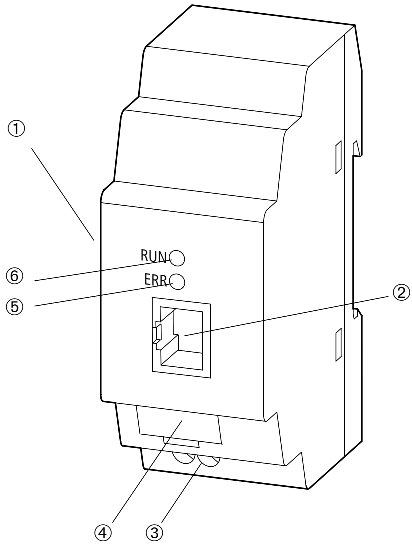


Figure 2: Setup of the EZ221-CO

① EZ-LINK socket

② CANopen terminal, 8-pin RJ45 socket

③ Power supply 24 V H

④ Device designation plate

⑤ ERR LED (Error)

⑥ RUN LED

Device function description

The EZ221-CO module allows the EZ and EZD series devices to be connected to a CANopen communication network. The following data can be transferred by selecting the appropriate SDO/PDO:

EZ700/800, EZD-CP8..

- S1 to S8
  Output data of the basic unit, RUN/STOP
  (read, as viewed from CANopen master)
- R1 to R16
  Input data of the basic unit, RUN/STOP
  (write, as viewed from CANopen master)
- All function relay data
  (read, as viewed from the CANopen master)
  - Timing relays
  - Counter relays
  - Time switches
  - Analog comparators
  - Weekday, time, summer/winter time (DST)
- The setpoints of the function relays
  (write, as viewed from CANopen master)
  - Timing relays
  - Counter relays
  - Time switches
  - Analog comparators
  - Weekday, time, summer/winter time (DST)

EZ800/EZD-CP8..

- All markers and EZ-NET data
- Function blocks
  (read/write, as viewed from the master)
  - Arithmetic function blocks
  - Frequency counters, high-speed counters, incremental encoder counters
  - 7-day and year time switch
  - Operating hours counters
  - PID controllers
  - PWM (pulse width modulation)
  - Real-time clock

Improper use

EZ may not be used to replace safety-relevant control circuits, e.g.:

- Burner,
- Emergency-stop,
- Crane or
- Two-hand safety controls.

2 Installation

The same principles apply as for EZ700, EZ800 and EZD basic units with expansion devices.

Connecting the EZ221-CO to the basic unit

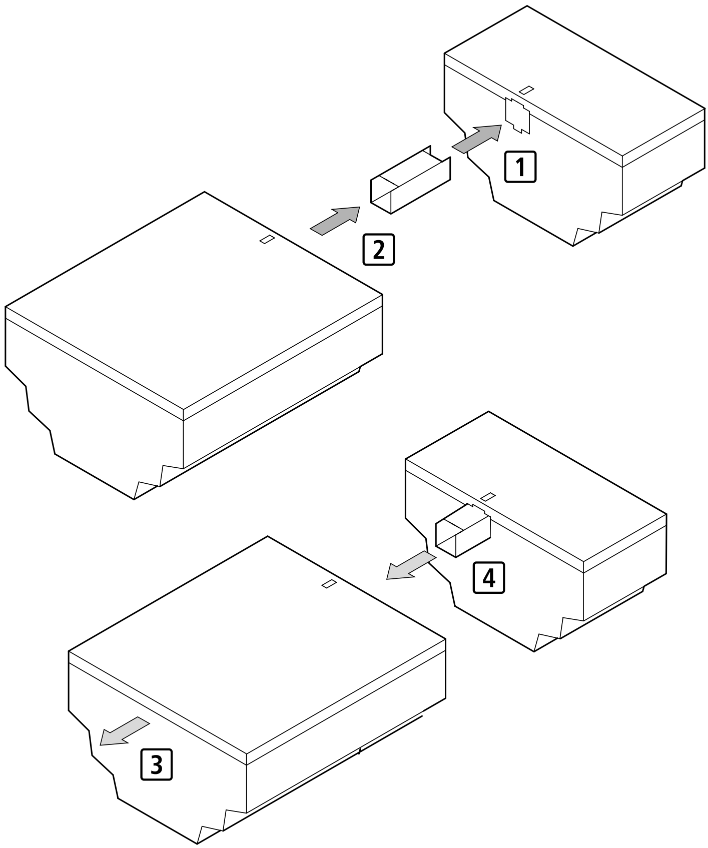


Figure 3: Fitting the EZ221-CO to the basic unit

1 + 2 Fitting

3 + 4 Removal

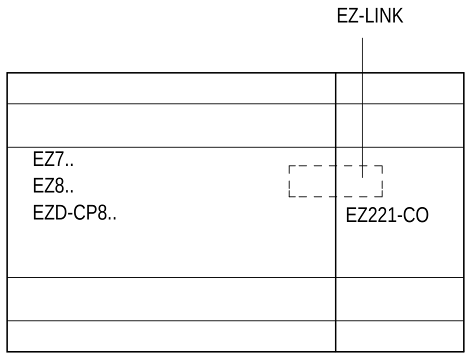


Figure 4: Connection between the basic unit and EZ221-CO

Connecting the power supply

The EZ221-CO device is run on a 24 V DC power supply (→ Technical data under “Power supply”, page 219).

Warning
Always ensure electrical safety isolation between the extra low voltage (SELV) and the 24 V power supply.

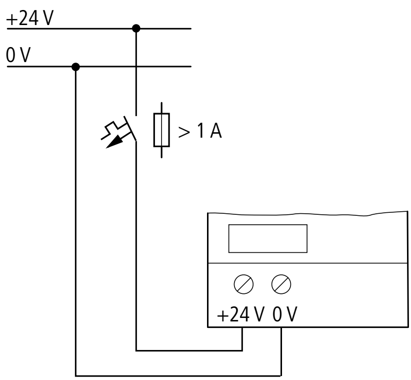


Figure 5: Power supply of the EZ221-CO

Connecting CANopen

The cable types, coupling connectors and terminating resistors to be used are specified in ISO 11898.

A shielded 8-pin RJ45 plug is used to connect the EZ221-CO. The pin assignment of the plug is specified below in accordance with CiA DR-303-1.

Pin assignment of the CANopen

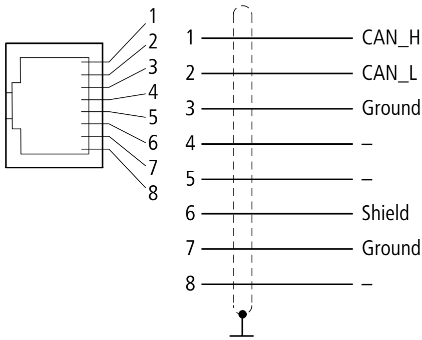


Figure 6: Pin assignment of the device socket

Pin	Signal	Description
1	CAN_H	CAN bus signal (dominant high)
2	CAN_L	CAN bus signal (dominant low)
3, 7	CAN_GND	CAN ground
6	CAN_SHILD	Optional shielding
4, 5, 8	–	n.c.

Bus terminating resistors

The first and last node of a CANopen network must be terminated by means of a 120 Ω bus terminating resistor. This is interconnected between the CAN_H and CAN_L terminals.

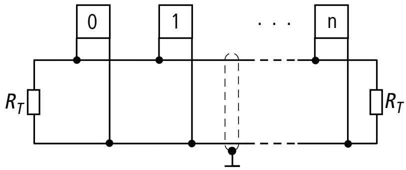


Figure 7: Terminating resistors R_T: CAN_H and CAN_L terminals

R_T = 120 Ω

EMC compliant wiring

Electromagnetic interference may lead to unwanted effects on the communication fieldbus. Such effects can be significantly reduced by using the cable described above, a shielded RJ45 connector and by terminating the shield.

The two figures below show the correct termination of the shielding.

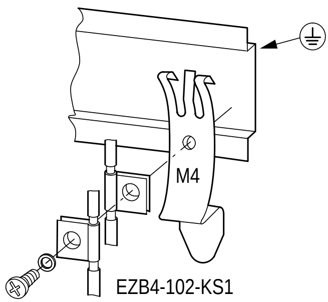


Figure 8: Shielding connection to the mounting rail

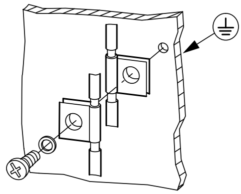


Figure 9: Shielding connection to the mounting plate

Electrical isolation

The following potential isolation must be provided for the interfaces of the EZ221-CO:

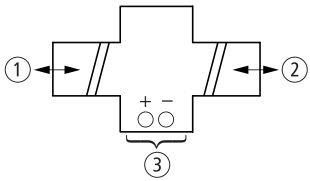


Figure 10: Potential isolation between the power supply and outputs

① Safe electrical isolation between EZ-LINK and 240 V AC

② Basic electrical isolation to the CANopen communication bus

③ Power supply 24 V DC

Transfer rates – automatic baud rate recognition

After it is switched on, the EZ221-CO module automatically detects the data transfer rate of the communication network. However, this requires that at least one station is transmitting valid telegrams in the network. The fast flashing Error and RUN LEDs of the EZ221-CO indicate this status.

After a correct CANopen message frame has been received, the used and thus set baud rate is considered correct and the device transmits a BootUp message frame. The RUN LED starts flashing and the ERR LED will be switched off.

The EZ221-CO supports the data transfer rates specified by CiA. The table below provides an overview of recommended bit rates and of the corresponding maximum cable lengths.

Bit rate kbps	Max. cable length m	Recommended conductor cross-section mm^2
10	5000	> 0.8
20	2500	> 0.8
50	1000	0.75 to 0.8
100	650	0.34 to 0.6
125	500	0.34 to 0.6
250	250	0.34 to 0.6
500	100	0.25 to 0.34
800	50	0.25 to 0.34
1000	25	0.25 to 0.34

3 Device Operation

Initial power up

- ▶ Before you switch on the device, verify that it is properly connected to the power supply, to the bus connector and to the basic unit.
- ▶ Switch on the power supply to the basic unit and the EZ221-CO.

The LEDs of the EZ221-CO will flicker. The device is in the mode for determining the correct baud rate (→ section “Transfer rates – automatic baud rate recognition” on page 24). The GW message (intelligent station connected) must be displayed on the basic unit.

Basic unit	GW display
EZ700	Flashing
EZ800	Flashing
EZD-CP8..	Flashing

As soon as the device is switched to Operational status, the GW message is static in the display, even on the devices with a flashing GW, → section “Network management” on page 38).

If the EZ221-CO has its default settings (node ID = 127), you need to define the CANopen slave address.

Setting the CANopen slave address

Each CANopen slave must be assigned a unique address (node ID) within the CANopen structure. You can assign a maximum of 127 addresses (1 to 127) within a CANopen structure. All node IDs must be unique within the entire bus structure.

There are three ways to set the CANopen address of an EZ221-CO:

- Using the integrated display and keypad on the EZ or EZD basic unit; address range: 1 to 127
- Using EZSoft on the PC
- Via the configuration software of the master PLC used (possibly by means of an explicit message).

Setting the address on the basic unit with display

Requirements:

- The appropriate basic unit (EZ700, EZ800 or EZD) and EZ221-CO must be fed with power.
- The basic unit must have been unlocked (no password activated).
- The basic unit must have a valid operating system version.
- The basic unit must be in STOP mode.

▶ Press the DEL + ALT buttons to change to the special menu.

```
PASSWORD...
SYSTEM...
GB D F E I
CONFIGURATOR
```

```
PASSWORD...
SYSTEM...
GB D F E I
CONFIGURATOR
```

▶ Use the cursor buttons ∧ or ∨ to change to CONFIGURATOR.

▶ Confirm with OK.

```
NET...
LINK...
```

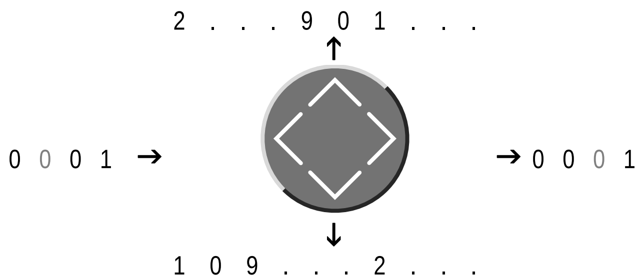
▶ With EZ800/EZD devices select LINK...

▶ Confirm with OK.

The CANOPEN menu appears.

```
CANOPEN
(MAX 127)
NODE ID 0127
221 -01.20- B
```

▶ Set the address with the cursor buttons:
- Set the current numeric value with the ∧ or ∨ buttons.
- You can change the actual numeric value via the < or > buttons.



▶Press OK to accept the address.

▶Press ESC to cancel address entry.

Information on the 4th display line:

```
xxx-xx.xx-xx
221-02.10- B
```

- B — Hardware version, Index: b
- 02.10 — Software version, OS version: 2.1
- 221 — Device designation: EZ221-CO

Setting the address by means of EZSoft

‹Menu → Communication → Configuration → Expansion Devices → EZ221-CO›.

→ The menu is only available in Communication View, therefore activate the Communication tab.

→ EZ221-CO devices accept the address automatically.

Setting the address via special configuration tools

A further option of setting or modifying the node ID of the gateway is provided by special configuration tools, which can be used for general configuration of the CANopen network. The gateway supports the LSS (Layer Setting Services) service accordingly.

Status LEDs

The EZ221-CO expansion unit is equipped with two LEDs: one green RUN LED and one red ERR LED. These indicate the current module status and allow quick error analysis.

Error LED

No.	Error LED	Status	Description
1	OFF	No error	The EZ221-CO is operating error-free. If the RUN LED is also off, the EZ221-CO is either switched off or is currently being reset.
2	Single flash	Alarm limit reached	At least one of the error counters of the CANopen Controller has either reached or exceeded the Warning Limit. Too many errors have occurred on the CANopen bus.
3	Flickering	AutoBaud/LSS	Auto baud rate detection is currently busy (flickers in alternation with the RUN LED).
4	Flashes twice	Error control event	A protective Guard Event or a Heartbeat Event has occurred.
5	ON	Bus-off	The CANopen controller has changed to BUS-OFF status.

RUN LED

No.	RUN LED	Status	Description
1	OFF	Reset	The EZ221-CO is either switched off or is currently being reset.
2	Flickering	AutoBaud	Auto baud recognition is currently busy (LED flickers, in alternation with the ERR LED).
3	Single flash	STOPPED[1)]	The device is in STOPPED state.
4	Flashing	PRE-OPERATIONAL[1)]	The device is in PRE-OPERATIONAL status.
5	ON	OPERATIONAL[1)]	The device is in OPERATIONAL status.

1) Detailed information on the various states is provided in section “Network management”, page 38.

Timing diagram of the ERR and RUN LEDs

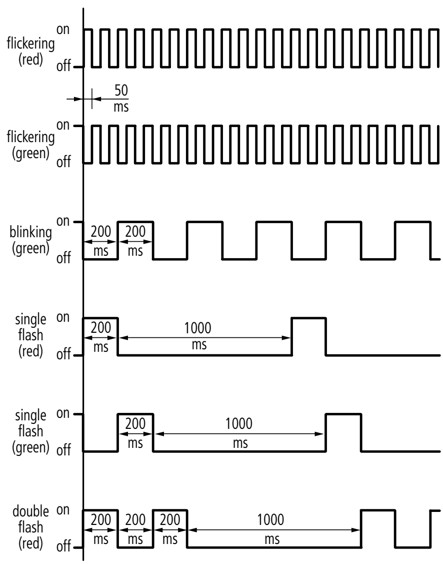


Figure 11: ERR and RUN LED

Cycle time of the EZ basic unit

Communication between the basic unit and the EZ221-CO via EZ-LINK increases the cycle time of the basic unit.

The worst case value is 25 ms.

Please take this factor into account when calculating response times of the basic unit.

EDS file

You can implement the EZ221-CO into the CANopen structure by means of a standardized EDS file (Electronic Data Sheet). The EDS defines the CANopen functions in machine code. It lists all objects, supported data transfer rates, the manufacturer and many other data.

The file “EZ221CO.eds” can be obtained at www.EatonElectrical.com. The file is also available on the EZSoft CD ROM.

4 CANopen Services

The functions for controlling the EZ221-CO on the CANopen bus are defined by the CANopen services.

Communication objects

The EZ221-CO supports service data objects (SDOs) and process data objects (PDOs) of the CANopen Predefined Connection Set.

Service data objects

Service data objects (SDO – Service Data Object) are used for read/write access to the entries of the object dictionary.

Server SDO

The system supports the first server SDO, which allows read/write access to the local object dictionary.

The EZ221-CO supports expedited transfer (of up to four data bytes) and segmented transfer (for more than four data bytes).

→ Block transfer is not supported!

More detailed information on the sequence is provided in section “PDO protocol”, page 59.

Client SDO

Client SDOs provide remote read/write access to the object dictionaries of CANopen devices on the network.

→ The EZ221-CO does not support client SDOs.

Process data objects

Process data is exchanged in the CANopen by means of PDOs (= Process Data Object). More detailed information on the sequence is provided in section “Manufacturer-specific objects”, page 54.

The table below lists the process data and the corresponding PDOs.

PDO	Process data	Length
Receive PDO	Command or identification for the image data R16 to R1 of EZ/EZD basic unit (output data to EZ)	3 bytes
Transmit PDO	Command or status for the image data S8 to S1 of EZ/EZD basic unit (input data from EZ)	3 bytes

→ For details on the structure of process data refer to section “Manufacturer-specific objects”, page 54.

Receive PDO

The EZ221-CO receives data from the CANopen network (PDO consumer) by means of receive PDOs and writes this data via EZ-LINK to the EZ/EZD basic unit as a command or identifier for the image data R16 to R1.

Transmit PDOs

In the opposite direction, the commands or status of the S8 to S1 image data of EZ/EZD are read via EZ-LINK and transmitted to the CANopen network as transmit PDOs of the EZ221-CO (PDO producer).

PDO mapping

The EZ221-CO supports **static PDO mapping**. The process data is here permanently assigned to the specific PDOs, with a granularity of 1 byte. The PDO mapping is permanently stored and cannot be modified by the user.

Transmission types of PDOs

Receive PDO:
The default transmission type setting for receive PDOs is "asynchronous" (Value: 255_{dec} = FF_{hex}).

Transmit PDO:
The default transmission type setting for transmit PDOs is "asynchronous" (Value: 255_{dec} = FF_{hex}).

Inhibit Time

The Inhibit Time is evaluated only for transmit PDOs. This time represents the data transfer inhibit time between two transmit PDOs, specified in steps of 100 µs. The passed value is rounded to the next lower millisecond. Values lower than 1 ms are stored as "0". In this case the module transfers the PDOs at maximum speed.

An Inhibit Time is not set by default, since data transferred via the EZ-LINK protocol is updated only at 180 ms intervals. However, the user can set an Inhibit Time definition for the transmit PDO as required.

Event Timed PDOs

The expiration of a counter can be considered as an event which triggers the transmission of a PDO. The EZ221-CO does not support Event Timed PDOs by default, however the user can enable this function for transmit PDOs as required.

Multiplexed PDOs

In addition to elementary process data, the multiplexed PDOs also contain address information consisting of an index and a subindex used for writing the PDO to a specific address in the object dictionary of the consumer device.

→ The EZ221-CO does not support multiplexed PDOs.

PDO mapping

Process data is mapped to a receive and transmit PDO as follows.

Receive PDO 1:
The table below shows the mapping of the first receive PDO.

Data byte	Contents	Description
Data byte 1	Cyclic command and identifier	Write input data of the EZ/EZD basic unit (from the point of view of the master) (index 2011, subindex 00_{hex})
Data byte 2	Image data R16 to R9	
Data byte 3	Image data R8 to R1	
Data bytes 4 to 8	Not transferred	

→ For details on the composition of the process data refer to section "Output data (2011_{hex}): operating mode, R1 – R16", page 74.

Receive PDO 2 to 4:
These receive PDOs of the Predefined Connection Set are not supported.

Transmit PDO 1:
The table below shows the mapping of the first transmit PDO.

Data byte	Contents	Description
Data byte 1	Cyclic command and status	Read output data of the EZ/EZD basic unit (from the point of view of the master) (index 2012_{hex}, subindex 00_{hex})
Data byte 2	Image data S8 to S1	
Data byte 3	empty (00_{hex})	
Data byte 4 to 8	Not transferred	

→ For details on the composition of the process data refer to section "Input data (2012_{hex}): operating mode, S1 – S8", page 77.

