User Manual

EASY221-CO
CANopen Slave Interface

09/02 AWB2528-1479GB

1st published 2002, edition date 09/02

Author: Ronny Happ
Editor: Michael Kämper
Translator: Harold Schierbaum

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.
- Earth and short circuit.
- Cover or enclose neighbouring 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 equalisation. 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 is targeted at automation technicians and engineers.

We presume a profound knowledge of programming fieldbus CANopen master systems CANopen and knowledge of the easy control relay.

Further manuals for this device

We principally refer to the control relay easy412, easy600 (AWB2528-1304-...) User Guide.

References

[1] CANopen – Application Layer and Communication Profile
CiA Draft Standard DS301
Version 4.01
June 1, 2000

[2] CANopen – Cabling and Connector Pin Assignment
CiA Draft Recommendation DR303-1
Version 1.0
October 10, 1999

[3] CANopen – Indicator Specification
CiA Draft Recommendation Proposal DRP303-3
Version 0.2
February, 22, 2001

[4] CANopen – Layer Setting Services and Protocol (LSS)
CiA Draft Standard Proposal DSP305
Version 1.0
May 31, 2000

Data types

CANopen specifies its own data types in [1] Chapters 9.1 and 9.5.3. The data types listed in the table below are used for the CANopen protocol handler of the EASY221-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 may not be delimited with 0_{hex}!	Allowed are all ASCII the characters 20_{hex} to $7E_{hex}$ as well as 0_{hex}	
DOMAIN	User-specific data format		

Device designation

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

- easy600 for
  - EASY6..-AC-RC(X)
  - EASY6..-DC-.C(X)
- easy-AC for
  - EASY412-AC-..
  - EASY6..-AC-RC(X)
- easy-DC for EASY620/621-DC-.C(X)

Abbreviations and symbols

This manual uses abbreviations and symbols which have the following meaning:

BCD	**B**inary **C**oded **D**ecimal code
CAL	**C**AN **A**pplication **L**ayer
CAN	**C**ontroller **A**rea **N**etwork
COB	**C**ommunication **Ob**ject
COB ID	**C**ommunication **Ob**ject **Id**entifier
COV	**C**hange **o**f **V**alue
DEC	**Dec**imal (Number system based on 10s)
EDS	**E**lectronic **D**ata **S**heets
EMCY	**Em**ergen**cy** Object
HEX	**Hex**adecimal (Number system based on 16)
ID	**Id**entifier
LSS	**L**ayer **S**etting **S**ervice
NMT	**N**etwork **M**anagemen**t**
NVM	**N**on-**V**olatile **M**emory
NVM-PA	**N**on-**V**olatile **M**emory **Pa**rameter (load and save area)
NVM-RO	**N**on-**V**olatile **M**emory-**R**ead **O**nly (read-only memory area)
PC	**P**ersonal **C**omputer
PDO	**P**rocess **D**ata **O**bject
ro	**R**ead **O**nly (read access only)
ROM	**R**ead **O**nly **M**emory
RTR	**R**emote **T**ransmit **R**equest
rw	**R**ead/**W**rite (read/write access)
SELV	**S**afety **E**xtra **L**ow **V**oltage"
SDO	**S**ervice **D**ata **O**bject
FS	**F**actory**S**setting

▶ indicates actions to be taken.

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

Caution!
Warns of the risk of major damage to assets and minor 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

For greater clarity, the name of the current chapter is shown in the header of the left-hand page and the name of the current section in the header of the right-hand page. Pages at the start of a chapter and empty pages at the end of a chapter are exceptions.

1 To EASY221-CO

EASY221-CO was developed for automation tasks using the fieldbus CANopen system. EASY221-CO represents a "gateway" and can only be operated in combination with expandable easy600 base units. The control relay easy600 with CANopen gateway EASY221-CO always operates as network slave.

System overview

The easy CANopen slaves are integrated into a CANopen fieldbus system.