Transmit PDO 2 to 4:
These transmit PDOs of the Predefined Connection Set are not supported.

System services

Synchronization object

The EZ221-CO as consumer supports the synchronization object (index: 1005_{hex}) in order to enable the synchronous transfer of PDOs.

Time Stamp object

A time producer uses the time stamp object (Index: 1012_{hex}) to provide a common time reference to all system nodes. The EZ221-CO does not support the Time Stamp object.

Emergency object

The EZ221-CO supports the emergency object (index: 1014_{hex}) in order to report device errors to the network. The content of this emergency message is determined by the error event. Errors detected are described under section “Error messages (Emergency)”, page 55.

Network management

A CANopen network contains only one NMT master (NMT = Network Management), while all other devices are NMT slaves. The NMT master has full control over all units and can thus change their status.

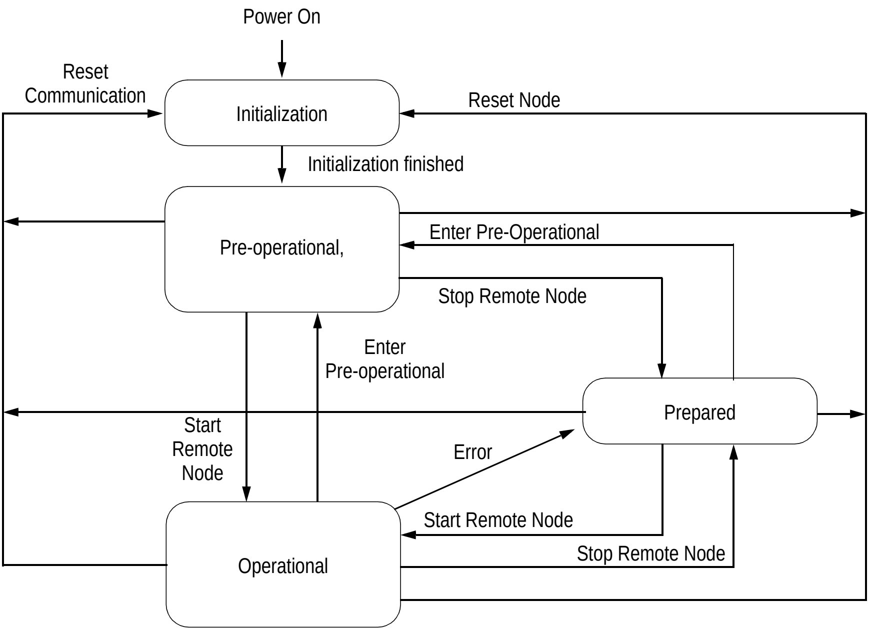


Figure 12: Network management

CANopen distinguishes between the following states:

- Initialization,
- Pre-operational,
- Operational and
- Prepared

Initialization

This is the status of a node after power on. Auto baud recognition, initialization of device applications and communication take place within this phase. The node automatically enters the next state, namely the Pre-operational state.

Pre-operational

In this mode it is possible to communicate with the node via SDOs (e.g setting the Guard Time, Lifetime Factor). The node is not able to execute PDO communication and does not transmit any emergency messages.

The RUN LED of the EZ221-CO flashes to indicate this state.

The state is indicated also on the basic unit with the flashing GW message in the EZ display. The diagnostics input I14 (on the basic unit) is set until GW is no longer flashing in the EZ display. This is achieved by setting the CAN node to Operational mode.

Operational

In this state, the CANopen node is fully ready for operation and can automatically transmit messages (PDOs, Emergency).

The RUN LED of the EZ221-CO is static to indicate this state.

The status is indicated also on the basic unit by the static display of the GW status message. The diagnostics input I14 (on the basic unit) is set to zero.

Prepared

In this state, the node connection is switched completely to bus-off state; neither SDO nor PDO communication are possible. The network status of a node can be changed only by means of an appropriate network command (e.g. the Start Remote Node service).

A Boot-Up message will be transmitted after power on of a device in order to indicate its ready state. This message frame uses the identifier of the NMT error control protocol and is permanently assigned to the set device address (1792_{dec} + device address).

→ For information on the PDO and SDO transfer refer also to section “PDO protocol”, on page 59.

Process data exchange by means of PDOs is enabled by setting the module to OPERATIONAL state via the Start Remote Node service. TxPDOs configured with transmission types 254 or 255 will be transmitted at each transition to OPERATIONAL state, irrespective of any changes in input data.

The module enters the PREPARED state after an error has occurred. Communication via SDOs and PDOs is then no longer possible and the module only responds to the NMT services:

- Start Remote Node, transition to OPERATIONAL state; making it possible to transfer data via SDOs and PDOs.
- Enter Pre-operational, transition to PRE-OPERATIONAL state; it is possible to transfer data via SDOs.
- Reset Node and
- Reset Communication, transition to INITIALIZATION state, i.e. the last settings will be loaded from memory, or the factory settings if nothing has been saved previously. The module then enters PRE-OPERATIONAL state.

Structure of the NMT services:

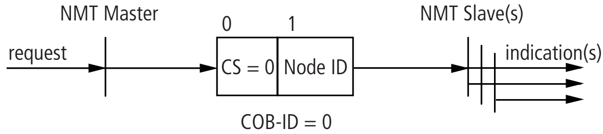


Figure 13: Structure of the NMT services, Start Remote Node

Node ID = 0: Sets all existing nodes to OPERATIONAL state.

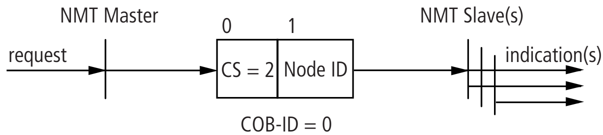


Figure 14: Structure of the NMT services, Stop Remote Node

Node ID = 0: Sets all existing nodes to PREPARED state.

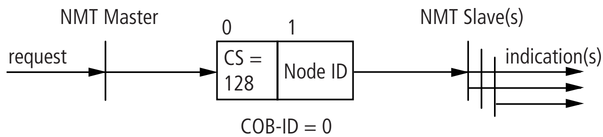


Figure 15: Structure of NMT services, PRE-OPERATIONAL state

Node ID = 0: Sets all existing nodes to PRE-OPERATIONAL state.

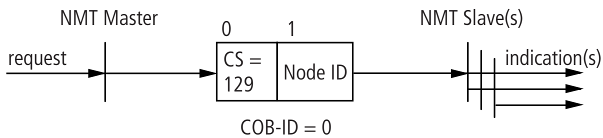


Figure 16: Structure of NMT services, Reset Node

Node ID = 0: Resets all existing nodes.

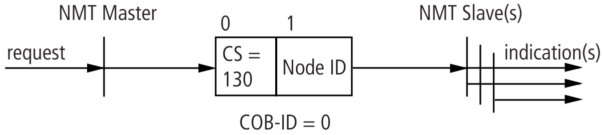


Figure 17: Structure of NMT services, Reset Communication

Node ID = 0: Resets all existing nodes.

Node monitoring

A CANopen node must be checked in particular if it does not continuously transmit messages (cyclic PDOs). Two methods can be used alternatively to monitor CANopen nodes.

→ EZ221-CO supports Node Guarding and Heartbeat Producer modes for node monitoring.

Node Guarding

The NMT master polls all NMT slaves at specified intervals (Node Guard Time) by means of a node-specific Remote Transmission Request message frame (RTR). The NMT slave responds to this request by transmitting its communication status. The NMT master reports a Node Guarding Event to its application if a node fails to respond to the RTR within the specific Node Life Time.

Failure of Node Guarding

Error events triggered after the Life Time has expired and a Node Guard frame has not been received from the EZ221-CO will be treated as communication error.

The R data for the EZ basic unit will be set to zero in this case. The ERR LED flashes twice to indicate Guarding failure.

When the Node Guarding protocol is resumed, the ERR LED will be switched off immediately and the outputs of the EZ basic unit can now receive PDO data again.

Heartbeat Producer

The EZ221-CO broadcasts a cyclic heartbeat frame to signal its communication status. If a responsible heartbeat consumer does not receive this heartbeat frame within the Heartbeat Consuming Time, its application will report a heartbeat error. The second parameter relevant to the heartbeat protocol is the Heartbeat Producer Time, which can be set in the EZ221-CO gateway. This time determines the interval between the transfer of two heartbeat frames by the node.

When the Heartbeat Producer Time is set to a value unequal to zero on the EZ221-CO node, the first heartbeat frame will be transmitted during the transition from the Initialization to the Pre-operational state. Concurrent use of both node monitoring methods is not allowed. The heartbeat protocol is used when the Heartbeat Producer Time is unequal to zero.

→ The EZ221-CO does not support the Heartbeat Consumer mode for receiving heartbeat frames of other CANopen devices.

Further services

Saving and restoring entries

The EZ221-CO supports the saving and restoring of the object dictionary entries 1000_{hex} to $1FFF_{hex}$ in the non-volatile memory (EEPROM or FRAM). In the object dictionary tables, this area is named NVM-PA, while manufacturer-specific entries are stored in the NVM-RO area.

Parameters are saved via the object 1010_{hex} (SAVE signature); this always includes all parameters.

The factory settings (FS) in the area 1000_{hex} to $1FFF_{hex}$ can be restored with the object 1011_{hex} (LOAD signature). This routine always restores all factory settings.

Layer Setting Service

The Layer Setting Service is used to configure the node ID via the CANopen network. The EZ221-CO supports this service for both of the specified slave modes Switch Mode Global and Switch Mode Selective.

→ Changes of the node ID will become directly effective on the EZ221-CO. To ensure that the correct node ID is displayed on the EZ basic unit as well (Configurator menu), you must switch on the coupling module again.

Device profile

In the extension of the CiA-DS-301 communication profile which describes the communication mechanisms between nodes, the CANopen uses so-called device profiles for the essential device classes. The device profiles describe the device functions. The EZ221-CO cannot be assigned to an existing device profile.

5 Object Dictionary

The object dictionary of the EZ221-CO contains the entries described below.

Communication parameters

A detailed description of the communication parameters is provided in the CiA specification (reference CiA document [1] for details.

The objects 1000_{hex}, 1001_{hex} and 1018_{hex} are required for all CANopen devices. All other objects are optional; the table below shows which of these are supported by the EZ221-CO.

The table below lists the object dictionary entries 1000_{hex} to 1018_{hex}.

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1000	00	Device Type	UNSIGNED32	ro ROM	00000000	CANopen device without device profile
1001	00	Error Register	UNSIGNED8	ro RAM		Error indication: 00_{hex} no error
1003	00	Pre-defined Error Field	UNSIGNED8	rw RAM	00	Error history
	01 to 10[1)]	Default Error Field	UNSIGNED32	ro RAM		Error description (→reference CiA document [1] for details)
1005	00	COB-ID SYNC Message	UNSIGNED32	rw NVM-PA	00000080	COB-ID of the SYNC object, device consumes the SYNC message

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1008	00	Manufacturer Device Name	VISIBLE_STRING2)	ro NVM-RO	454153 593232 312D43 4F	Device name of the module (EZ221-CO)
1009	00	Manufacturer Hardware Version	VISIBLE_STRING8	ro NVM-RO	0001.000 (Example)	Hardware version of the module
A 100	00	Manufacturer Software Version	VISIBLE_STRING8	ro NVM-RO	0001.001 (Example)	Software version of the module
C 100	00	Guard Time	UNSIGNED16	rw NVM-PA	00 00_{hex} Resolution in 1 ms	Guard Time in milliseconds
100D	00	Life Time Factor	UNSIGNED8	rw NVM-PA	00_{hex}	Multiplier for the Guard Time, the result is equivalent to the maximum interval between the transfer of two Guarding message frames
1010	00	Store Parameters	UNSIGNED8	ro ROM	01	Max. number of storing options
	01	SAVE all Parameters	UNSIGNED32	rw RAM	wr: 65766173 rd: 00000001	→ reference CiA document [1] for details)

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1011	00	Restore default Parameters	UNSIGNED8	ro ROM	01	Loads the default parameters
	01	LOAD all Parameters	UNSIGNED32	rw RAM	wr: 64616F6C rd: 00000001	The device restores factory set parameters. These parameters are retained until the next power on event (→ reference CiA document [1] for details)
1014	00	COB-ID EMCY Message	UNSIGNED32	ro ROM	00000080 + node ID	CAN identifier of the emergency message
1015	00	Inhibit Time EMCY	UNSIGNED16	rw NVM-PA	0000 Resolution in 100 µs	Time interval between the transmission of two EMCY messages
1017	00	Producer Heartbeat Time	UNSIGNED16	rw NVM-PA	0000 Resolution in 1 ms	Time interval between the transmission of two heartbeat messages

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1018	00	Identity Object	UNSIGNED8	ro NVM-RO	04	General device information
	01	Vendor ID	UNSIGNED32	ro NVM-RO	00000003	Manufacturer
	02	Product Code	UNSIGNED32	ro NVM-RO	0323353	Product number
	03	Revision Number	UNSIGNED32	ro NVM-RO	00010001 (Example)	Version
	04	Serial Number	UNSIGNED32	ro NVM-RO	4010016 (Example)	Serial number

1) The EZ221-CO supports up to 16 entries in the error log.

2) The maximum string length is 31 characters, including the delimiter "\0".

The EZ221-CO supports the first server SDO of the Predefined Connection Set. The table below shows the object dictionary entry 1200_{hex}: Server SDO parameters of the first server SDO.

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1200	00	Server SDO Parameter	UNSIGNED8	ro ROM	02	Number of valid subindexes
	01	COB-ID Client → Server (rx)	UNSIGNED32	ro ROM	00000600 + node ID	COB-ID of the RxSDO. The ID is derived from the Predefined Connection Set.
	02	COB-ID Server → Client (tx)	UNSIGNED32	ro ROM	00000580 + node ID	COB-ID of the TxSDO. The ID is derived from the Predefined Connection Set.

The EZ221-CO supports the first receive SDO of the Predefined Connection Set. The receive PDOs 2 to 4 are not supported. The table below shows the object dictionary entry 1400_{hex}: Communication parameters of the first receive PDO.

Index hex	Subin dex hex	Object name	Data type	Access location	FS hex	Meaning
1400	00	Receive PDO Parameter	UNSIGNED8	ro NVM-PA	02	Communication parameter of the first RxPDO, number of valid subindexes
	01	COB-ID	UNSIGNED32	rw NVM-PA	00000200 + node ID	COB ID of the first Rx PDO, reference CiA document [1] for details)
	02	Transmission Type	UNSIGNED8	rw NVM-PA	FF	PDO transmission type: asynchronous

With the first receive PDO, the output data is stored in the object dictionary (index 2011_{hex}, subindex 00_{hex}) and is transferred by means of a standard protocol to the basic unit via EZ-LINK. The table below shows the object dictionary entry 1600_{hex}: Mapping parameters of the first receive PDO.

Index hex	Subin dex hex	Object name	Data type	Access location	FS hex	Meaning
1600	00	Receive PDO Mapping	UNSIGNED8	ro ROM	01	Mapping parameters of the first Rx PDO, number of valid subindexes
	01	Mapped Object 1	UNSIGNED32	ro ROM	2011001	Index 2011_{hex}, subindex 00_{hex}, length = 24 bits

The EZ221-CO supports the first Transmit PDO of the Predefined Connection Set. The transmit PDOs 2 to 4 are not supported. The table below shows the object dictionary entries 1800_{hex}: Communication parameters of the first Transmit PDO.

Index hex	Subi ndex hex	Object name	Data type	Access location	FS hex	Meaning
1800	00	Transmit PDO Parameter	UNSIGNED8	ro NVM-PA	05	Communication parameters of the first TxPDO. Number of valid subindexes
	01	COB-ID	UNSIGNED32	rw NVM-PA	00000180 + node ID	COB identifier, reference CiA document [1] for details)
	02	Transmission Type	UNSIGNED8	rw NVM-PA	FF	PDO transmission type: asynchronous
	03	Inhibit Time	UNSIGNED16	rw NVM-PA	0000	Inhibit time (min. time interval between the next transmission of a PDO) in ms 0000_{hex} = transmit now
	05	Event Timer	UNSIGNED16	rw NVM-PA	0000	Event counter 0000_{hex} = not used

With the first TxPDO, the input data is retrieved from the object dictionary (index 2012_{hex}, subindex 00_{hex}) and transferred after the first RxPDO has been received. The table below shows the object dictionary entry $1A00_{hex}$: Mapping parameters of the first Transmit PDO.

Index hex	Subindex hex	Object name	Data type	Access location	FS hex	Meaning
1A00	00	Transmit PDO Mapping	UNSIGNED8	ro ROM	01	Mapping parameters of the first TxPDO, number of valid subindexes
	01	Mapped Object 1	UNSIGNED32	ro ROM	20120018	Index 2012_{hex}, subindex 00_{hex}, length = 24 bits

Manufacturer-specific objects

In addition to the device profile objects, the object dictionary also contains the definitions of manufacturer-specific objects. The area between index 2000_{hex} and $5FFF_{hex}$ in the object dictionary of the EZ221-CO is reserved for these objects. The table below lists the corresponding manufacturer-specific objects used.

Index hex	Sub-index hex	Object name	Data type	Access location	Mapp able	FS hex	Meaning
2001[1)]	00	Coupling error	UNSIGNED8	ro EZ	No	–	Error status of the EZ221-CO
2002[1)]	00	EZ error	UNSIGNED8	ro EZ	No	–	Error status of the EZ/EZD basic unit
2011	00	Output data	UNSIGNED24	rw EZ	Yes	140000	Output data to the EZ/EZD basic unit
2012	00	Input data	UNSIGNED24	ro EZ	Yes	–	Input data from the EZ/EZD basic unit
2020	00	Status	UNSIGNED8	ro EZ	No	FF	Status 00_{hex} = valid data, 01_{hex} = invalid data, FF_{hex} = Initialization
2021	00	Command	DOMAIN Length = 7	rw EZ	No	–	Not used
2022	00	Response	DOMAIN Length = 7	ro EZ	No	–	Not used
3020	00	Status	UNSIGNED8	ro EZ	No	FF	Status 00_{hex} = valid data, 01_{hex} = invalid data, FF_{hex} = Initialization

Index	Sub-index	Object name	Data type	Access location	Mappable	FS	Meaning
hex	hex					hex	
3021	00	Command					
			Length = 8	EZ/EZD			EZ700/800, EZD-CP8..
3022	00	Response	DOMAIN	ro	No	–	Response
			Length = 8	EZ/EZD			from EZ700/800, EZD-CP8..

1) These two entries are also transmitted via the emergency message frame in the first two bytes of the Manufacturer Specific Error Field → section "Error messages (Emergency)".

Error messages (Emergency)

The EZ221-CO supports the defined generic error (1000_{hex}) described in the table below. It is triggered when the Generic Error bit 0 is set in the error register (index 1001_{hex}, subindex 00_{hex}).

In the manufacturer-specific error entry (Manufacturer Specific Error Field), byte 0 outputs the error code of the EZ221-CO (index 2001_{hex}, subindex 00_{hex}), and byte 1 outputs the error code of the connected EZ/EZD (index 2002_{hex}, subindex 00_{hex}).

Data byte	Contents	Value	Description
1	Generic Error Code	1000_{hex}	Generic Error (→ reference CiA document [1] for details
2			
3	Error Register	01_{hex}	Error register (index 1001_{hex}, subindex 00_{hex})
4	Manufacturer Specific Error Field (0)	xx_{hex}	Coupling error (index 2001_{hex}, subindex 00_{hex})
5	Manufacturer Specific Error Field (1)	00_{hex}	EZ error (index 2002_{hex}, subindex 00_{hex})

Data byte	Contents	Value	Description
6	Manufacturer Specific Error Field (2)	00_{hex}	not used
7	Manufacturer Specific Error Field (3)	00_{hex}	not used
8	Manufacturer Specific Error Field (4)	00_{hex}	not used

The last 16 errors are stored in the Predefined Error Field 1003_{hex} of the object dictionary and can be retrieved via server SDO. Format of entries in the Standard Error Fields (Subindex 01_{hex} to 10_{hex}):

Data byte	Contents	Value	Description
1	Error Code	1000_{hex}	Generic Error (→ reference CiA document [1] for details
2			
3	Additional Information	xx_{hex}	Coupling error (index 2001_{hex}, subindex 00_{hex})
4		00_{hex}	EZ error (index 2002_{hex}, subindex 00_{hex})

Third data byte: coupling module status

Value 00_{hex}
The EZ basic unit is connected to the EZ221-CO gateway via EZ-LINK.