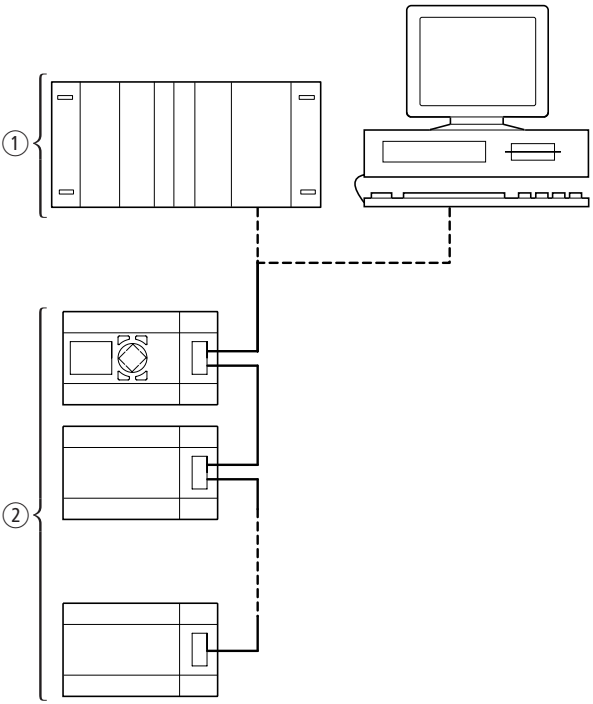


Figure 1 : Integration of EASY221-CO in the CANopen network

① Master area, PLC (e.g.: XC600) or PC with CAN card

② Slave area, e.g.: Control relay easy with CANopen interface

Structure of the unit

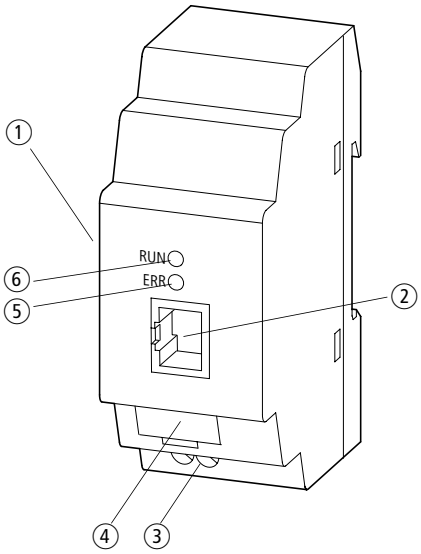


Figure 2 : Structure of EASY221-CO

① EASY-LINK socket
② CANopen connection, 8-pin RJ45 socket
③ Power supply 24 V ⎓
④ Equipment rating plate
⑤ ERR LED (Error)
⑥ RUN LED

Hardware and operating system requirements

The EASY221-CO expansion unit is operated with an easy600 basic unit as of operating system version 2.4.

Use other than intended

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

- furnaces,
- emergency-stop,
- cranes or
- two-hand safety controls.

2 Installation

Applicable are the same guidelines as for easy600 basic units with expansion modules.

Connecting the EASY221-CO to the basic unit

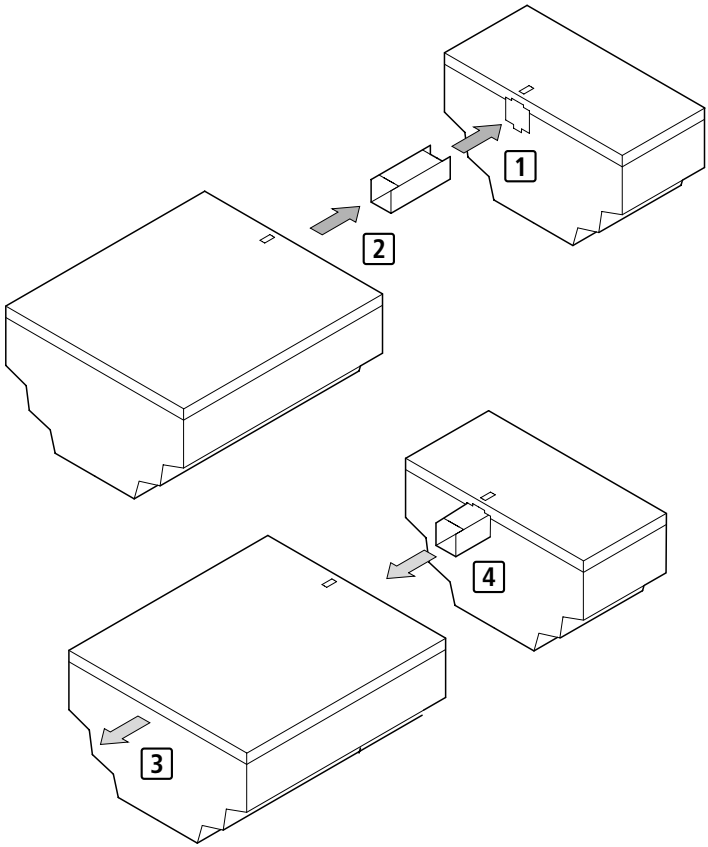


Figure 3: Mounting the EASY221-CO to the basic unit easy600

1 + 2 Installation

3 + 4 Removal

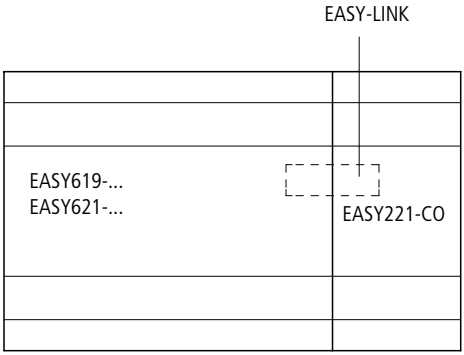


Figure 4: Connection between easy600 and EASY221-CO

Connecting the power supply

The device EASY221-CO operates with a 24 VDC supply voltage (→ section "Power supply", Page 109).