Value 04_{hex}
The EZ basic unit is either switched off or is not connected to the EZ221-CO gateway via EZ-LINK.

6 CANopen Protocols

The following protocols are used for the transfer of data via the CANopen bus:

- PDO protocol for the transfer of I/O data and operating mode.
  Information on the data contents → chapter 7.
- SDO protocol for the transfer of control commands:
  - Date and time, summer/winter time
  - Read/write image
  - Read/write function blocks.

Information on data contents → chapter 8 (EZ700) and chapter 9 (EZ800/EZD).

- Emergency protocol
  Information on the data contents → page 55.

PDO protocol

The EZ221-CO by default uses the Write PDO Protocol as shown in the figure below. The Read PDO Protocol (not shown) can be called if required.

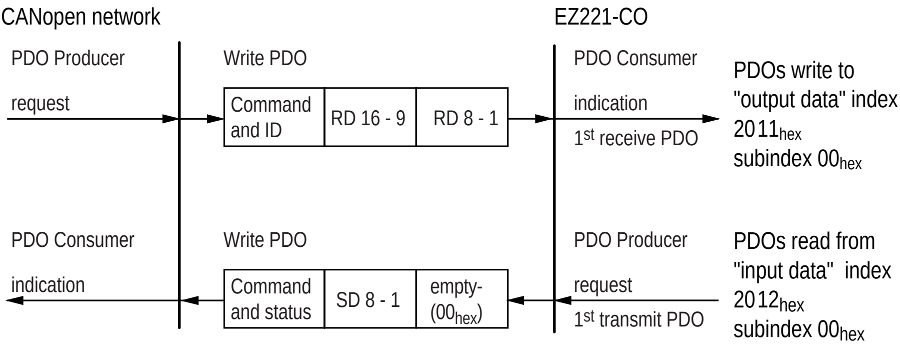


Figure 18: Write PDO Protocol

An indication informs the application that new data can be received via the first receive PDO and stored in the "output data" entry of the object dictionary (index 2011_{hex}, subindex 00_{hex}). The application then requests the transmission of data from the "input data" entry of the object dictionary (index 2012_{hex}, subindex 00_{hex}) via the first Tx PDO.

SDO protocol

General overview

Service Data Objects, or SDOs for short, are used for the confirmed transfer of variable length data between two stations. The data transfer from one station to another is described in the client server model. An SDO client (initiating station) has here direct access to the entries of the object dictionary of an SDO server and can download data records of any length to a server and upload them from a server. The data record to be transferred is specified by the index and subindex of the object dictionary entry that represents the data record. The connection between an SDO server requires two CAN identifiers as a message ID is required for each transfer direction. The connection between a client and a server is also called the SDO channel.

Segmented transfer is required in order for data of any length to be transferred via an SDO channel since the maximum transfer capacity of a CAN telegram is only 8 bytes. This is based on the SDO protocols specified. Reference CiA document [1] for details.

Segmented protocol

If access to the object dictionary requires the transfer of more than 4 bytes, access to the object dictionary entry is specified with a 16-bit index and 8-bit subindex within a confirmed initialization sequence. The confirmed and segmented data is then transferred. Every transfer moves 7 bytes of data. The protocol on which this is based ensures receive-based flow control as well as the detection of any data segments that are transferred twice. The data transfer can be aborted by either the client or the server.

The transfer is initiated by means of an Initiate Download sequence for a segmented (non-expedited) data transfer. The data is then transferred in segments. Figure 19 shows the basic principle of the segmented SDO transfer.

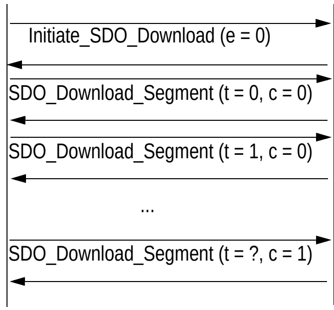


Figure 19: Segmented SDO download protocol

Expedited transfer protocol

If no more than 4 bytes are to be transferred, this can be executed with the expedited transfer protocol. This transfers the data already with a one byte protocol information as well as the address of the OD entry (index, subindex) within the initialization sequence (→ figure 20).

Figure 20: Expedited SDO download protocol.

Control byte

The control byte specifies the type of telegram (request/ response), type of transfer (normal/expedited) and the number of bytes in the data field that do not contain any data.

Figure 21 shows the protocol for writing an OD entry using the expedited SDO protocol. The client control byte indicates that an Initiate Download Request is present. This byte also indicates the transfer type as “expedited transfer”, as well as the number of data bytes contained in the data field.

The server control byte indicates an Initiate Download Response accordingly. The logical address of the OD entry is then sent as a 16-bit index and 8-bit subdindex following the control bytes.

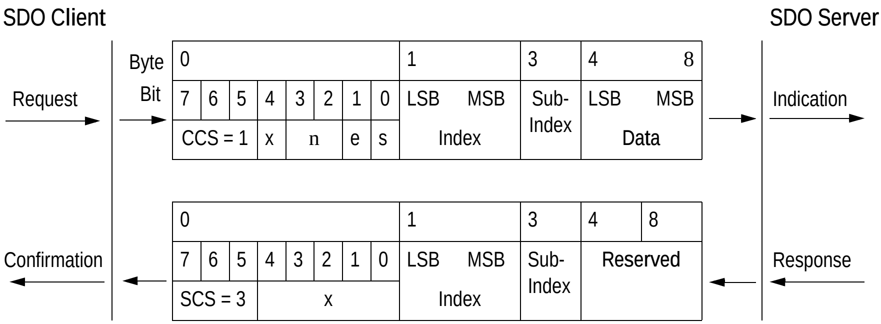


Figure 21: Writing an OD entry using the Expedited Domain Download Protocol

e = transfer type 0: non-expedited transfer, 1: expedited transfer

s = size indicator 0: size not indicated, 1: size indicated

n = number of bytes in data that do not contain data

x = nc

CCS = 1 Client Command Specifier = initiate download request

SCS = 3 Server Command Specifier = initiate download response

→ Reference CiA document [1] for details.

The Download_SDO_Segment_Protocol is presented here for a better understanding of the following examples.

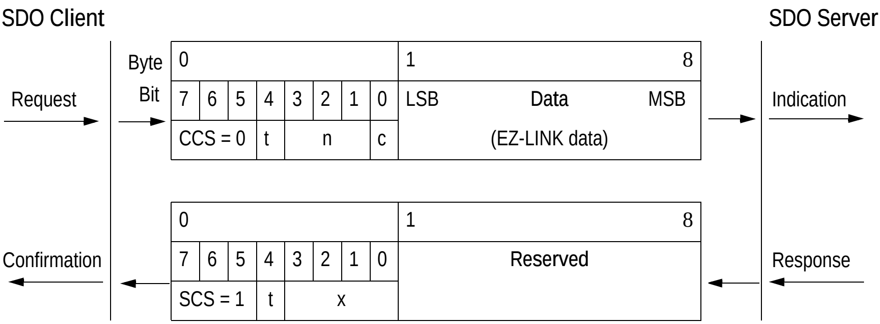


Figure 22: Download Transfer Segment after successful initialization.

t	Toggle Bit: The Toggle bit must be inverted with every transferred data packet. Request and Response must use the same bit as Toggle bit.
c	Indication whether segments still have to be transferred: 0: Other segments still have to be transferred 1: No other segments have to be transferred
n	Number of Bytes in Data that do not contain data
x	Value has no meaning
CCS = 0	Client Command Specifier = download segment request
SCS = 1	Server Command Specifier = download segment response

SDO protocol for EZ/EZD

Different CANopen telegram sequences have to be initiated in order to access the acyclic data of the basic unit. The entire sequence is illustrated in Figure 23.

1

First of all, the client initiates with Initiate SDO Download the write operation to the object dictionary Command entry in the server:

Device series	Object dictionary entries
EZ 700/800/EZD	Index 3021_{hex} Subindex 00_{hex}

2

Data lengths for the SDO transfer with the EZ221-CO are used as follows:

Device series	Length of the EZ-LINK data
EZ700/800/EZD	8 bytes

As the data length is more than 4 bytes, a Download SDO Segment is required in order to complete the segmented transfer. The EZ Protocol Handler then downloads the received data to EZ/EZD, using the extended protocol.

3 The client then checks with Initiate SDO Upload whether the transfer is completed. This is indicated by the status in the object dictionary:

Device series	Object dictionary entries
EZ 700/800/EZD	Index 3020_{hex} Subindex 00_{hex}

As only one byte is transferred at this stage, this is executed with the Expedited Transfer.

4 The client polls the status cyclically (at intervals of approx. 50 to 100 ms), until the content is 00_{hex}. The response from EZ/EZD is then provided in the object dictionary.

Device series	Object dictionary entries
EZ 700/800/EZD	Index 3022_{hex} Subindex 00_{hex}

5 In order to read the message, the client initiates the read operation with Initiate SDO Upload.

6 Since this data also has a length of up to 8 bytes, a subsequent Upload SDO Segment is required in order to read the remaining data.

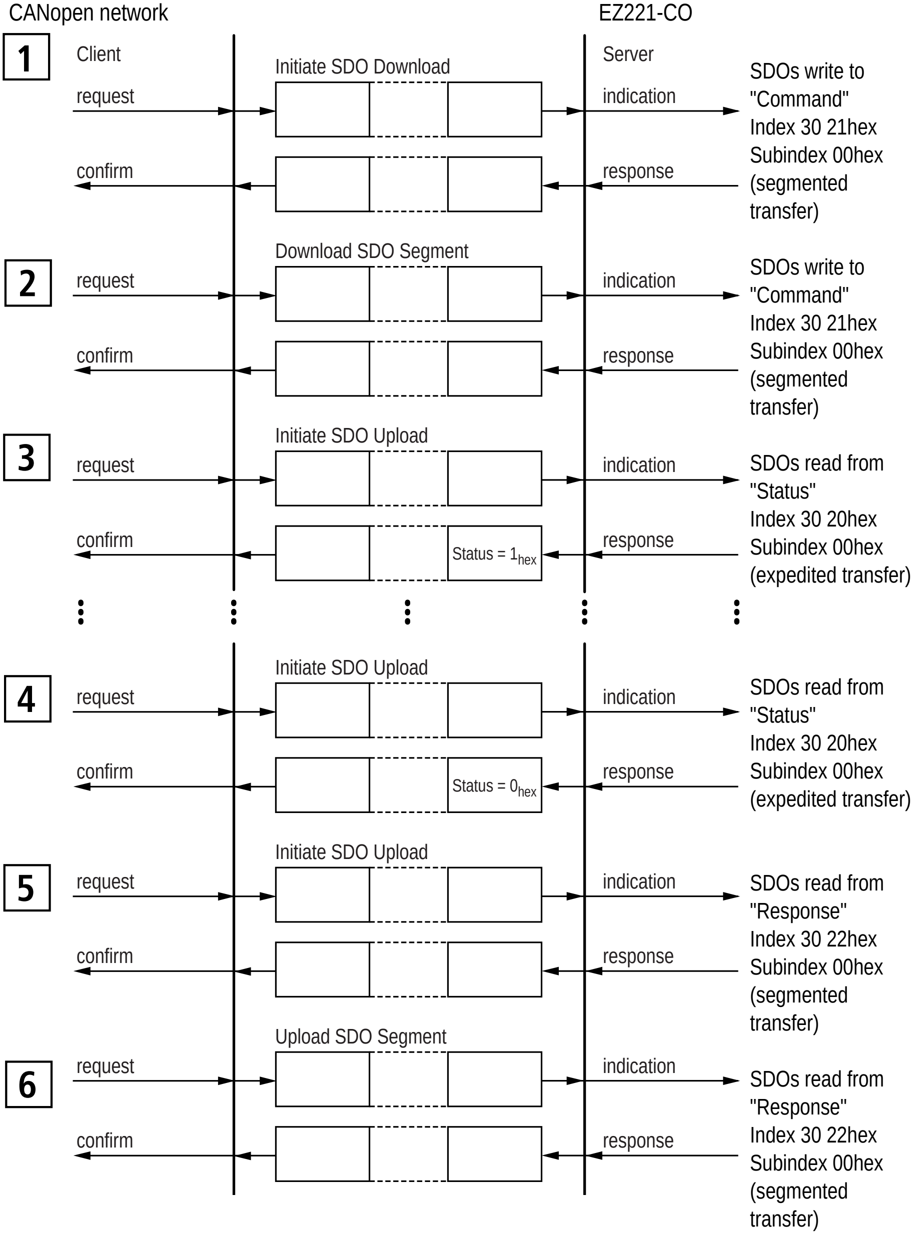


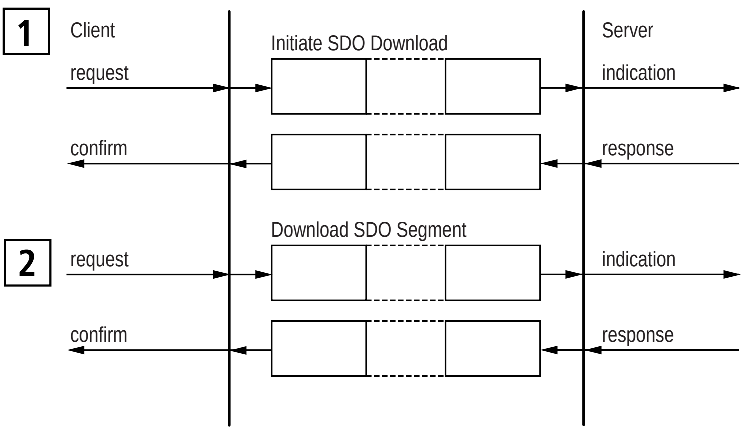
Figure 23: Sequence for extended SDO protocol

Example of EZ800: Read time (8 bytes)

The time is read from the basic unit via the SDO Transfer. The following EZ telegram structure is specified for this in the manual (→ page 128).

Byte	Meaning		Value
0	Command: Read		93
1	Len		05
2	Index		00
3	Data 1	Hour	00
4	Data 2	Minute	00
5	Data 3	Day	00
6	Data 4	Month	00
7	Data 5	Year	00

This data must be transferred with the CANopen protocol.



SDOs write to "Command" Index 3021_{hex} Subindex 00_{hex} (segmented transfer)

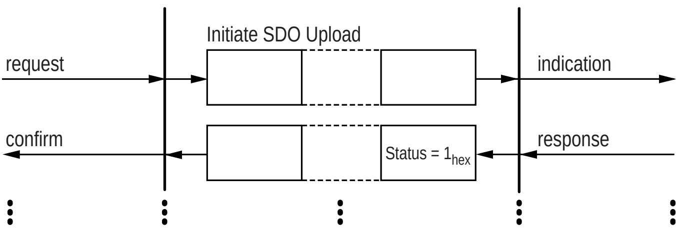
SDOs write to "Command" Index 3021_{hex} Subindex 00_{hex} (segmented transfer)

Description	ID (hex)	CAN data – byte (hex)							
Write command to EZ221-CO		0	1	2	3	4	5	6	7
Initialize the SDO download with 7 data bytes	602	21	21	30	00	08[1]	00	00	00
Confirmation of the SDO Block Transfer	582	60	21	30	00	00	00	00	00
Transfer of block 1 with 7 data bytes	602	00	93[2]	05[2]	00[2]	00[2]	00[2]	00[2]	00[2]
Confirmation of the data block to be transferred	582	20	00	00	00	00	00	00	00
Transfer of block 2 with 8th data byte	602	1D	00[2]	xx	xx	xx	xx	xx	xx
Confirmation of the data block to be transferred	582	30	00	00	00	00	00	00	00

1) Number of EZ data bytes to be transferred: EZ700/800/EZD – 8 bytes

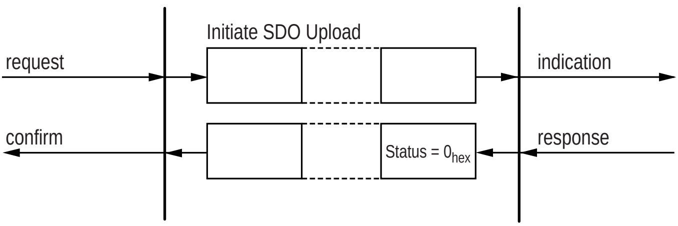
2) Valid data from EZ basic unit

3



SDOs read from "Status"
Index 3020_{hex}
Subindex 00_{hex}
(expedited transfer)

4

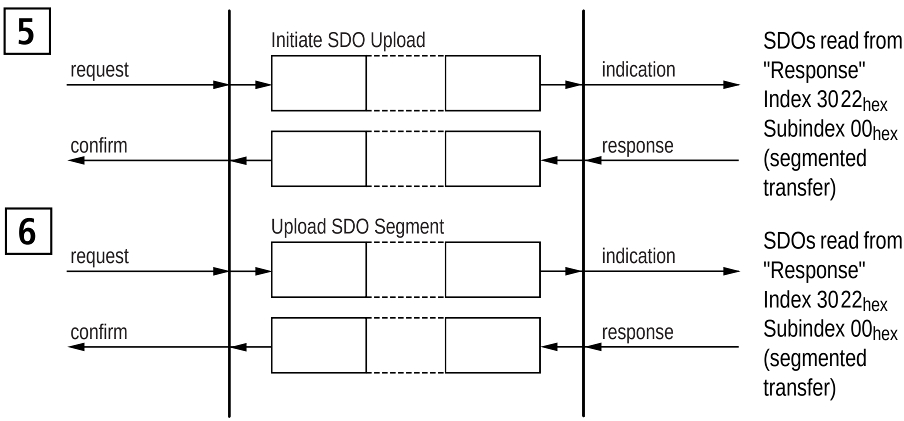


SDOs read from "Status"
Index 3020_{hex}
Subindex 00_{hex}
(expedited transfer)

Description	ID (hex)	CAN data – byte (hex)							
Scan status		0	1	2	3	4	5	6	7
Initialize the SDO upload	602	40	20	30	00	00	00	00	00
Transfer of status byte	582	4F	20	30	00	01[3]	xx	xx	xx
New attempt		Data 1 is scanned via index 3020_{hex} and Subindex 00_{hex} until the value = 00_{hex}.							
Initialize the SDO upload	602	40	20	30	00	00	00	00	00
Transfer of status byte	582	4F	20	30	00	00[3]	xx	xx	xx

3) Only if the value 00_{hex} is shown is it ensured that the corresponding response data is available in the receive buffer.

xx = Value has no meaning



Description	ID (hex)	CAN data – byte (hex)							
Call response		0	1	2	3	4	5	6	7
Initialize the SDO upload	602	40	22	30	00	00	00	00	00
Confirmation of the SDO Upload Block Transfer with 8 bytes	582	41	22	30	00	08[4]	00	00	00
Scan of 1st data block	602	60	22	30	00	00	00	00	00
Transfer of the first 7 EZ response bytes	582	00	C2[5]	05[5]	00[5]	16[5]	21[5]	01[5]	05[5]
Scan of 2nd data block	602	70	22	30	00	00	00	00	00
Transfer of the last EZ response byte	582	1D	03[5]	xx	xx	xx	xx	xx	xx

4) Number of EZ data bytes to be transferred: EZ700/800/EZD – 8 bytes

5) Valid data from EZ basic unit

xx = Value has no meaning

Evaluation of the received data

Byte	Meaning		Value
0	Response: read successful		C2
1	Len		05
2	Index		00
3	Data 1	Hour	16
4	Data 2	Minute	21
5	Data 3	Day	01
6	Data 4	Month	05
7	Data 5	Year	03

22:31 pm, 01.05.2003

Emergency protocol

The Write EMCY Protocol is used for the EZ221-CO, as shown in the figure below. The Emergency protocol does not require confirmation.

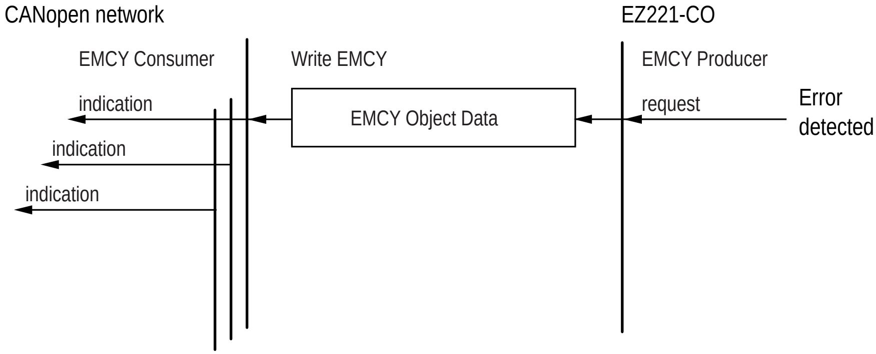


Figure 24: Emergency Object Protocol

7 PDO – Direct Data Exchange with EZ/EZD

The CANopen master can exchange the following data with the EZ/EZD via the direct cyclic data exchange (PDO):

- Write operation:
  - Set and reset the EZ/EZD inputs
  - Determine the RUN/STOP mode.
- Read operation:
  - Scan the output states of the EZ/EZD
  - Scan the operating mode of the EZ/EZD

The PDO protocol is used for the direct data exchange. Detailed information on this is provided on page 59. The direct data exchange is executed via the object dictionary entries 2011_{hex} (input data) and 2012_{hex} (output data) (→ page 54).

→ The terms “input data” and “output data” are used from the point of view of the CANopen master.

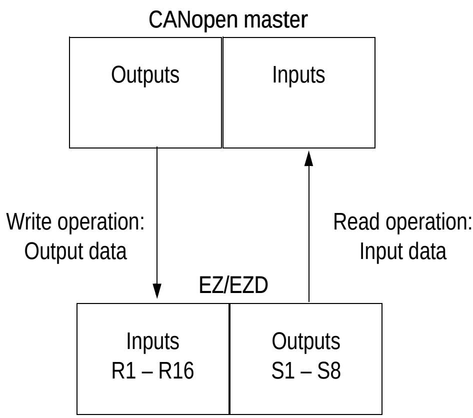


Figure 25: Input and output data as viewed from the CANopen master

Output data (2011_{hex}): operating mode, R1 – R16

The entries 2011_{hex} and 2012_{hex} can be mapped and can be transferred via PDOs. The object 2011_{hex} contains the output data of the CANopen master that is written via the EZ221-CO gateway to the inputs (R1 – R16) of the EZ/EZD. The output data is provided in bytes 0 to 2 and is described in detail in the following tables:

Table 1: Byte 0 to 2: output data, operating mode

Byte	Meaning	Value
0	Set operating mode	→ table 2
1	Set/reset the EZ/EZD inputs R9 to R16	→ table 3
2	Set/reset the EZ/EZD inputs R1 to R8	→ table 4

The master writes the following data to the bytes 0, 1 and 2:

Table 2: Byte 0: Operating mode

EZ/EZD operating mode	Bit 7	6	5	4	3	2	1	0
Index for setting the basic unit to the safety state	0	0	0	0	0	0	0	0
Index for transferring valid data	0	0	0	1	0	1	0	0
RUN command	0	0	1	1	0	1	0	0
STOP command	0	1	0	0	0	1	0	0

0 = status “0”, 1 = status “1”

Explanation:

Value 14_{hex} = $0001\,0100_{bin}$:
Byte 0 must always contain this value if data is to be written to the EZ/EZD basic unit via the EZ221-CO gateway.

Value 34_{hex} = $0011\,0100_{bin}$:
This value sets the EZ/EZD status from STOP to RUN. It is only interpreted as a command and therefore does not permit an additional transfer of data. The index value 14_{hex} must be used in this situation.

Value 44_{hex} = $0100\,0100_{bin}$:
This value sets the EZ/EZD status from RUN to STOP. It is also used only as command and it therefore works in the same way as the RUN command.

Value 00_{hex} = $0000\,0000_{bin}$:
If this value is written to the control byte, the gateway overwrites the Rx data with zero. This function is only required if a master is to be set to STOP mode and as a resultant final measure transfers zero values in all mapped PDOs in order to ensure a safety state.

→ Even if the I/O of a control relay can be assigned directly to a specific memory area of the master PLC, the correct data structure format (e.g.: input data byte 0 = 14_{hex}) must nevertheless still be observed.

Table 3: Byte 1: Set/reset the EZ/EZD inputs R9 to R16

EZ/EZD input	Bit							
	7	6	5	4	3	2	1	0
R9								0/1
R10							0/1	
R11						0/1		
R12					0/1			
R13				0/1				
R14			0/1					
R15		0/1						
R16	0/1							

0 = status “0”, 1 = status “1”

Example:
Value 19_{hex} = 00011001_{bin}:
Enables R13, R12 and R9.

Table 4: Byte 2: Set/reset the EZ/EZD inputs R1 to R8

EZ/EZD input	Bit							
	7	6	5	4	3	2	1	0
R1								0/1
R2							0/1	
R3						0/1		
R4					0/1			
R5				0/1				
R6			0/1					
R7		0/1						
R8	0/1							

0 = status “0”, 1 = status “1”

Example:
Value $2B_{hex}$ = $0010\ 1011_{bin}$:
Enables R6, R4, R2 and R1.

→ If control commands and I/O data are used at the same time:

- The inputs will retain their previous state until this control command has been executed.
- The input bytes will be updated after the data exchange control command has been terminated.

Input data (2012_{hex}): operating mode, S1 – S8

The entries 2011_{hex} and 2012_{hex} can be mapped and can be transferred via PDOs. The object 2012_{hex} contains the output data of the EZ/EZD (S data) that is transferred via the EZ221-CO gateway to the CANopen master. The tables below describe the structure of the input data in detail.

Table 5: Input data, operating mode

Byte	Meaning	Value
0	Scan the operating mode	→ table 6
1	Scan status of the EZ outputs S1 to S8	→ table 7
2	n.c.	00_{hex}

The master reads the following data from bytes 0, 1 and 2:

Table 6: Byte 0: Operating mode

EZ/EZD identification	Bit 7	6	5	4	3	2	1	0 STOP/RUN
Without debounce	0	0	0	1	0	0	0	0/1
With debounce	0	0	1	0	0	0	0	0/1

0 = status “0” 1 = status “1”

Example:
Value 21_{hex} = 00100001_{bin}:
EZ is in RUN mode and operates with debounce

Table 7: Byte 1: Status of the EZ outputs S1 to S8

EZ/EZD output	Bit 7	6	5	4	3	2	1	0
S1								0/1
S2							0/1	
S3						0/1		
S4					0/1			
S5				0/1				
S6			0/1					
S7		0/1						
S8	0/1							

0 = status “0” 1 = status “1”

Example:
Value 19_{hex} = 00011001_{bin}:
S5, S4 and S1 are active

Byte 2: not used

→ If control commands and I/O data are used at the same time:

- The inputs will retain their previous state until this control command has been executed.
- The input bytes will be updated again after the data exchange control command has been executed.

If the status value of the coupling module is invalid (= 04_{hex}), then byte 1 (data byte) is transferred with the value 00_{hex} to the communication bus.

8 SDO – Control Commands for EZ700

The object dictionary entries Status (3020_{hex}), Command (3021_{hex}) and Response (3022_{hex}) represent the interface for extended data exchange with EZ700 on the CANopen communication bus. This allows you to transfer services from the following areas:

- Read/write date and time (page 82)
- Read/write image data (page 87)
- Read/write function block data (page 108).

The SDO-CANopen protocol (→ page 60) is required in order to ensure the safe exchange of data via CANopen from master to slave and vice versa.

Attention!
While a control command is being executed, the input and output data will remain in the state before the control command was called. Only after the control command data exchange has been completed, will the I/O data be refreshed.

Caution!
Only those values specified for the command code should be used. Check the values that you write in order to avoid malfunctions.

Read/write date and time

→ Please also note the relevant description of the real-time clock provided in the EZ500/700 manual (MN05013003E).

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command			
		Read	93	–
		Write	B3	–
	Response			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Len		05	05
2	Index		0 – 2[1]	0 – 2[1]
3 – 7	Data 1 – 5		depending on index, → table 8	

1) 0 = Time/date, → table 8
1 = Summer time, → table 9
2 = Winter time, → table 10

Table 8: Index 0 – date and time of real-time clock

Byte	Contents	Operand		Value (hex)
3	Data 1	Hour	0 to 23	0x00 to 0x17h
4	Data 2	Minute	0 to 59	0x00 to 0x3Bh
5	Data 3	Day	Day (1 to 28; 29, 30, 31; depending on month and year)	0x01 to 0x1Fh
6	Data 4	Month	1 to 12	0x01 to 0x0Ch
7	Data 5	Year	0 to 99 (corresponds to 2000-2099)	0x00 to 0x63h

Table 9: Index 1 – Summer time

Byte	Contents		Value (hex)
3	Data 1	Area	
		None	00
		Rule	01
		Automatic EU	02
		Automatic GB	03
		Automatic US	04
for "Area" = "Rule":			
4	Data 2	Summer time switching rule	→ table 11
5	Data 3		
6	Data 4		
7	Data 5		

Table 10: Index 2 – Winter time
(only valid if Area = “Rule” selected)

Byte	Contents		Value (hex)
3	Data 1	Area = Rule	01
4 – 7	Data 2 – 5	Winter time switching rule	→ table 11

Switching rule bit array

→ Please also read the detailed description in the EZ500/700 manual (MN05013003E).

The following table shows the composition of the corresponding data bytes.

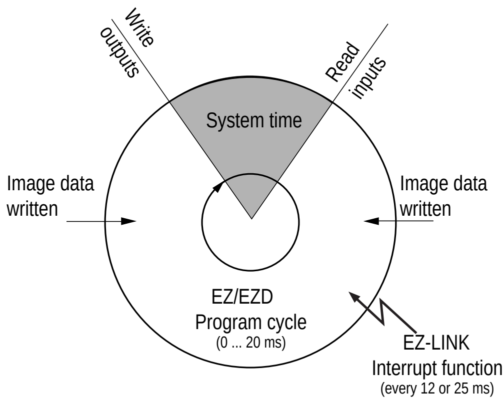
Table 11: Switching rule bit array

	Data 5	Data 5 / Data 4	Data 4	Data 4 / Data 3	Data 3	Data 2	Data 2	Data 2
Bit	31 30 29 28	27 26 25 24 23 22	21 20 19 18 17	16 15 14 13	12 11 10 9 8	7 6	5 4 3	2 1 0
	Difference	Time of time change		Month	Day	Rule_2	Day	Rule_1
	0: 0:30h	Minute: 0 to 59	Hour: 0 to 23	0 to 11	0 to 30	0: month	0: Su	0: on
	1: 1:00h					1: after the	1: Mo	1: on the first
	2: 1:30h					2: before the	2: Tu	2: on the second
	3: 2:00h						3: We	3: on the third
	4: 2:30h						4: Thu	4: on the fourth
	5: 3:00h						5: Fr	5: on the last
							6: Sa	

Reading image data

→ Please also observe the relevant description of possible image data provided in the EZ500/700 manual (MN05013003E) or in the EZSoft Help.

General notes on working with image data



When writing to image data, it must be taken into account that an image (e.g. inputs, outputs,...) used in the EZ/EZD program is also written cyclically by the actual program. The only image data that is unchanged is the data that is not used in the program and is therefore not overwritten in the program cycle. This operating principle also means that an image written via EZ-LINK, such as output data is only then output at the physical outputs of the EZ/EZD when the control relay is in RUN mode.

Read/write image data — Overview

Operands	Meaning	Read/ write	Type (hex)	Page
A1 – A16	Analog value comparators/threshold comparators: A1 – A16	Read	8C	88
C1 – C16	Counters: C1 – C16	Read	EE	89
D1 – D16	Text function blocks: D1 – D16	Read	94	90
I1 – I16	Local inputs: I1 – I16	Read	84	91
IA1 – IA4	Local analog inputs: IA1 – IA4	Read	8C	92
M1 – M16, N1 – N16	Write markers: M1 – M16/N1 – N16	Write	86/87	94
M1 – M16, N1 – N16	Read markers: M1 – M16/N1 – N16	Read	86/87	96
O1 – O4	Operating hours counters: O1 – O4	Read	EF	98
P1 – P4	Local P buttons: P1 – P4	Read	8A	99
Q1 – Q8	Local outputs: Q1 – Q8	Read	85	101
R1 – R16/ S1 – S8	Inputs/outputs of EZ-LINK: R1 – R16/S1 – S8	Read	88/89	102
T1 – T16	Timing relays: T1 – T16	Read	ED	104
Y1 – Y4	Year time switch: Y1 – Y8	Read	91	105
Z1 – Z3	Master reset: Z1 – Z3	Read	93	106
H1 – H4	7-day time switch: ⌚1 – ⌚8	Read	90	107

Analog value comparators/threshold comparators: A1 – A16

The following commands are used to read the logic state of the individual analog value comparators A1 to A16.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	8C	8C
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 12
5	Data 2 (Low Byte)	00	→ table 12
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 12: Byte 4 to 5: Data 1 to 2

Data 1	Bit 7	6	5	4	3	2	1	0
A1								0/1
A2							0/1	
...				...				
A8	0/1							
Data 2	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
A9								0/1
A10							0/1	
...				...				
A16	0/1							

Counters: C1 – C16

The following commands are used to read the logic state of the individual counters C1 – C16.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	EE	EE
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 22
5	Data 2 (Low Byte)	00	→ table 22
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 13: Byte 4 to 5: Data 1 to 2

Data 1	Bit 7	6	5	4	3	2	1	0
C1								0/1
C2							0/1	
...				...				
C8	0/1							
Data 2	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
C9								0/1
C10							0/1	
...				...				
C16	0/1							

Text function blocks: D1 – D16

The following commands are used to read the logic state of the individual text function blocks (D markers).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	94	94
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 14
5	Data 2 (High Byte)	00	→ table 14
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 14: Byte 4 to 5: Data 1 to 2

Data 1	Bit 7	6	5	4	3	2	1	0
D1								0/1
D2							0/1	
...				...				
D8	0/1							
Data 2	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
D9								0/1
D10							0/1	
...				...				
D16	0/1							

Local inputs: I1 – I16

This command string enables you to read the local inputs of the EZ700 basic unit. The relevant input word is stored in Intel format.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1)]
1	Len	02	02
2	Type	84	84
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 15
5	Data 2 (High Byte)	00	→ table 15
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 15: Byte 4 to 5: Data 1 to 2

Data 1	Bit 7	6	5	4	3	2	1	0
I1								0/1
I2							0/1	
..				..				
I8	0/1							
Data 2	**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I9								0/1
I10							0/1	
..				..				
I16	0/1							

Local analog inputs: IA1 – IA4

The analog inputs on the EZ700 basic unit (I7, I8, I11, I12) can be read directly via CANopen. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	02	02
2	Type	8C	8C
3	Index	00 – 03[2]	00 – 03[2]
4	Data 1 (Low Byte)	00	→ table 16
5	Data 2 (High Byte)	00	→ table 16
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

2) 00 = Analog input I7
01 = Analog input I8
02 = Analog input I11
03 = Analog input I12

Example:
A voltage signal is present at analog input 1. The required telegrams for reading the analog value are as follows:

Table 16: Example telegram for reading the value at the analog input

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response: Read successful	–	C2
1	Len	02	02
2	Type	8C	8C
3	Index	02[1]	02[1]
4	Data 1	00	4B
5	Data 2	00	03
6	Data 3	00	00
7	Data 4	00	00

1) 02 = Analog input I11

Byte 4 – Data 1 (Low Byte): $4B_{hex}$
Byte 5 – Data 2 (High Byte): 03_{hex}
→ corresponding 16-bit value: $034B_{hex}$ = 843

The value 843 corresponds to the 10 bit value of the analog converter. The following conversion is required for the actual analog value:

$$\frac{10\ \text{V}}{1023} \times \text{10 bit value} \quad => \quad \frac{10\ \text{V}}{1023} \times 843 = 8.24\ \text{V}$$

Write markers: M1 – M16/N1 – N16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	8C	–
	Response:		
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Len	01	01
2	Type[2]		
	With M marker	86	86
	With N marker	87	87
3	Index[2]	00 – 0F	00 – 0F
4	Data 1 (Low Byte)[3]	00/01	00/01
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

2) There are 16 M markers and 16 N markers. The markers are addressed by Type and Index:
Use Type to select the M or N marker.
Use Index to select the marker number.

3) The marker is set if a value is written to the data byte that does not equal zero. The marker is reset accordingly if the value 0 is written to data byte Data 1.

Example:
Marker M13 is set.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	8C	–
	Response:		
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Len	01	01
2	Type		
	M marker	86	86
3	Index	0C	0C
4	Data 1	01	00
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Read markers: M1 – M16/N1 – N16

Unlike the write operation, the marker read operation reads the entire marker area of a particular marker type (M or N) is read.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type		
	M marker	86	86
	N marker	87	87
3	Index[2]	00	00
4	Data 1 (Low Byte)	00	→ table 17
5	Data 2 (Low Byte)	00	→ table 17
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

2) There are 16 M markers and 16 N markers. The markers are addressed by Type and Index:
Use Type to select the M or N marker.
Use Index to select the marker number.

Table 17: Byte 4 to 5: Data 1 to 2

Data 1		Bit 7	6	5	4	3	2	1	0
M	N								
M1	N1								0/1
M2	N2							0/1	
...	...				...				
M8	N8	0/1							
Data 2		**Bit 7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
M9	N9								0/1
M10	N10							0/1	
...	–				...				
M16	N16	0/1							

Example:
The N markers are read:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type		
	N marker	87	87
3	Index	00	00
4	Data 1 (Low Byte)	00	04
5	Data 2 (Low Byte)	00	84
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

The markers N3, N11 and N16 are set.

Operating hours counters: O1 – O4

The following commands are used to read the logic state of the operating hours counters O1 – O4.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	EF	EF
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 18
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 18: Byte 4 : Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
O1								0/1
O2							0/1	
O3						0/1		
O4					0/1			
...	...	...	...	...				

Local P buttons: P1 – P4

The local P buttons are the display cursor buttons of the EZ700 basic unit. You can scan the buttons in both RUN and STOP mode.

→ Ensure that the P buttons are also activated via the System menu (in the basic unit).

Only one byte has to be transferred for the P buttons.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	8A	8A
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 19
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 19: Byte 4: Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
P1								0/1
P2							0/1	
P3						0/1		
P4					0/1			
–				0				
–			0					
–		0						
–	0							

Example:
Data 1 = 2_{hex} → P3 is active.

Local outputs: Q1 – Q8

The local outputs can be read directly via the CANopen fieldbus.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1)]
1	Len	01	01
2	Type	85	85
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 20
6 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 20: Byte 4 : Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
Q1								0/1
Q2							0/1	
..				..				
Q8	0/1							

Example:
Data 1 = 52_{hex} I Q2, Q5 and Q7 are active.

Inputs/outputs of EZ-LINK: R1 – R16/S1 – S8

This service allows you to read the local R and S data and the data of the NET stations (1 – 8) transferred via EZ-LINK, again from the relevant EZ700 image.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1)]
1	Len	01	01
2	Type		
	for R data	88	88
	for S data	89	89
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 21
5	Data 2 (Low Byte)	00	→ table 21
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 21: Byte 5 to 6: Data 1 to 2

Data 1		Bit	7	6	5	4	3	2	1	0
RW	SW									
R1	S1									0/1
R2	S2								0/1	
...	...					...				
R8	S8		0/1							
Data 2		Bit	7	6	5	4	3	2	1	0
R9	–									0/1
R10	–								0/1	
...	–					...				
R16	–		0/1							

Timing relays: T1 – T16

The following commands are used to read the logic state of the individual timers T1 - T16.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	ED	ED
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 22
5	Data 2 (Low Byte)	00	→ table 22
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 126

Table 22: Byte 4 to 5: Data 1 to 2

Data 1	Bit	7	6	5	4	3	2	1	0
T1									0/1
T2								0/1	
...					...				
T8		0/1							
Data 2	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
T9									0/1
T10								0/1	
...					...				
T16		0/1							

Year time switch: Y1 – Y8

The following commands are used to read the logic state of the individual year time switches.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	91	91
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 23
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 23: Byte 4: Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
HY1								0/1
HY2							0/1	
HY3						0/1		
HY4					0/1			
HY5				0				
HY6			0					
HY7		0						
HY8	0							

Example:
Data 1 = 1_{hex} → HY2 is active

Master reset: Z1 – Z3

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	93	93
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 24
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 24: Byte 4: Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
Z1 for Q outputs								0/1
Z2 for M markers							0/1	
Z3 for outputs and markers						0/1		
...	0	0	0	0	0			

7-day time switch: Ö1 – Ö8

The following commands are used to read the logic state of the individual 7-day time switches.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	90	90
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 25
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 126

Table 25: Byte 4: Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
HW1								0/1
HW2							0/1	
HW3						0/1		
HW4					0/1			
HW5				0				
HW6			0					
HW7		0						
HW8	0							

Example:
Data 1 = 2_{hex} → Ö3 is active.