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

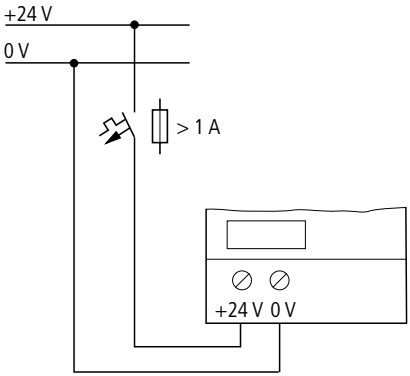


Figure 5: Power supplyEASY221-CO

connecting CANopen

ISO 11898 specifies the type of cable and coupling connectors to be used, as well as the terminating resistors.

A shielded 8-pin RJ45 plug is used to connect the EASY221-CO. The pin assignment of the plug is specified in the CiA DR-303-1 and also below.

Terminal assignmentCANopen

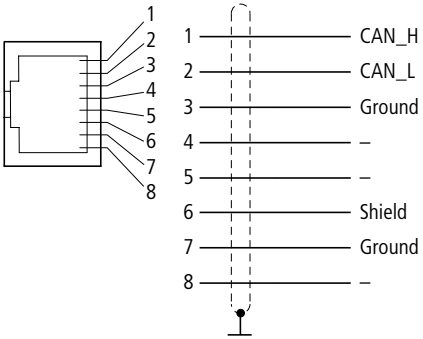


Figure 6: Pin assignment of the equipment socket

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	–	Not used

Terminating resistors

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

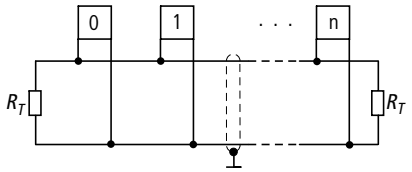


Figure 7: Terminating resistors R_T: CAN_H and CAN_L terminals

R_T = 120 Ω

EMC compatible wiring

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

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

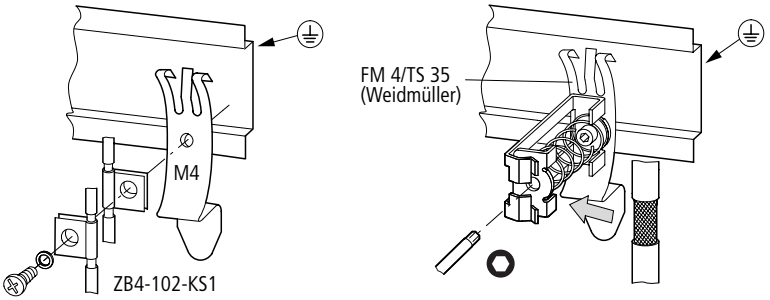


Figure 8: Shielding connection to the mounting rail

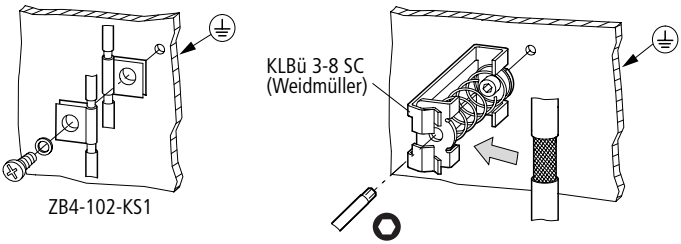


Figure 9: Shielding connection to the mounting plate

Potential isolation

The following potential isolation specifications apply to the interfaces of the EASY221-CO:

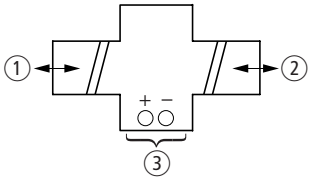


Figure 10: Potential isolation between the supply voltage and outputs

① Safe electrical isolation between EASY-LINK and 240 VAC

② Simple electrical isolation to the CANopen communication bus

③ Power supply 24 V ⎓

Data transfer rates – Auto baud recognition

After it is switched on, the EASY221-CO module automatically detects the data transfer rate of the communication network. However, this is possible only if at least one network node transmits valid message frames. The fast flashing ERR and RUN LEDs of EASY221-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 changes to flashing mode and the ERR LED will be switched off.