Read/write function block data

→ Please also observe the relevant description of the function blocks provided in the EZ500/700 manual (MN05013003E) or in the EZSoft Help.

General notes

Always note the following when working with function blocks:

- The relevant data is transferred in Intel format. In other words, the first byte is the low byte (Byte 5) and the last byte (byte 8) the high byte.
- The maximum data length is 4 bytes. All values must be transferred in hexadecimal format.

Overview

Operands	Meaning	Read/ write	Type (hex)	Page
A1 – A16	Analog value comparator/threshold comparator: A1 – A16	Read/write	8D	109
C1 – C16	Counter relays: C1 – C16	Read/write	8F	112
O1 – O4	Operating hours counters: O1 – O4	Read/write	92	115
T1 – T16	Timing relays: T1 – T16	Read/write	8E	117
Y1 – Y8	Year time switch: Y1 – Y8	Read/write	A2	120
⌚1 – ⌚8	7-day time switch: ⌚1 – ⌚8	Read/write	A1	123

Analog value comparator/threshold comparator: A1 – A16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8D	8D
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 26	
4 – 7	Data 1 – 4	depending on index, → table 27	

1) Possible causes → page 126

2) EZ provides 16 analog comparators A1 to A16 for use as required. These can be addressed using the instance (0 – F).

Table 26: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 27		×	
01	Control byte → table 28		×	
02	Comparison value 1	I1[2]	×	c[1]
03	Comparison value 2	I2[2]	×	c[1]
04	Gain factor for I1 (I1 = F1 × I1)	F1[2]	×	c[1]
05	Gain factor for I2 (I2 = F2 × I2)	F2[2]	×	c[1]
06	Offset for value I1 (I1 = OS + actual value at I1)	OS[2]	×	c[1]
07	Switching hysteresis for value I2	HY[2]	×	c[1]

1) The value can only be written if it is assigned a constant in the program.

2) A 16-bit value is transferred in data bytes Data 1 – Data 2. It should be remembered that the low byte 1 is in Data 1 (Byte 5) and the high byte 2 (byte 8) in Data 2.
Example: $5327_{dec} = 14CF_{hex}$ → Data 1 = 0xCF, Data 2 = 0x14

Table 27: Index 00 – Parameters

Meaning	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Appears in the parameter menu																	
Yes/no																	0/1
Compare																	
FB not used														0	0	0	
EQ (=)														0	0	1	
GE (≧)														0	1	0	
LE (≦)														0	1	1	
GT (>)														1	0	0	
LT (<)														1	0	1	
Use as constant and therefore can be written to																	
I1= Constant													0/1				
F1= Constant												0/1					
I2= Constant											0/1						
F2 = Constant										0/1							
OS = Constant									0/1								
HY = Constant								0/1									
Not used		0	0	0	0	0	0										

Example:
Data 1 (Byte 4) = 0xA3, Data 2 (Byte 5) = 0x03
→ Resulting 16-bit value = $03A3_{hex}$

Meaning: HY, OS, F2, F1 are assigned a constant; I1, I2 are assigned to a variable such as I7, I8 C2...etc., appears in the Parameter menu;

The output of the analog value comparator is active for as long as the comparison (I1 × F1) + OS = (I2 × F2) + HY is fulfilled.

Table 28: Index 01 – Control byte

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	Q1[1]

1) Status 1 if comparison condition is fulfilled.

Counter relays: C1 – C16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8F	8F
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 29	
4 – 7	Data 1 – 4	depending on index, → table 30	

1) Possible causes → page 126
2) EZ provides 16 counters C1 to C16 for use as required. These can be addressed using the instance (0 – F).

Table 29: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 30		×	
01	Control byte → table 31		×	
02	Actual value	S1[2]	×	c[1]
03	Counter setpoint 2	S2[2]	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.
2) A 16-bit value is transferred in data bytes Data 1 – Data 2. It should be remembered that Data 1 is the low byte and Data 2 the high byte.

Table 30: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the parameter menu									
Yes/no									0/1
Counter mode									
FB not used							0	0	
Up/down counter (N)							0	1	
High-speed up/down counter (H)							1	0	
Frequency counter (F)							1	1	
Use as constant and therefore can be written to									
Counter setpoint S1						0/1			
Unused bits		–	–	–	–				

Example:
Data 1 (Byte 4) = 0x07

Meaning:
The values appear in the Parameter menu. The counter is used in the mode of the frequency meter. The counter setpoint 1 is not assigned to a constant and cannot therefore be written to.

Table 31: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	C4	RE3	D2	Q11

1) Switch contact
2) Count direction: 0 = up counting, 1 = down counting
3) Reset, the timing relay is reset (reset coil)
4) Count coil, counts on every rising edge

Example:
the actual value of C3 is to be read:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	89	–
	Response: Read successful	–	C2
1	Type	8F	8F
2	Instance	02	02
3	Index	02	02
4	Data1	00	12
5	Data 2	00	03
6	Data 3	00	00
7	Data 4	00	00

Explanation:

Data 1 = 12
Data 2 = 03
→ resulting 16-bit value = 0312_{hex} = 786_{dec}

Counter status = 786

Operating hours counters: O1 – O4

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	92	92
2	Instance[2]	00 – 03	00 – 03
3	Index	→ table 32	
4 – 7	Data 1 – 4	depending on index, → table 33	

1) Possible causes → page 126
2) EZ provides 4 operating hours counters O1 to O4. These can be addressed using the instance (0 – 3).

Table 32: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 33		×	
01	Control byte → table 34		×	
02	Actual value	S1[2]	×	c[1]
03	Counter setpoint 2	S2[2]	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.
2) A 32-bit value is transferred in data bytes Data 1 – Data 4. It should be remembered that the Data 1 is the low byte and Data 4 the high byte.

Table 33: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the parameter menu									
Yes/no									0/1
Use in the program									
Setpoint S1								0/1	
Unused bits		–	–	–	–	–	–		

Example:
Data 1 (Byte 4) = 0x01

Meaning:
The values appear in the Parameter menu.

Table 34: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	–	RE[3]	EN[2]	Q1[1]

1) Switch contact
2) Enable, the timing relay is started (trigger coil)
3) Reset, the timing relay is reset (reset coil)

Example:
Index 02/03

Transferred values: Data 1 0x21
Data 2 0x23
Data 3 0x40
Data 4 0x00

Resulting value: $00402321_{hex} = 4203297_{dec}$

Timing relays: T1 – T16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8E	8E
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 35	
4 – 7	Data 1 – 4	depending on index, → table 36	

1) Possible causes → page 126
2) EZ provides 16 timing relays T1 to T16 for use as required. These can be addressed using the instance (0 – F).

Table 35: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 36		×	
01	Control byte → table 37		×	
02	Actual value 1	T	×	c[1]
03	Time setpoint 1	S1[2]	×	c[1]
04	Time setpoint 2	S2[2]	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.
2) A 16-bit value is transferred in data bytes Data 1 – Data 2. It should be remembered that the low byte is in Data 1 and the high byte in Data 2.

Table 36: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the parameter menu									
Yes/no									0/1
Timer mode									
On-delayed						0	0	0	
Off-delayed						0	0	1	
On-delayed with random setpoint						0	1	0	
Off-delayed with random setpoint						0	1	1	
On and off delayed (two time setpoints)						1	0	0	
On and off delayed each with random setpoint (two time setpoints)						1	0	1	
Impulse transmitter						1	1	0	
Flashing relay (two time setpoints)						1	1	1	
Time base									
FB not used				0	0				
Millisecond: S				0	1				
Second: M:S				1	0				
Minute: H:M				1	1				
Use as constant and therefore can be written to									
Time setpoint S1			0/1						
Time setpoint S2		0/1							

Example:
Data 1 (Byte 4) = 0xAC

Meaning:
The values appear in the Parameter menu. The time is used in the impulse transmitter mode with the Second time base. The time setpoint S1 is assigned a constant and the time setpoint S2 is assigned a variable such as I7, I8 C2...etc.

Table 37: Index 01 – Control byte

	Bit	7	6	5	4	3	2	1	0
FB input/output Data 3		–	–	–	–	ST4)	RE3)	EN2)	Q11)

1) Switch contact
2) Enable, the timing relay is started (trigger coil)
3) Reset, the timing relay is reset (reset coil)
4) Stop, the timing relay is stopped (Stop coil)

Example:
The time setpoint 1 is to be read:

Byte	Meaning	Value (hex), sent by Master	Slave
0	Command: Read	89	–
	Response: Read successful	–	C2
1	Type	8E	8E
2	Instance	00	00
3	Index	03	03
4	Data1	00	4C
5	Data 2	00	06
6	Data 3	00	00
7	Data 4	00	00

Explanation:

Data 1 = 4C
Data 2 = 06
→ resulting 16-bit value = $064C_{hex}$ = 1612_{dec}

Meaning depending on set time base:

millisecond	S	16120 ms	16,120 s
Seconds	M:S	1620 s	26:52 Minutes
Minute	H:M	1612 min	67:04 Hours

Year time switch: Y1 – Y8

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	A2	A2
2	Instance[2]	00 – 07	00 – 07
3	Index	→ table 38	
4 – 7	Data 1 – 4	depending on index, → table 39	

1) Possible causes → page 126

2) EZ provides 8 year time switches Y1 to Y8 for use as required. These can be addressed using the instance (0 – 7).

Table 38: Operand overview

Index (hex)	Operand	Read	Write
00	Parameters → table 39	×	
01	Control byte → table 40	×	
	Channel A	×	c1)
11	Time point ON	×	c1)
12	Time point OFF	×	c1)
	Channel B	×	c1)
21	Time point ON	×	c1)
22	Time point OFF	×	c1)
	Channel C	×	c1)
31	Time point ON	×	c1)
32	Time point OFF	×	c1)
	Channel D	×	c1)
41	Time point ON	×	c1)
42	Time point OFF	×	c1)

1) The value can only be written if it is assigned to a constant in the program.

Note: The switch points are transferred in data bytes Data 1 – Data 3.

Table 39: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the parameter menu									
Channel A									0/1
Channel B								0/1	
Channel C							0/1		
Channel D						0/1			
Unused bits		–	–	–	–				

Example:
Data 1 (Byte 4) = 0x03 → The values of the year time switch of channel A and B in the parameter menu.

Table 40: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	–	–	–	Q1[1]

1) Status 1, if the count condition is fulfilled.

Channel A, Index 11/12
Index 0x11 channel A ON time
Index 0x12 channel A OFF time
Data 1 (Byte 4) – Day
Data 2 (Byte 5) – Month
Data 3 (Byte 6) – Year

Example:
The year time switch channel A is to be activated on the 21.04.2004.

Index = 0x11
Data 1 = 0x15
Data 2 = 0x04
Data 3 = 0x04

The year time switch channel B is to be deactivated on the 05.11.2012.

Index = 0x22
Data 1 = 0x05
Data 2 = 0x0B
Data 3 = 0x0C

7-day time switch: ⏲1 – ⏲8

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	A1	A1
2	Instance[2]	00 – 07	00 – 07
3	Index	→ table 41	→ table 41
4 – 7	Data 1 – 4	depending on index, → table 42	

1) Possible causes → page 126

2) EZ provides 8 7-day time switches ⏲1 to ⏲8 for use as required. These can be addressed using the instance (0 – 7).

Table 41: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 42		×	
01	Control byte → table 43		×	
11	Channel A	Day on/off	×	c[1]
12		On time	×	c[1]
13		Off time	×	c[1]
21	Channel B	Day on/off	×	c[1]
22		On time	×	c[1]
23		Off time	×	c[1]
31	Channel C	Day on/off	×	c[1]
32		On time	×	c[1]
33		Off time	×	c[1]
41	Channel D	Day on/off	×	c[1]
42		On time	×	c[1]
43		Off time	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

Note: A 16-bit value is transferred in data bytes Data 1 – Data 4. It should be remembered that Data 1 is the low byte and Data 2 the high byte.

Table 42: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the parameter menu									
Channel A									0/1
Channel B								0/1	
Channel C							0/1		
Channel D						0/1			
Unused bits		–	–	–	–				

Example:
Data 1 (Byte 4) = 0x03

Meaning:
The values of the 7-day time switch from channel A and B appear in the parameter menu.

Table 43: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	–	–	–	Q1[1]

1) Status 1, if the count condition is fulfilled.

Channel A, Index 11/12/13
Index 0x11 channel A Weekday on/off
Data 1 (Byte 4) – Weekday on
Data 2 (Byte 5) – Weekday off
0x01 = Sunday ... 0x07 = Saturday

The 16-bit value equals 0x00 if the channel is not used.

Index 0x12 – On time (2 Byte)
Index 0x13 – Off time (2 Byte)
Data 1 (Byte 4) – Hour
Data 2 (Byte 5) – Minute

Example: On time at 13:43 p.m.
Data 1 = 0x0D
Data 2 = 0x2B

Analysis – error codes via EZ-LINK

The EZ700 basic unit will return a defined error code in the event of an incorrectly selected operating mode or an invalid telegram. The error code transferred has the following structure:

Telegram structure

Byte	Meaning	Slave transmits (value hex)
0	Response	
	Command rejected	C0
1	Type	00
2	Instance	00
3	Index	00
4	Error code	→ table 44

Table 44: Error codes

Error code	Description
0x01	Unknown telegram transmitted.
0x02	Unknown object transmitted.
0x03	Unknown command transmitted.
0x04	Invalid instance transmitted.
0x05	Invalid parameter set transmitted.
0x06	An attempt was made to write to a variable that is not a constant.
0x0C	The device is in an invalid device mode. STOP → RUN or RUN → STOP
0x0D	Invalid display access. Exit the menu level so that the status display is showing in the display. The clock cannot be written to.
0xF0	Attempt made to control an unknown parameter.
0xF1	Impermissible value

9 SDO – Control Commands for EZ800/EZD

The OD entries Status (3020_{hex}), Command (3021_{hex}) and Response (3022_{hex}) provide the interface for extended data exchange with EZ800 and EZD on the CANopen communication bus. This allows you to transfer services from the following areas:

- Read/write date and time (page 128)
- Read/write image data (page 133)
- Read/write function block data (page 153).

The SDO-CANopen protocol (→ page 60) is required in order to ensure the safe exchange of data via CANopen from master to slave and vice versa.

Attention!
While a control command is being executed, the input and output data will remain in the state before the control command was called. Only after the Control commands data exchange has been completed, will the I/O data be refreshed.

Caution!
Only those values specified for the command code should be used. Check the values that you write in order to avoid malfunctions.

Read/write date and time

→ Please also note the relevant description of the real-time clock provided in the EZ800 manual (MN05013004E) and the EZD manual (MN05013005E).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	05	05
2	Index	00	00
3 – 7	Data 1 – 5		
	Read operation	00	→ table 45
	Write operation	→ table 45	00

Table 45: Byte 3 to 7: Data 1 to 5

Byte	Contents		Value (hex)
3	Data 1	Hour (0 to 23)	00 – 17
4	Data 2	Minute (0 to 59)	00 – 3B
5	Data 3	Day (1 to 31; depending on month and year)	01 – 1F
6	Data 4	Month (1 to 12)	01 – 0C
7	Data 5	Year (0 – 99, corresponds to 2000 – 2099)	00 – 63

Winter/summer time, DST

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	05	05
2	Index		
	Summer/winter time	01 → table 46	→ table 46
	Winter time (according to Area = Rule)[1]	02 → table 47	02 → table 47
3 – 7	Data 1 – 5		
	Read operation	00	depending on index, → table 46, 47
	Write operation	depending on index, → table 46, 47	00

1) Detailed setting options for EZ800/EZD

Table 46: Index 01 – Summer / winter time change

Byte	Contents			Value (hex)
3	Data 1	Area		
			None	00
			Manual	01
			Automatic EU	02
			Automatic GB	03
			Automatic US	04
			Rule[1]	05
For Area = Manual				
4	Data 2	Set summer time day 1 to 28, 29, 30, 31 (depending on month and year)		00 – 3B
5	Data 3	Set Summer time month (1 – 12)		01 – 1F
6	Data 4	Set winter time day 1 to 28, 29, 30, 31 (depending on month and year)		01 – 0C
7	Data 5	Set winter time month (1 – 12)		00 – 63
For Area = Rule[1]:				
4 – 7	Data 2 – 5	Summer time switching rule		→ table 48

1) Detailed setting options for EZ800/EZD

Table 47: Index 02 – Winter time (only valid if Area = Rule selected)

Byte	Contents		Value (hex)
3	Data 1	Area = Rule	01
4 – 7	Data 2 – 5	Winter time switching rule	→ table 48

Switching rule bit array

→ Please also read the detailed description in the EZ800 manual (MN05013004E). The following table shows the composition of the corresponding data bytes.

Table 48: Switching rule bit array

	Data 5				Data 4			Data 3		Data 2	
Bit	31 30 29	28 27 26	25 24	23 22 21 20 19	18 17 16	15	14 13 12 11 10	9 8	7 6 5 4	3 2 1 0
	Rule_1	Day	Rule_2	Day	Month		Time of time change			Difference
	0: on	0: Su	0: month	0 to 30	0 to 11		Hour: 0 to 23		Minute: 0 to 59	0: 0:30h
	1: on the first	1: Mon	1: after the							1: 1:00h
	2: on the second	2: Tue	2: before the							2: 1:30h
	3: on the third	3: We								3: 2:00h
	4: on the fourth	4: Thu								4: 2:30h
	5: on the last	5: Fri								5: 3:00h
		6: Sa								

Example
The real-time clock of the EZ800 is to be set to Friday 23.05.2003, 14:36.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	B3	–
	Response: Write successful	–	C1
1	Len	05	05
2	Index	00	00
3	Data 1 (hour)	0E	00
4	Data 2 (Minute)	24	00
5	Data 3 (day)	17	00
6	Data 4 (Month)	05	00
7	Data 5 (year)	03	00

→ All values must be transferred as hexadecimal values.

Read/write image data

→ Please also observe the relevant description of possible image data provided in the EZ800 manual (MN05013004E) or in the EZSoft Help. The information provided in section “General notes on working with image data” on page 86 also applies to EZ800.

Overview

Operands	Meaning	Read/write	Command (hex)	Page
IA1 – IA4	Local analog inputs: IA1 – IA4	Read	02	134
ID1 – ID16	Local diagnostics: ID1 – ID16	Read	03	136
IW0	Local inputs: IW0	Read	01	138
IW1 – IW8	Inputs of the stations: IW1 to IW8	Read	01	140
M...	Markers: M..	Read/write	0B – 0E	141
P1 – P4	Local P buttons: P1 – P4	Read	06	144
QA1	Local analog output: QA1	Read/write	05	146
QW0, QW1 – QW8	Local outputs: QW0/ outputs of the stations QW1 – QW8	Read/write	04	147
R1 – R16 S1 – S8	Inputs/outputs of EZ-LINK: RW/SW	Read	07/09	149
RN1 – RN32 SN1 – SN32	Receive Data Network: RN1 – RN32/ Transmit Data Network: SN1 – SN32	Read	08/0A	151

Local analog inputs: IA1 – IA4

The analog inputs on the EZ800 and EZD basic units can be read directly via CANopen. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	02	02
3	Index	01 – 04[1]	01 – 04[1]
4	Data 1 (Low Byte)	00	See example
5	Data 2 (High Byte)	00	See example
6 – 7	Data 3 – 4	00	00

1) 01 = Analog input I7
02 = Analog input I8
03 = Analog input I11
04 = Analog input I12

Example
A voltage signal is present at analog input 1. The required telegrams for reading the analog value are as follows:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response: Read successful	–	C2
1	Len	02	02
2	Type	02	02
3	Index	01[1]	01[1]
4	Data 1	00	D9
5	Data 2	00	02
6	Data 3	00	00
7	Data 4	00	00

1) 01 = Analog input 1

Byte 4 – Data 1 (Low Byte): $D9_{hex}$

Byte 5 – Data 2 (High Byte): 02_{hex}

→ corresponding 16-bit value: $02D9_{hex}$ = 729 (7.29 V)

Local diagnostics: ID1 – ID16

The local diagnostics (ID1 – ID8) indicate the status of the individual NET stations. The connection to the remote station (only EZD) is indicated via ID9.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	03	03
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 49
5	Data 2 (High Byte)	00	→ table 49
6 – 7	Data 3 – 4	00	00

Table 49: Byte 4 to 5: Data 1 to 2

Data 1	Bit	7	6	5	4	3	2	1	0
ID1									0/1
ID2								0/1	
..					..				
ID8		0/1							
Data 2	Bit	7	6	5	4	3	2	1	0
ID9									0/1
–								1	
...					...				
–		1							

0/1= active/inactive NET station, –= not assigned

Example

Data 1 = F8, Data 2 = FF → In the EZ-NET network, the three stations are present with the NET IDs 1, 2, 3

Local inputs: IW0

This command string enables you to read the local inputs of the EZ800/EZD. The relevant input word is stored in Intel format.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	01	01
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 50
5	Data 2 (High Byte)	00	→ table 50
6 – 7	Data 3 – 4	00	00

Table 50: Byte 4 to 5: Data 1 to 2

Data 1	Bit	7	6	5	4	3	2	1	0
I1									0/1
I2								0/1	
..					..				
I8		0/1							
Data 2	**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I9									0/1
I10								0/1	
..					..				
I16		0/1							

Example: Read local inputs IW0

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response: Read successful	–	C2
1	Len	02	02
2	Type	01	01
3	Index	00	00
4	Data 1	00	C4
5	Data 2	00	02
6	Data 3	00	00
7	Data 4	00	00

→ All values must be transferred as hexadecimal values.

The values Data 1 = C4 and Data 2 = 02 indicate that the inputs I8, I7, I3 and I10 have been set to 1.

Inputs of the stations: IW1 to IW8

The EZ800 and EZD devices can be remotely expanded very simply using the EZ-NET. The service offered here makes it possible to implement read access to the inputs of individual NET stations.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	01	01
3	Index	01 – 08[1]	01 – 08[1]
4	Data 1 (Low Byte)	00	→ table 50 on page 138.
5	Data 2 (High Byte)	00	
6 – 7	Data 3 – 4	00	00

1) Corresponds to address of network station

Markers: M..

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command			
		Read	91	–
		Write	B1	–
	Response			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Len		→ table 51	→ table 51
2	Type			
3	Index			
4 – 7	Data 1 – 4			
		Read operation	00	→ Example 1: setting/resetting a marker bit on page 143
		Write operation	→ Example 2: write marker word on page 143	00

Table 51: Byte 1 to 3: Len, Type, Index

Operand				Len	Type	Index
Marker bit	M1	..	M96	01_{hex}	$0B_{hex}$	01 to 60_{hex}
Marker byte	MB1	..	MB96	01_{hex}	$0C_{hex}$	01 to 60_{hex}
Marker word	MW1	..	MW96	02_{hex}	$0D_{hex}$	01 to 60_{hex}
Marker double word	MD1	..	MD96	04_{hex}	$0E_{hex}$	01 to 60_{hex}

If required, refer to the more detailed description of the marker allocation in the EZ800 manual. Only a small extract of this manual is shown at this point in order to illustrate the allocation principle.

Attention!
The function blocks and DW markers (32-bit values) of EZ800/EZD operate with signed values.

Applies to MD, MW, MB, M	Left = Most significant bit, byte, word			Right = Least significant bit, byte, word
32 bit	MD1			
16 bit	MW2		MW1	
8 bit	MB4	MB3	MB2	MB1
1 bit	M32 to M25	M24 to M17	M16 to M9	M8 to M1
32 bit	MD2			
16 bit	MW4		MW3	
8 bit	MB8	MB7	MB6	MB5
1 bit	M64 to M57	M56 to M49	M48 to M41	M40 to M33

→ The relevant marker values are transferred in Intel format. In other words, the first byte is the low byte (Byte 4) and the last byte the high byte.

Example 1: setting/resetting a marker bit

Marker bit 62 is to be set or reset. To set the marker bit write a 1 in the least significant bit of the Data 1 and a 0 to reset it.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	B1	–
	Response: Write successful	–	C1
1	Len	01	01
2	Type	0B	0B
3	Index	3E	3E
4	Data 1	01/00[1)]	00
5 – 7	Data 2 – 4	00	00

1) 01 = set, 00 = reset

Example 2: write marker word

The value 823 is to be written to the marker word MW32:
$823_{dec} = 337_{hex}$ → Data 1 = 37_{hex}, Data 2 = 03_{hex}

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	B1	–
	Response: Write successful	–	C1
1	Len	01	01
2	Type	0D	0D
3	Index	20	20
4	Data 1	37	00
5	Data 2	03	00
6	Data 3	00	00
7	Data 4	00	00

Local P buttons: P1 – P4

The local P buttons are the display cursor buttons of the EZ800/EZD basic unit. You can scan the buttons in both RUN and STOP mode.

→ Ensure that the P buttons are also activated via the SYSTEM menu (in the basic unit).

Only one byte has to be transferred for the P buttons.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	06	06
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 52
5 – 7	Data 2 – 4	00	00

Table 52: Byte 4: Data

Data 1	Bit 7	6	5	4	3	2	1	0
P1								0/1
P2							0/1	
P3						0/1		
P4					0/1			
–				0				
–			0					
–		0						
–	0							

Local analog output: QA1

The commands provided can be used to access the local analog output of the EZ800 or EZD basic unit. When writing to the analog output, however, the value will only be output externally if the device concerned is in RUN mode and the image concerned has not been overwritten by the actual program. → section "Read/write image data" on page 133.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	91	–
	Write	B1	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	02	02
2	Type	05	05
3	Index	00	00
4 – 5	Data 1 – 2		
	Read operation	00	→ Example
	Write operation	→ Example	00
6 – 7	Data 3 – 4	00	00

Example:
The analog output is to output a value of approx. 5 V.

$500 = 01F4_{hex}$ Byte 4 – Data 1 (Low Byte): $F4_{hex}$
Byte 5 – Data 2 (High Byte): 01_{hex}

Local outputs: QW0/ outputs of the stations QW1 – QW8

The local outputs can be read and written directly via CANopen. However, the outputs are only switched externally if the device is in RUN mode and the addressed output is not being used in the circuit diagram. → section "Read/write image data" on page 133.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	91	–
	Write	B1	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	02	02
2	Type	04	04
3	Index[1)]	00/01 – 08	00/01 – 08
4	Data 1		
	Read operation	00	→ table 49
	Write operation	→ table 53	00
5 – 7	Data 2 – 4	00	00

1) 00 = Local output
01 – 08 = Outputs of network stations 1 – 8

Table 53: Byte 4: Data

Data 1	Bit	7	6	5	4	3	2	1	0
Q1									0/1
Q2								0/1	
Q3							0/1		
Q4						0/1			
Q5					0				
Q6				0					
Q7			0						
Q8		0							

Inputs/outputs of EZ-LINK: RW/SW

This service allows you to read the local R and S data and the data of the NET stations (1 – 8) transferred via EZ-LINK, again from the relevant EZ800/EZD image.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	For RW: 07	For RW: 07
		For SW: 09	For SW: 09
3	Index	00/01 – 08[1]	00/01 – 08[1]
4	Data 1 (Low Byte)	00	→ table 54
5	Data 2 (High Byte)	00	→ table 54
6 – 7	Data 3 – 4	00	00

1) 00 = Local input/output
01 – 08 = Address of network station (NET-ID 1 – 8)

Table 54: Byte 4 to 5: Data 1 to 2

Data 1		Bit	7	6	5	4	3	2	1	0
RW	SW									
R1	S1									0/1
R2	S2								0/1	
R3	S3							0/1		
R4	S4						0/1			
R5	S5					0/1				
R6	S6				0/1					
R7	S7			0/1						
R8	S8		0/1							
Data 2		Bit	7	6	5	4	3	2	1	0
R9	–									0/1
R10	–								0/1	
R11	–							0/1		
R12	–						0/1			
R13	–					0/1				
R14	–				0/1					
R15	–			0/1						
R16	–		0/1							

Receive Data Network: RN1 – RN32/ Transmit Data Network: SN1 – SN32

EZ-NET allows a point-to-point connection to be implemented between the individual NET stations. The RN and SN data are used for the data exchange (see the EZ800 manual).

→ The RN SN data of the local device (Index = 0) to which the EZ221-CO is fitted cannot be scanned. In this case the command would be denied with the $0C_{hex}$ signal.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	04	04
2	Type	For RN1 – RN32: 08	
		For SN1 – SN32: 0A	
3	Index	01 – 08[1]	01 – 08[1]
4 – 7	Data 1 – 4	00	→ table 55

1) Corresponds to NET-ID

Table 55: Byte 4 to 7: Data 1 to 4

Data 1		Bit	7	6	5	4	3	2	1	0
RN1	SN1					...				0/1
...									0/1	
RN8	SN8		0/1							
Data 2		**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RN9	SN9									0/1
....						...				
RN16	SN16		0/1							
Data 3		**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RN17	SN17									0/1
...						...				
RN24	SN24		0/1							
Data 4		**Bit**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RN25	SN25									0/1
...						...				
RN32	SN32		0/1							

Read/write function block data

→ Please also note the relevant description of the function blocks provided in the EZ800 manual.

General notes

Always note the following when working with function blocks:

- The relevant data is transferred in Intel format. In other words, the first byte is the low byte (Byte 4) and the last byte (byte 7) the high byte.
- The maximum data length is 4 bytes. All values must be transferred in hexadecimal format.
- All 32-bit values are treated as signed values. When transferring 32-bit values, ensure that the appropriate value range is suitable for long integers, i.e. signed.
  32-bit value: –2147483648 .. 0 .. +2147483647

Overview

Operands	Meaning	Read/write	Type (hex)	Page
A01 – A32	Analog value comparators: A01 – A32	Read/write	11	155
AR01 – AR32	Arithmetic function block: AR01 – AR32	Read/write	12	157
BC01 – BC32	Block compare: BC01 – BC32	Read/write	25	159
BT01 – BT32	Block transfer: BT01 – BT32	Read/write	26	161
BV01 – BV32	Boolean operation: BV01 – BV32	Read/write	13	163
C01 – C32	Counters: C01 – C32	Read/write	14	165
CF01 – CF04	Frequency counters: CF01 – CF04	Read/write	15	167
CH01 – CH04	High-speed counters: CH01 – CH04	Read/write	16	169
CI01 – CI02	Incremental counters: CI01 – CI02	Read/write	17	171
CP01 – CP32	Comparators: CP01 – CP32	Read/write	18	173
D01 – D32	Text output function blocks: D01 – D32	Read/write	19	175
DB01 – DB32	Data function blocks: DB01 – DB32	Read/write	1A	178
DC01 – DC32	PID controllers: DC01 – DC32	Read/write	27	180
FT01 – FT32	Signal smoothing filters: FT01 – FT32	Read/write	28	183
GT01 – GT32	Receive network data: GT01 – GT32	Read	1B	185
HW01 – HW32	7-day time switches: HW01 – HW32	Read	1C	187
HY01 – HY32	Year time switches: HY01 – HY32	Read	1D	190
LS01 – LS32	Value scaling: LS01 – LS32	Read/write	29	193
MR01 – MR32	Master reset: MR01 – MR32	Read	0F	195
NC01 – NC32	Numerical converters: NC01 – NC32	Read/write	2A	197
OT01 – OT04	Operating hours counters: OT01 – OT04	Read/write	1E	199
PT01 – PT32	Transmit network data: PT01 – PT32	Read	1F	201
PW01 – PW02	Pulse width modulation: PW01 – PW02	Read/write	2B	203
SC01	Synchronize clock: SC01	Read	20	205
ST01	Set cycle time: ST01	Read/write	2C	206
T01 – T32	Timing relays: T01 – T32	Read/write	21	208
VC01 – VC32	Value limitation: VC01 – VC32	Read/write	2D	211

Analog value comparators: A01 – A32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	11	11
2	Instance	01 – 20	01 – 20
3	Index	→ table 56	→ table 56
4 – 7	Data 1 – 4	00	depending on index, → table 57, 58

Table 56: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 57		×	
01	Mode, → table 58		×	
02	Comparison value 1	I1	×	c[1]
03	Gain factor for I1 (I1 = F1 × value)	F1	×	c[1]
04	Comparison value 2	I2	×	c[1]
05	Gain factor for I2 (I2 = F2 × value)	F2	×	c[1]
06	Offset for value I1	OS	×	c[1]
07	Switching hysteresis for value I2 (the value of HY is for both positive and negative hysteresis.)	HY	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 7 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 57: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	CY[1]	Q1[2]

1) Status 1 if the value range is exceeded
2) Status 1 if the condition is fulfilled (e.g. I1 < I2 with LT mode)

Table 58: Index 1 - Mode

Data 1 (hex)		
00	LT	Less than (I1 < I2)
01	EQ	Equal to (I1 = IGT)
02	GT	Greater than (I1 > I2)

Arithmetic function block: AR01 – AR32

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	12	12
2	Instance	01 – 20	01 – 20
3	Index	→ table 59	→ table 59
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 60, 61
	Write operation	depending on index, → table 60, 61	00

Table 59: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 60		×	
01	Mode, → table 61		×	
02	First operand	I1	×	c[1]
03	Second operand	I2	×	c[1]
04	Result	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 4 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 60: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1
FB output Data 3		–	–	–	–	–	ZE[1]	CY[2]

1) Status 1 if the value of the function block output QV (the calculation result) equals zero
2) Status 1 if the value range is exceeded

Table 61: Index 1 - Mode

Data 1 (hex)		
00	ADD	Add (I1 + I2 = QV)
01	SUB	Subtract (I1 – I2 = QV)
02	MUL	Multiply (I1 × I2 = QV)
03	DIV	Divide (I1 : I2 = QV)

Block compare: BC01 – BC32

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	25	25
2	Instance	01 – 20	01 – 20
3	Index	→ table 62	→ table 62
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 63, 64
	Write operation	depending on index, → table 63, 64	00

Table 62: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 63		×	
01	Mode, → table 64		×	
02	Source range 1	I1	×	c[1]
03	Target range 2	I2	×	c[1]
04	Number of elements to compare: 8 (max. 192 bytes)	NO	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 4 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 63: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN[1]
FB output Data 3		–	–	–	–	EQ[2]	E3[3]	E2[4]	E1[5]

1) Activates the function block on status 1.
2) Status 1 if the data ranges are equal; status 0 if not equal

Error outputs
3) Status 1 if the number of elements exceeds the source or target range.
4) Status 1 if the source and target range overlap.
5) Status 1 if the source or target range are outside of the available marker range (offset error)

Table 64: Index 1 - Mode

Mode	Data 1 (hex)	Operating mode
	02	Compare (internal EZ status signal for Block Compare mode)

Block transfer: BT01 – BT32

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command:			
		Read	92	–
		Write	B2	–
	Response:			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Type		26	26
2	Instance		01 – 20	01 – 20
3	Index		→ table 65	→ table 65
4 – 7	Data 1 – 4			
		Read operation	00	depending on index, → table 66, 67
		Write operation	depending on index, → table 66, 67	00

Table 65: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 66		×	
01	Mode, → table 67		×	
02	Source range 1	I1	×	c[1]
03	Target range 2	I2	×	c[1]
04	Number of elements to compare: max. 192 bytes	NO	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

→ The data for Index 2 and 3 is transferred as a 32-bit value in Intel format (Low Byte first) (Data 1 – Low Byte .. Data 2 - High Byte).

Table 66: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1]
FB output Data 3		–	–	–	–	–	E3[2]	E2[3]	E1[4]

1) Transfer of the source address specified at I1 to the target address specified at I2 on rising edge.

Error outputs

2) Status 1 if the number of elements exceeds the source or target range.
3) Status 1 if the source and target range overlap.
4) Status 1 if the source or target range are outside of the available marker range (offset error)

Table 67: Index 1 - Mode

Data 1 (hex)	Operating mode
00	INI: Initializes the target range with a byte value stored at the source address.
01	CPY: Copies a data block from a source to a target range. Data block size is specified at NO.

Boolean operation: BV01 – BV32

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	13	13
2	Instance	01 – 20	01 – 20
3	Index	→ table 68	→ table 68
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 69, 70
	Write operation	depending on index, → table 69, 70	00

Table 68: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 69		×	
01	Mode, → table 70		×	
02	First operand	I1	×	c[1]
03	Second operand	I2	×	c[1]
04	Operation result	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 4 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 69: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1
FB output Data 3		–	–	–	–	–	–	ZE[1]

1) Status 1 if the value of the function block output QV (the operation result) equals zero

Table 70: Index 1 - Mode

Data 1 (hex)		
00	AND	And operation
01	OR	Or operation
02	XOR	Exclusive Or operation
03	NET	Inverts the individual bits of the value at I1. The inverted value is represented as a signed decimal value.

Counters: C01 – C32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	14	14
2	Instance	01 – 20	01 – 20
3	Index	→ table 71	→ table 71
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 72
	Write operation	depending on index, → table 72	00

Table 71: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 72	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2147483648 to +2147483647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 5 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 72: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	SE[1]	D[2]	C[3]	RE[4]
FB output Data 3		–	–	–	–	ZE[5]	CY[6]	FB[7]	OF[8]

1) Transfer preset actual value on rising edge
2) Count direction: 0 = up counting, 1 = down counting
3) Count coil, counts on every rising edge
4) Reset actual value to zero
5) Zero: Status 1 if the value of the function block output QV (the counter status) equals zero
6) Carry: Status 1 if the value range is exceeded
7) Fall below: Status 1 if the actual value ≦ lower setpoint
8) Overflow: Status 1 if the actual value ≧ upper setpoint

Frequency counters: CF01 – CF04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	15	15
3	Instance	01 – 04	01 – 04
4	Index	→ table 73	→ table 73
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 74
	Write operation	depending on index, → table 74	00

Table 73: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 74		×	
01	Mode/Parameters		–	–
02	Upper setpoint	SH	×	c[1]
03	Lower setpoint	SL	×	c[1]
04	Actual value in RUN mode	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 4 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 74: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN[1]
FB output Data 3		–	–	–	–	–	ZE[2]	FB[3]	OF[4]

1) Counter enable
2) Zero: Status 1 if the value of the function block output QV (the counter status) equals zero
3) Fall below: Status 1 if the actual value ≦ lower setpoint
4) Overflow: Status 1 if the actual value ≧ upper setpoint

High-speed counters: CH01 – CH04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	16	16
2	Instance	01 – 04	01 – 04
3	Index	→ table 75	→ table 75
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 76
	Write operation	depending on index, → table 76	00

Table 75: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 76	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2147483648 to +2147483647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 5 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 76: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	EN[1]	SE[2]	D[3]	RE[4]
FB output Data 3		–	–	–	–	ZE[5]	CY[6]	FB[7]	OF[8]

1) Counter enable
2) Transfer preset actual value on rising edge
3) Count direction: 0 = up counting, 1 = down counting
4) Reset actual value to zero
5) Zero: Status 1 if the value of the function block output QV (the counter status) equals zero
6) Carry: Status 1 if the value range is exceeded
7) Fall below: Status 1 if the actual value ≦ lower setpoint
8) Overflow: Status 1 if the actual value ≧ lower setpoint

Incremental counters: CI01 – CI02

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	17	17
2	Instance	01 – 02	01 – 02
3	Index	→ table 77	→ table 77
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 78
	Write operation	depending on index, → table 78	00

Table 77: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 78	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2147483648 to +2147483647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 5 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 78: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	EN[1]	SE[2]	RE[3]
FB output Data 3		–	–	–	–	ZE[4]	CY[5]	FB[6]	OF[7]

1) Counter enable
2) Transfer preset actual value on rising edge
3) Reset actual value to zero
4) Zero: Status 1 if the value of the function block output QV (the counter status) equals zero
5) Carry: Status 1 if the value range is exceeded
6) Fall below: Status 1 if the actual value ≦ lower setpoint
7) Overflow: Status 1 if the actual value ≧ lower setpoint

Comparators: CP01 – CP32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	18	18
2	Instance	01 – 20	01 – 20
3	Index	→ table 79	→ table 79
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 80
	Write operation	depending on index, → table 80	00

Table 79: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 80		×	
01	Mode/Parameters		–	–
02	Comparison value	I1	×	c[1]
03	Comparison value	I2	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 and 3 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 80: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1
FB output Data 3		–	–	–	–	GT[1]	EQ[2]	LT[3]

1) greater than: Status 1 if the value at I1 is greater than value at I2 (I1 > I2)
2) equal: Status 1 if the value at I1 is equal to value at I2 (I1 = I2)
3) less than: Status 1 if the value at I1 is less than value at I2 (I1 < I2)

Text output function blocks: D01 – D32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	19	19
2	Instance	01 – 20	01 – 20
3	Index	→ table 81	→ table 81
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 82
	Write operation	depending on index, → table 82	00

Table 81: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 82	×	
01	Mode/Parameters	–	–
02	Text line 1, column 1 - 4	×	
03	Text line 1, column 5 - 8	×	
04	Text line 1, column 9 - 12	×	
05	Text line 1, column 13 - 16	×	
06	Text line 2, column 1 - 4	×	
07	Text line 2, column 5 - 8	×	
08	Text line 2, column 9 - 12	×	
09	Text line 2, column 13 - 16	×	
10	Text line 3, column 1 - 4	×	
11	Text line 3, column 5 - 8	×	
12	Text line 3, column 9 - 12	×	
13	Text line 3, column 13 - 16	×	
14	Text line 4, column 1 - 4	×	
15	Text line 4, column 5 - 8	×	
16	Text line 4, column 9 - 12	×	
17	Text line 4, column 13 - 16	×	
18	Variable 1	×	c[1]
19	Variable 2	×	c[1]
20	Variable 3	×	c[1]
21	Variable 4	×	c[1]
22	Scaling minimum value 1	×	
23	Scaling minimum value 2	×	
24	Scaling minimum value 3	×	
25	Scaling minimum value 4	×	
26	Scaling maximum value 1	×	

Index (hex)	Operand	Read	Write
27	Scaling maximum value 2	×	
28	Scaling maximum value 3	×	
29	Scaling maximum value 4	×	
30	Control information line 1	×	
31	Control information line 2	×	
32	Control information line 3	×	
33	Control information line 4	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The variables 1 to 4 (index 18 to 21) are transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 82: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN[1]
FB output Data 3		–	–	–	–	–	–	–	Q1[2]

1) Text function block enable
2) Status 1, text function block is active

Data function blocks: DB01 – DB32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	1A	1A
2	Instance	01 – 20	01 – 20
3	Index	→ table 83	→ table 83
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 84
	Write operation	depending on index, → table 84	00

Table 83: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 84		×	
01	Mode/Parameters		–	–
02	Input value: value that is transferred to the QV output when the FB is triggered.	I1	×	c[1]
03	Output value	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 and 3 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 84: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1]
FB output Data 3		–	–	–	–	–	–	–	Q1[2]

1) Transfer of the value present at I1 on rising edge.
2) Status 1 if the trigger signal is 1.

PID controllers: DC01 – DC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	27	27
2	Instance	01 – 20	01 – 20
3	Index	→ table 85	→ table 85
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 86, 87
	Write operation	depending on index, → table 86, 87	

Table 85: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 86		×	
01	Mode, → table 87		×	
02	Setpoint: –32768 to +32767	I1	×	c[1]
03	Actual value: –32768 to +32767	I2	×	c[1]
04	Proportional gain [%], Value range: 0 to 65535	KP	×	c[1]
05	Reset time [0.1 s], Value range: 0 to 65535	TN	×	c[1]
06	Rate time [0.1 s], Value range: 0 to 65535	TV	×	c[1]
07	Scan time = Time between function block calls. Value range: 0.1s to 6553.5s. If 0 is entered as the value, the scan time will be determined by the program cycle time.	TC	×	c[1]
08	Manual manipulated variable, value range: –4096 to +4095	MV	×	c[1]
09	Manipulated variable • Mode: UNI, value range: 0 to +4095 (12 bit) • Mode: BIP, value range: –4096 to +4095 (13 bit)	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for Index 2 and 9 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte .. Data 2 – High Byte).

Table 86: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	SE[1]	ED[2]	EI[3]	EP[4]	EN[5]
FB output Data 3		–	–	–	–	–	–	–	LI[6]

1) Transfer of manual manipulated variable on status 1
2) Activation of D component on status 1
3) Activation of I component on status 1
4) Activation of P component on status 1
5) Activates the function block on status 1.
6) Status 1 if the value range of the medium-voltage was exceeded

Table 87: Index 1 - Mode

Data 1	Operating mode
UNP unipolar	The manipulated variable is output as a unipolar 12-bit value. Corresponding value range for QV 0 to 4095.
BIP bipolar	The manipulated variable is output as a bipolar 13-bit value. Corresponding value range for QV –4096 to 4095

Signal smoothing filters: FT01 – FT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	28	28
2	Instance	01 – 20	01 – 20
3	Index	→ table 88	→ table 88
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 89
	Write operation	depending on index, → table 89	00

Table 88: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 89		×	
01	Mode/Parameters		–	–
02	Input value, value range: –32768 to +32767	I1	×	c[1]
03	Recovery time [0.1 s], Value range: 0 to 65535	TG	×	c[1]
04	Proportional gain [%], Value range: 0 to 65535	KP	×	c[1]
05	Delayed output value, value range: –32768 to +32767	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

Table 89: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN[1]

1) Activates the function block on status 1.

Receive network data: GT01 – GT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1B	1B
2	Instance	01 – 20	01 – 20
3	Index	→ table 90	
4 – 7	Data 1 – 4	00	depending on index, → table 91, 92

Table 90: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 91	×	
01	Mode/Parameters, → table 92	×	–
02	Output value: actual value from the network QV	×	

→ The data for index 2 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 91: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	Q[1]

1) Status 1 if a new value is present that is transferred from the NET network.

Table 92: Index 1 – Mode/Parameters (designation of PUT FB with data to be received)

Mode	Data 1	NET-ID[1]	
		0	NET-ID 1
		..	..
		7	NET-ID 8
Parameters	Data 3	Instance[2]	
		0	PT01
		..	..
		31	PT32

1) Number of station transmitting the value. Possible station numbers: 01 to 08
2) Transmit FB (e.g. PT 20) of the transmitting NET station. Possible station numbers: 01 – 32

7-day time switches: HW01 – HW32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1C	1C
2	Instance	01 – 20	01 – 20
3	Index	→ table 93	
4 – 7	Data 1 – 4	00	depending on index, → table 94

Table 93: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 94	×	
01	Mode/Parameters	–	–
02	Parameters → table 95 Channel A	×	
03	Channel B		
04	Channel C		
05	Channel D		

Table 94: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	Q[1]

1) Status 1 if the switch-on condition is fulfilled.

The data in the following table is shown in the Motorola format although it is actually transferred in Intel format.

Table 95: Index 2 to 5, Parameter channels A to D

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 2								Date 1							
ON	d4	d3	d2	d1	d0	h4	h3	h2	h1	h0	m5	m4	m3	m2	m1	m0
	Weekday					Hour					Minute					

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 4								Date 3							
OFF	d4	d3	d2	d1	d0	h4	h3	h2	h1	h0	m5	m4	m3	m2	m1	m0
	Weekday					Hour					Minute					

m5 to m0: Minute (0 to 59)
h4 to h0: Hour (0 to 23)
d5 to d0: Weekday (0 = Sunday to 6 = Saturday)

Example
The channel A parameters of 7-day time switch HW19 are to be read.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response: Read successful	–	C2
1	Type	1C	1C
2	Instance	13	13
3	Index	02	02
4	Data 1	00	62
5	Data 2	00	0B
6	Data 3	00	7B
7	Data 4	00	25

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 2 = $0B_{hex}$								Date 1 = 62_{hex}							
ON	0	0	0	0	1	0	1	1	0	1	1	0	0	0	1	0
	Weekday					Hour					Minute					

Switch-on time:
Weekday = 01_{hex} .. Monday
Hour = $0D_{hex}$ .. 1300 hours
Minute = 22_{hex} .. 34 Minutes

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 4 = 25_{hex}								Date 3 = $7B_{hex}$							
OFF	0	0	1	0	0	1	0	1	0	1	1	1	1	0	1	1
	Weekday					Hour					Minute					

Switch-off time:
Weekday = 04_{hex} .. Thursday
Hour = 15_{hex} .. 2100 hours
Minute = 59_{hex} .. 34 minutes

Year time switches: HY01 – HY32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1D	1D
2	Instance	01 – 20	01 – 20
3	Index	→ table 96	
4 – 7	Data 1 – 4	00	depending on index, → table 97

Table 96: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 97	×	
01	Mode/Parameters	–	–
02	Parameters → table 98 Channel A	×	
03	Channel B		
04	Channel C		
05	Channel D		

Table 97: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	Q[1]

1) Status 1 if the switch-on condition is fulfilled.

The data in the following table is shown in the Motorola format although it is actually transferred in Intel format.

Table 98: Index 2 to 5, parameter channels A to D

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 2								Date 1							
ON	y6	y5	y4	y3	y2	y1	y0	m3	m2	m1	m0	d4	d3	d2	d1	d0
	Year							Month				Day				

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 4								Date 3							
OFF	y6	y5	y4	y3	y2	y1	y0	m3	m2	m1	m0	d4	d3	d2	d1	d0
	Year							Month				Day				

d4 ... d0: Day (1 .. 31), m3 ... m0: Month (1 .. 12), y6 ... y0: Year (0: 2000 .. 99: 2099)

Example

The channel A parameters of year time switch HY14 are to be written.

Index 2 – 5, Parameter channels A – D

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 2								Date 1							
ON	0	0	0	0	0	1	1	0	1	1	0	0	1	1	1	0
	Year							Month				Day				

Switch-on time:
Day = 14 = $0E_{hex}$ = 0000 1110b
Month = 6 (June) = 06_{hex} = 0000 0110b
Year = 2003 = 03_{hex} = 0000 0011b

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

Index 2 – 5, Parameter channels A – D

Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
	Date 2								Date 1							
OFF	y6	y5	y4	y3	y2	y1	y0	m3	m2	m1	m0	d4	d3	d2	d1	d0
	Year							Month				Day				

Switch-off time:
Day = 3 = 03_{hex} = 0000 0011b
Month = 10 (October) = $0A_{hex}$ = 0000 1010b
Year = 2012 = $0C_{hex}$ = 0000 1100b

Resulting telegram:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Write	B2	–
	Response: Write successful	–	C1
1	Type	1D	1D
2	Instance	0E	0E
3	Index	02	02
4	Data 1	8E	00
5	Data 2	06	00
6	Data 3	43	00
7	Data 4	19	00

Value scaling: LS01 – LS32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	29	29
2	Instance	01 – 20	01 – 20
3	Index	→ table 99	→ table 99
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 100
	Write operation	depending on index, → table 100	

Table 99: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 100		×	
01	Mode/Parameters		–	–
02	Input value, value range: 32 bit	I1	×	c[1]
03	Interpolation point 1, X coordinate, value range: 32 bit	X1	×	c[1]
04	Interpolation point 1, Y coordinate, value range: 32 bit	Y1	×	c[1]
05	Interpolation point 2, X coordinate, value range: 32 bit	X2	×	c[1]
06	Interpolation point 2, Y coordinate, value range: 32 bit	Y2	×	c[1]
07	Output value: contains the scaled input value	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

Table 100: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN[1]

1) Activates the function block on status 1.

Master reset: MR01 – MR32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	0F	0F
2	Instance	01 – 20	01 – 20
3	Index		
	Bit IO	00	00
	Mode	01	01
4 – 7	Data 1 – 4	00	depending on index, → table 101, 102

Table 101: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1]
FB output Data 3		–	–	–	–	–	–	–	Q1[2]

1) Trigger coil. The appropriate Reset is executed if the coil is triggered (with a rising edge).
2) Status 1 if the trigger coil MR..T is 1.

Table 102: Index 1 - Mode

Data 1 (hex)		
00	Q	Outputs Q.., *Q.., S.., *S.., *SN.., QA01 are reset to 0. * depending on the NET-ID
01	M	The marker range MD01 to MD48 is reset to 0.
02	ALL	Has an effect on Q and M.

Numerical converters: NC01 – NC32

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2A	2A
2	Instance	01 – 20	01 – 20
3	Index	→ table 103	→ table 103
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 104, 105
	Write operation	depending on index, → table 104, 105	00

Table 103: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 104		×	
01	Mode, → table 105		×	
02	Input value: operand to be converted	I1	×	c[1]
03	Output value: contains the conversion result	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for Index 2 and 3 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte .. Data 2 – High Byte).

Table 104: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN[1]

1) Activates the function block on status 1.

Table 105: Index 1 - Mode

Data 1 (hex)		
00	BCD	Converts a BCD coded decimal value to an integer value.
01	BIN	Converts an integer value to a BCD coded decimal value.

Operating hours counters: OT01 – OT04

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	1E	1E
2	Instance	01 – 04	01 – 04
3	Index	→ table 106	→ table 106
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 107
	Write operation	depending on index, → table 107	00

Table 106: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 107		×	
01	Mode/Parameters		–	–
02	Upper threshold value	I1	×	c[1]
03	Actual value of operating hours counter	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

Table 107: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	RE[1]	EN[2]
FB output Data 3		–	–	–	–	–	–	–	Q1[3]

1) Reset coil: Status 1 resets the counter actual value to zero.
2) Enable coil
3) Status 1 if the setpoint was reached (greater than/equal to)

→ The data for index 2 and 3 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Transmit network data: PT01 – PT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1F	1F
2	Instance	01 – 20	01 – 20
3	Index	→ table 108	
4 – 7	Data 1 – 4	00	depending on index, → table 109

Table 108: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 109		×	
01	Mode/Parameters		–	–
02	Input value: Setpoint that it transmitted to the NET network	I1	×	

→ The data for index 2 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 109: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1]
FB output Data 3		–	–	–	–	–	–	–	Q1[2]

1) Trigger coil. The value is provided on the NET if the coil is triggered (with a rising edge).
2) Status 1 if the trigger coil PT..T_ is also 1.

Pulse width modulation: PW01 – PW02

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2B	2B
2	Instance	01 – 02	01 – 02
3	Index	→ table 110	→ table 110
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 111
	Write operation	depending on index, → table 111	00

Table 110: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 111		×	
01	Mode/Parameters		–	–
02	Manipulated variable, value range: 0 to 4095 (12 bit)	SV	×	c[1]
03	Period duration [ms], Value range: 0 to 65535	PD	×	c[1]
04	Minimum on duration [ms], Value range: 0 to 65535	ME	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

Table 111: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN[1]
FB output Data 3		–	–	–	–	–	–	–	E1[2]

1) Activates the function block on status 1.

2) Status 1 if below the minimum on duration or minimum off duration

Synchronize clock: SC01

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	20	20
2	Instance	01	01
3	Index	→ table 112	
4 – 7	Data 1 – 4	00	depending on index, → table 113

Table 112: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 113	×	
01	Mode/Parameters	–	–

Table 113: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1]
FB output Data 3		–	–	–	–	–	–	–	Q1[2]

1) Trigger coil. If the coil is triggered (rising edge), the current date, weekday and time of the transmitting station are automatically sent to the NET network.

2) Status 1 if the trigger coil SC01T_ is also 1.

Set cycle time: ST01

Telegram structure

Byte	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2C	2C
2	Instance	01	01
3	Index	→ table 114	→ table 114
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 115
	Write operation	depending on index, → table 115	00

Table 114: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 115		×	
01	Mode/Parameters		–	–
02	Cycle time in ms, value range: 0 – 1000	I1	×	c[1]

1) The value can only be written if it is assigned to a constant in the program.

Table 115: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN[1]

1) Activates the function block on status 1.

Timing relays: T01 – T32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	21	21
2	Instance	01 – 20	01 – 20
3	Index	→ table 116	→ table 116
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 117, 118
	Write operation	depending on index, → table 117, 118	

Table 116: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 117		×	
01	Mode/Parameters, → table 118		×	
02	Setpoint 1: Time setpoint 1	I1	×	c[1]
03	Setpoint 2: Time setpoint 2 (with timing relay with 2 setpoints)	I2	×	c[1]
04	Actual value: Time elapsed in RUN mode	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

→ The data for index 2 to 4 is transferred as a 32-bit value in Intel format (Data 1 – Low Byte to Data 4 – High Byte).

Table 117: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	ST[1]	EN[2]	RE[3]
FB output Data 3		–	–	–	–	–	–	–	Q1[4]

1) Stop, the timing relay is stopped (Stop coil)
2) Enable, the timing relay is started (trigger coil)
3) Reset, the timing relay is reset (reset coil)
4) Switch contact

Table 118: Index 1 - Mode/Parameters

Mode	Data 1	Operating mode
	0	On-delayed
	1	On-delayed with random setpoint
	2	Off-delayed
	3	Off-delayed with random setpoint
	4	On and off delayed (two time setpoints)
	5	On and off delayed each with random setpoint (two time setpoints)
	6	Impulse transmitters
	7	Flashing relay (two time setpoints)
	8	Off-delayed, retriggerable (not used)
	9	Off-delayed with random setpoint, retriggerable (not used)
Para-meters	Data 3	Operating mode
	0	S (milliseconds)
	1	M:S (seconds)
	2	H:M (minutes)

Value limitation: VC01 – VC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2D	2D
2	Instance	01 – 20	01 – 20
3	Index	→ table 119	→ table 119
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 120
	Write operation	depending on index, → table 120	00

Table 119: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 120		×	
01	Mode/Parameters		–	–
02	Input value	I1	×	c[1]
03	Upper limit value	SH	×	c[1]
04	Lower limit value	SL	×	c[1]
05	Output value: outputs the value present at input I1 within the set limits.	QV	×	

1) The value can only be written if it is assigned to a constant in the program.

Table 120: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN[1]

1) Activates the function block on status 1.

Analysis – error codes via EZ-LINK

The EZ800/EZD basic unit will return a defined error code in the event of an incorrectly selected operating mode or an invalid telegram. The error code transferred has the following structure:

Telegram structure

Byte	Meaning	Slave transmits (value hex)
0	Response	
	Command rejected	C0
1	Type	
2	Instance	
3	Index	
4	Error code	→ table 121
5 – 7	Data 2 – 4	

Table 121: Error codes

Error code	Description
0x00	No error
0x03	Formal error in the response related to type, instance or index
0x04	No communication possible (Timeout)
0x05	DP module has only transmitted 0xC0 (EZ800/EZD).
0x45	The value selected by Type and Index must not be overwritten (Bit IO, Mode/Parameters or output value).
0x46	The value selected by Type and Index is not assigned to a constant and cannot therefore be written.
0x9E	Access to the FB data not possible (program download active).
0x9F	Type is invalid (no defined FB, depending also on the version of the addressed device).
0xA0	FB selected by Type and Index does not exist in the program.
0xA1	Index related to the specified FB type is invalid.

What happens if...? RUN LED

Status of the RUN LED	Possible cause	To correct or avoid error
OFF	The EZ221-CO is either switched off or is currently being reset.	Switch on the EZ221-CO and supply with mains voltage.
Flickering	Auto baud recognition is currently busy (LED flickers, alternating with the ERR LED).	Check the communication of the master PLC or the bus.
Single flash	The device is in STOPPED state.	Change the status of NMT (network management), see Section 4.3
Flashing	The device is in PRE-OPERATIONAL state.	
ON	The device is in OPERATIONAL state.	

Error LED

Status of the error LED	Possible cause	To correct or avoid error
OFF	The EZ221-CO is operating error-free. If the RUN LED is also off, the EZ221-CO is either switched off or is currently being reset.	Switch on the power supply.
Single flash	At least one of the error counters of the CANopen PLC has either reached or exceeded the Warning Limit. Too many errors have occurred on the CANopen bus.	Check for external interference on the bus. EMC problems – is the shielding properly terminated? Is the correct baud rate set at the other nodes?
Flickering	Auto baud rate recognition is currently busy (flickers alternating with the RUN LED).	Check the communication of the master PLC or the bus.
Flashes twice	A protective Guard Event or a Heartbeat Event has occurred.	Check configuration data.
ON	The CANopen PLC has changed to BUS-OFF state.	Verify the correct setting of the NODE ID.

Technical Data

General		
Standards and regulations		EN 61000-6-1; EN 61000-6-2; EN 61000-6-3; EN 61000-6-4, IEC 60068-2-27, IEC 50178
Dimensions (W × H × D)	mm	35.5 × 90 × 56.5
Weight	g	150
Mounting		DIN 50022 rail, 35 mm Screw fixing with fixing bracket EZB4-101-GF1 (accessories)
Climatic environmental conditions (Cold to IEC 60068-2-1, Heat to IEC 60068-2-2)		
Ambient temperature Installed horizontally/vertically	°C	–25 to +55
Condensation		Prevent condensation with suitable measures
Storage/transport temperature	°C	–40 to +70
Relative humidity (IEC 60068-2-30), no moisture condensation	%	5 to 95
Air pressure (operation)	hPa	795 to 1080
Corrosion resistance (IEC 60068-2-42, IEC 60068-2-43)		SO_2 10 cm^3 /m^3, 4 days H_2S 1 cm^3 /m^3, 4 days
Ambient mechanical conditions		
Pollution degree		2
Degree of protection (EN 50178, IEC 60529, VBG4)		IP20
Vibration (IEC 60068-2-6)		
constant amplitude 0.15 mm	Hz	10 to 57
Constant acceleration 2 g	Hz	57 to 150
Shocks (IEC 60068-2-27) semi-sinusoidal 15 g/11 ms	Shocks	18
Drop (IEC 60068-2-31) height	mm	50
Free fall, when packed (IEC 60068-2-32)	m	1

Electromagnetic compatibility (EMC)		
Electrostatic discharge (ESD), (IEC/EN 61000-4-2, severity level 3)		
Air discharge	kV	8
Contact discharge	kV	6
Electromagnetic fields RFI), (IEC/EN 61000-3	V/m	10
Radio interference suppression (EN 55011, EN 55022), class		B
Burst (IEC/EN 61000-4-4, severity level 3)		
Power cables	kV	2
Signal cables	kV	2
High-energy pulses (surge) of EZ AC current (IEC/ EN 61000-4-5), power cable symmetrical	kV	1
High-energy pulses (surge) of EZ DC current (IEC/ EN 61 000-4-5, severity level 2), power cable symmetrical	kV	0.5
Line-conducted interference (IEC/EN 61000-4-6)	V	10
Dielectric strength		
Measurement of the clearance and creepage distance		EN 50178, UL508, CSA C22.2 No. 142
Dielectric strength		EN 50 178
Tools and cable cross-sections		
Conductor cross-sections		
Solid, minimum to maximum	mm^2	0.2 to 4
	AWG	22 to 12
Flexible with ferrule, minimum to maximum	mm^2	0.2 to 2.5
	AWG	22 to 12
Slot-head screwdriver, width	mm	3.5 × 0.8
Tightening torque	N/m	0.5

Power supply		
Rated voltage		
Rated value	V DC	24 (–15, +20)
Permissible range	V DC	20.4 to 28.8
Residual ripple	%	< 5
Input current at 24 V DC, typical	mA	200
Voltage dips, IEC/EN 61131-2	ms	10
Power dissipation at 24 V DC, typical	W	4.8
LEDs		
Module Status LED MS	Color	Green/red
Network Status LED NS	Color	Green/red
CANopen		
Device connection		8-pin RJ45 socket
Electrical isolation		Bus to power supply (basic) Bus and power supply to EZ/EZD basic unit (safe isolation)
Function		CANopen slave
Interface		CANopen (CAN)
Bus protocol		CANopen
Auto baud recognition max.	kbps	1000
Bus termination resistors		Separate installation at the bus possible
Bus addresses, accessible via EZ basic unit with display or EZSoft		1 to 127
Services		
Module inputs		All data S1 to S8
Module outputs		All data R1 to R16
Module control commands		Read/Write Time, day, summer/winter time All parameters of the EZ/EZD functions

Dimensions

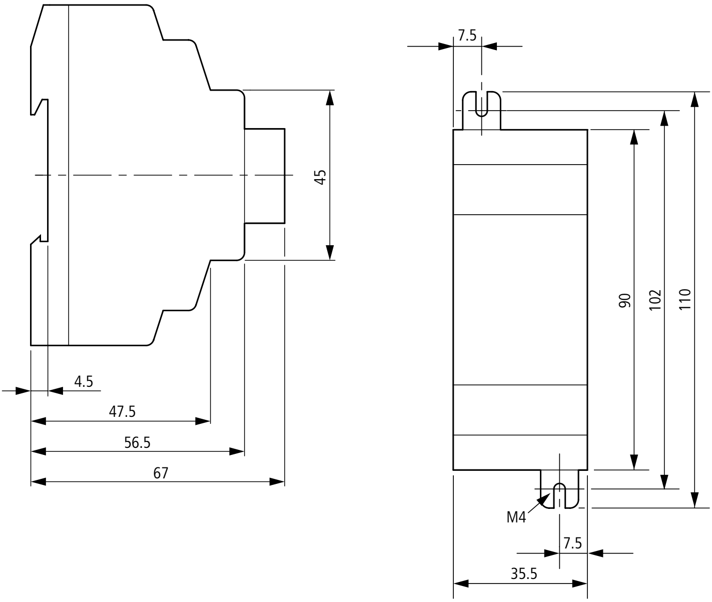


Figure 26: EZ221-CO dimensions in [mm]

Glossary

This glossary refers to the topics related to CANopen.

Access Type	Access rights to an object.
Acknowledge	Acknowledgement returned by the receiving station after having received a signal.
Active metallic component	Conductor or conductive component that is live when in operation.
Address	Number, for example, for identifying a memory location, a system or a module within a network.
Addressing	Assignment or setting of an address such as for a module in a network.
Analog	Value, such as voltage, that is infinitely variable and proportional. Analog signals can acquire any value within specific limits.
Arbitration	A bus access mechanism used by CANopen.
Auto Baud Recognition	Automatic recognition of the communication speed in a bus system, when at least two stations communicate or one station transmits messages across the communication bus.
Automation device	I/O controlling device that is interconnected to a system process. Programmable controllers (PLCs) are a special group of automation devices.
Basic CAN	Concept for the implementation of a CAN controller. All CAN messages are stored in an intermediate Tx and Rx buffer, that is, without causing high load on the host controller that has to evaluate all messages.
Baud	Unit for the data transfer rate. One baud is equivalent to the transmission of one bit per second (bps).
Baud rate	Unit of measure of the data transmission speed in bit/s.
Bidirectional	Operation in both directions.
Bit	Abbreviation for the term “binary digit”. Represents the smallest information unit of a binary system. Its significance can be 1 or 0 (Yes/No decision).

Bit Stuffing	Method used in CAN: After a sequence of five bits of the same polarity, a “stuff bit” with reversed polarity is inserted into the current message frame.
Bridge	A bridge connects the CANopen network to the electronic modules which represent the network slaves.
Bus	Bus system for data exchange, for example between the CPU, memory and I/O. A bus can consist of several parallel segments, such as the data bus, address bus, control bus and power supply bus.
Bus cycle time	Time interval in which a master will serve all slaves or stations in a bus system, i.e. writes their outputs and reads their inputs.
Bus line	Smallest unit connected to the bus. Consists of the PLC, a module and a bus interface for the module.
Bus system	The entirety of all units which communicate across a bus.
Bus terminating resistor	Resistor at the beginning and end of a bus line for preventing disturbance caused by signal reflections and for adapting bus cables. Bus terminating resistors must always be the last unit at the end of a bus segment.
Byte	A sequence of 8 bits
CAL	CAN Application Layer. Standardized Layer 7 Protocol according to CiA DS 201 to 207.
CAN	Controller Area Network
CAN 2.0A	11-bit identifier
CAN 2.0B	29-bit identifier
CAN high-speed	Up to 1 Mbps, normally 500 kbps
CAN low speed	max. 250 kbps
CAN nodes	In a CAN system, the network slaves are also referred to as CAN nodes.
CAN Transceiver	CAN controllers are interconnected to the bus medium by means of an ISO/DIS 11898 interface. The structure of this interface is usually not formed by a discrete circuit, but rather by a CAN Transceiver chip.

CANopen	Profile families based on CAL for high-speed data exchange. CiA standardizes the communication profile in CiA-DS-301.
Capacitive coupling	Capacitive (electrical) coupling develops between two conductors carrying different potentials. Typical interference sources are, for example parallel signal cables, contactor relays and static discharge.
Change of State	In CAN: The producer automatically and immediately sends its data when the position changes.
Chassis ground	Entirety of all interconnected inactive equipment parts that do not have any contact voltage, even in the event of a fault.
CiA	CiA e. V./CAN in Automation. International CAN manufacturer and user organization.
CiA DS	CAN in Automation Draft Standard, communication profile
CiA DSP	CAN in Automation Draft Standard Proposal
CMS	CAN Based Message Specification. One of the services of the application layer in the CAN Reference Model.
COB	Communication Object/CAN Message. A message in the CAN network. All data to be sent via CAN are transported in COBs.
COB-ID	COB identifier. Unambiguous identification of a COB in the entire CAN network. The COB-ID determines the bus assignment priority of the COB.
Code	Data transfer format
Coding element	Two-part element for the unambiguous allocation of electronic and basic module.
Command modules	Command-capable modules are modules with an internal memory that are capable of executing particular commands (such as output substitute values).
Common potential	Electrical interconnection of the reference potentials of the control and load circuit of I/O modules.
Communication Profile	Here: CANopen communication profile. Described in the CiA Draft Standard CiA-DS-301.
Configuring	Systematic arrangement of the I/O modules of a station.

Const	Constant object. The value is read-only and does not change during runtime. Example: Device Software Version.
CPU	Abbreviation for “Central Processing Unit”. Central unit for data processing, which represents the core element of a computer.
CRC	Cyclic Redundancy Check: CAN data integrity check routine with low residual error probability. Also used in other areas of data transfer.
CSA certification	Canadian certification (Canadian Standards Association)
CSMA	Carrier Sense Multiple Access. Bus access routine used in CAN. Each node can independently access the bus as soon as the bus is free.
Data Frame	CAN message frame used by a transmitter to broadcast data to several receivers.
Data request telegram	CAN remote transmission request frame, which a network node transmits to another node.
DBT	Distributor. One of the services of the application layer in the CAN Reference Model. Used for the configuration of layers in the CAN Reference Model. The assignment of COB-IDs to the COBs used by CMS represents a DBT task.
DBT master	Special CAN node. Its task is to assign and manage the COB-IDs in a CAL or CANopen network.
DBT slave	All CAN nodes assigned a COB-ID by the DBT master.
Device Profile	Here: CANopen Device Profile. Described in CiA Draft Standards CiA-DS-401 ff.
Digital	Represents a value that can acquire only definite states within a finite set, e.g. a voltage. Mostly defined as “0” and “1”.
DIN	Abbreviation for “Deutsches Institut für Normungen e. V.” (German Institute for Standardization).
Download	The download of configuration data, parameters or programs to a CAN node.
Dual Code	Natural binary code. Frequently used code for absolute measurement systems.

Earthing strip	Flexible conductor, mostly braided. Interconnects inactive parts of equipment, e.g. the doors of a control panel and the switch cabinet body.
EDS	Electronic Data Sheet: File containing device-specific parameter definitions (provided by the manufacturer of CANopen or DeviceNet devices)
EEPROM	Abbreviation for “Electrically Erasable Programmable Read-only.
EIA	Abbreviation for “Electronic Industries, USA”.
Electrical equipment	Comprises all equipment used for the generation, conversion, transfer, distribution and application of electrical energy, e.g. power lines, cables, machines, controllers.
EMC	Abbreviation for “Electromagnetic Compatibility”. The ability of electrical equipment to function trouble-free within a particular environment without a negative effect on the environment concerned.
EN	Abbreviation for “European Norm”.
Equipotential bonding	Adaptation of the electrical level of the body of electrical equipment and auxiliary conductive bodies by means of an electrical connection.
ESD	Abbreviation for “Electrostatic Discharge”.
Fault Mode	Determines the mode of reaction to errors. When this bit is set for an output, this output will be set to the value declared in its fault state parameter.
Field supply	Voltage supply to field devices as well as signal voltage.
Fieldbus	Data network on the sensor/actuator level. The fieldbus interconnects the devices at field level. Characteristic feature of the fieldbus is their highly reliable transfer of signals and real-time response.
Galvanic coupling	A galvanic coupling occurs when two circuits use the same cable. Typical sources of interference are, for example, starting motors, static discharges, clocked devices, and a potential difference between the housing of components and the common power supply.

GND	Abbreviation for “GROUND” (0 potential).
Ground	In electrical engineering the name for conductive grounding with an electrical potential at any point equal to zero. In the environment of grounding devices, the electrical ground potential may not equal zero. This is called a “reference ground”.
Ground (verb)	Represents the connection of an electrically conductive component to the equipotential earth via a grounding device.
Grounding device	One or several components that have a direct and good contact with the ground.
Guard Identifier	Guarding protocol identifier used for node monitoring. The NMT master here transmits an RTR to the monitored slaves, requesting it to return its current status.
Guard Time	Node monitoring time. Configurable time utilized for monitoring the CAN nodes. After this Guard Time, the NMT master transmits an RTR frame including the Guard Identifier to the corresponding NMT slave requesting it to return its current status data.
Guarding	Node monitoring performed by means of the Guarding protocol.
Hexadecimal	Number system with base 16. Counting from 0 to 9 and then with the letters A, B, C, D, E and F.
I/O	Abbreviation for “Input/Output”.
Identifier	Frame identifier. Standard CAN uses 11-bit, Extended CAN 29-bit identifiers.
Impedance	Apparent resistance that a component or circuit of several components has for an alternating current at a particular frequency.
Inactive metal parts	Conductive parts that cannot be touched and which are insulated from active metal parts. They can, however, carry voltage in the event of a fault.

Index	The index (in arrays and records) and the subindex specify an object address that conforms with CANopen standard. This address represents an index in the object dictionary. Only an index is output for simple variables. Array structures have subindexes which are appended comma-separated to the index. Example: [1800,01] = index 1800, subindex 1.
Inductive coupling	Inductive (magnetic) coupling occurs between two current carrying conductors. The magnetism produced by the currents induces an interference voltage. Typical interference sources are, for example transformers, motors, mains cables installed parallel and RF signal cables.
Inhibit Time	Time interval during which a PDO may not be transmitted again, in order to avoid excess load on the network.
Life Time	Life Time/node monitoring time. Configurable time utilized for monitoring the CAN nodes. The CAN node to be monitored expects at least one Guarding message within this Life Time.
Lightning protection	Represents all measures for preventing system damage due to overvoltage caused by lightning strike.
LMT	Layer Management. One of the services of the application layer (CAL) in the CAN Reference Model. Described in the CiA Draft Standard CiA-DS-205. It contains the so-called layer-specific management functions. These include in particular the module name and ID as well as the timing parameters of the physical transmission layer, i.e. the baud rate of the CAN nodes.
LMT master	In the LMT model, this CAN node is assigned the task of configuring the LMT parameters of the other CAN nodes.
LMT slave	CAN node that communicates in the LMT model with the LMT master in a master/slave model.
Low impedance connection	Connection with low alternating-current resistance.
LSB	Abbreviation for “Least Significant Bit”. Bit with the lowest value.

Mapping	All connection data, i.e. the assignment of network variables to PDOs. A PDO can transmit one or multiple network variables (see CiA DS-301). The assignment of variables to PDOs is defined in the Mapping tables. These can be addressed via the object dictionary.
Master	Station or node in a bus system that controls communication between the other stations of the bus system.
Master-slave mode	Operating mode in which a station or node of the system acts as master that controls communication on the bus.
Mode	Operating mode.
Module bus	Represents the internal bus of an XI/ON station. Used by the XI/ON modules for communication with the gateway. Independent of the fieldbus.
MSB	Abbreviation for “Most Significant Bit”. Bit with the most significant value.
Multimaster Mode	Operating mode in which all stations or nodes of a system have equal rights for communicating on the bus.
NAMUR	Abbreviation for “Normen-Arbeitsgemeinschaft für Mess- und Regeltechnik” (Standards Work Group for Instruments and Controls). NAMUR proximity switches represent a special category of 2-wire proximity switches. They are highly resistant to interference and reliable due to their special construction, e.g. low internal resistance, few components and short design.
NMT	Network Management. One of the services of the application layer in the CAN Reference Model. Used in a CAN network for initialization, configuration and error handling routines.
Nodes	Network slaves.
Noise emission (EMC)	Testing procedure to EN 61000-6-4
Noise immunity (EMC)	Testing procedure to EN 61000-6-2
NV memory	Non-volatile electronic memory for electronic counters and for data backup during power loss.

Object Dictionary	Object dictionary. The object dictionary contains all objects accessible via the network in a defined sequence. These objects are accessed via a 16-bit index.
Operational	Active status of a CANopen node. In this state the node can transmit and receive PDOs, depending on the type and configuration. SDO communication is still possible.
Overhead	System management time required in the system in each transmission cycle.
Parameter assignment	Definition of parameters for individual bus slaves or their modules in the configuration software of the CANopen master.
PDO	Process Data Object. Object for the data exchange between different CAN nodes.
PLC	Abbreviation for Programmable Logic Controller.
Polling mode	A slave returns data only after it has received an RTR from the bus master.
Potential-free	Galvanic isolation between the reference potentials of the control and load circuit of I/O modules.
Pre-operational	Status of a CANopen node such as EZ221-CO after power on and automatic initialization. The node can be addressed by means of SDO, and can be set to “Operational” from this state.
Priorities	The CAN frame identifiers also determine the priorities for bus access. This allows fast bus access according to the significance of messages.
Protected against short-circuit	Property of electrical equipment. Short-circuit-proof equipment has the ability to withstand the thermal and dynamic loads that may occur at the location of installation on account of a short-circuit.
Protective conductor	A conductor required for the protection against dangerous currents, designated by the letters PE (abbreviation of “Protective Earth”).

Radiated coupling	Radiated coupling occurs when an electromagnetic wave makes contact with a conductor structure. The impact of the wave induces currents and voltages. Typical interference sources are, for example ignition circuits (spark plugs, commutators of electrical motors) and transmitters (e.g. radio-operated devices), which are operated near the corresponding conductor structure.
Reference ground	Ground potential in the area of grounding devices. Unlike "ground", which always has zero potential, it may have any potential except zero.
Reference potential	Represents a reference point for measuring and/or visualizing the voltage of any connected electrical circuits.
Repeater	Amplifier for signals transferred across a bus.
Response time	In a bus system the time interval between the sending of a read job and the receipt of the response. Within an input module, it represents the time interval between the signal change at an input and its output to the bus system.
RO	Read Only. Object assigned the read only attribute.
RW	Read/Write. Object assigned read/write attributes.
RWR	Read/Write/Read. Object assigned read/write attributes. It can only be read, however when data is transferred via PDOs (as network variable).
RWW	Read/Write/Write. Object assigned read/write attributes. It can only be written, however when data is transferred via PDOs (as network variable). This corresponds, for example with a digital output that is normally write accessed, but also allows (via SDO) read back of the last entered value.
SDO	Service Data Object. Object for peer to peer communication with access to the Object Dictionary of a CAN node.
SDO Manager	CANopen manager/master that can access all devices via SDO and of which several may exist in complex or large plants (e.g. for distributed tasks).
Serial	Describes an information transfer technique. Data is transferred in a bit-stream across the cables.

Shield	Term that describes the conductive covering of cables, cubicles and cabinets.
Shielding	Refers to all measures and equipment used to connect system parts to the shield.
Slave	Station in a bus system that is subordinate to the master.
Station	Function unit or module, consisting of several elements.
Subindex	See Index.
Sync	The SYNC object is a frame a station broadcasts periodically. Can be used to transfer device data at defined time intervals. PDOs that should respond to these frames are assigned the synchronous Transmission Type attribute (see Transmission Type).
Topology	Geometric structure of a network or circuit arrangement.
Transmission Type	Transmission characteristics of a PDO.
UART	Abbreviation for “Universal Asynchronous Receiver/ Transmitter”. A “UART” is a logic circuit used for converting an asynchronous serial data sequence into a bit-parallel data sequence or vice versa.
Unidirectional	Working in one direction.
WO	Write Only. Object with write access only.

Index

Eaton Electrical
1000 Cherrington Parkway
Moon Township, PA 15108-4312
USA
Tel: 1-800-525-2000
www.EatonElectrical.com

Printed in USA
Publication No. MN05013007E
May 2005