EASY221-CO supports the data transfer rates specified by the CiA. The table below provides an overview of recommended bit rates and 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 Operating the unit

Initial power on

Before you switch on the unit, verify that it is properly connected to the power supply, to the bus connectors and to the basic unit.

▶ Switch on the power supply for the basic unit and the EASY221-CO.

If the EASY221-CO has factory 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 the CANopen structure. All node IDs must be unique within the entire bus structure.

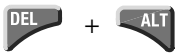
There are three ways to set the CANopen address of an EASY221-CO:

- Using the integrated display and keyboard on the "easy" basic unit
- Using EASY-SOFT V3.01 or higher on the PC
- Via the configuration software of the master PLC used (possibly by means of an explicit message).

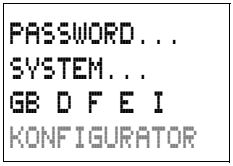
Setting the address at the basic unit with display

Precondition:

- The basic unit easy600 and EASY221-CO are connected to the power supply.
- The basic unit is not password-protected.
- The basic unit is configured with an operating system version 2.4 or higher.
- The basic unit must be in STOP mode.
- No active communication between the EASY221-CO and the CANopen master.



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

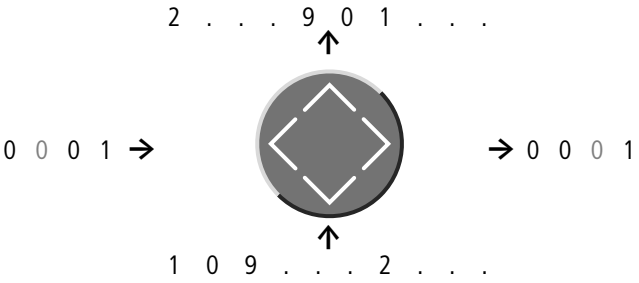


▶ Use the cursor keys ∧ or ∨ to change to the KONFIGURATOR.

▶ Confirm with OK.

The CANopen menu appears.

▶ Set the address by means of the cursor keys:
  – Set the current numeric value via ∧ or ∨.
  – You can change the actual numeric value via < or >.



▶ Accept the address with OK.

▶ Cancel address input.

Setting the address by means of EASY-SOFT

As of EASY-SOFT V3.1 or higher:
Menu → Online → configuration of expansion units

After you have modified the node ID via the basic unit, you must restart the EASY221-CO by switching power off and on.

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.

LED status displays

The EASY221-CO expansion unit is equipped with two LED indicator elements, i.e. 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	EASY221-CO operates without error. When the RUN LED is switched off also, the EASY221-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 PLC has either reached or exceeded the "Warning Limit". Too many errors have occurred on the CANopen bus.
3	Flickering	AutoBaud/LSS	Auto baud rate recognition is currently busy (flickers alternating with the RUN LED).
4	Flashes twice	Error control event	A protective "Guard Event" or a "Heartbeat Event" has occurred.
5	ON	Bus switched off	The CANopen PLC has changed to BUS-OFF state.

RUN LED

no.	RUN LED	Status	Description
1	OFF	Reset	EASY221-CO is either switched off or is currently being reset.
2	Flickering	AutoBaud	Auto baud recognition is currently busy (flickers alternating 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 state.
5	ON	OPERATIONAL[1)]	The device is in OPERATIONAL state .

1) Detailed information on the various states is found in Section "Network management", Page 28.

Timing chart of the ERR and RUN LEDs

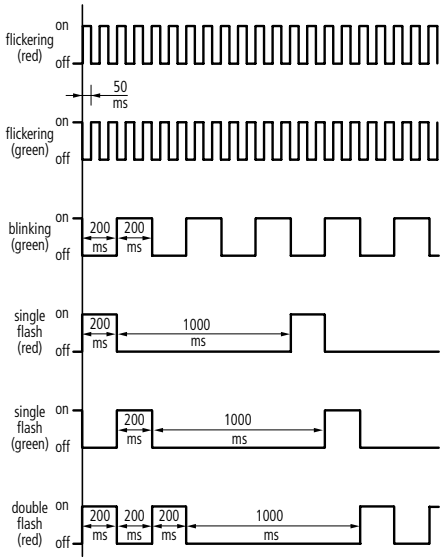


Figure 11: ERR and RUN LED

Cycle time of the "easy" basic unit

Communication between the easy600 basic unit and the EASY221-CO via EASY-LINK extends the cycle scan time of the basic unit

in the worst case by 25 ms.

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

EDS file

You can implement the EASY221-CO into the CANopen structure by means of a standardised EDS file (Electronic Data Sheet). The EDS defines the functions of a CANopen in machine code. It lists all objects, supported data transmission rates, the manufacturer and many other data.

You can either order the current version of the EDS file directly at Moeller or download updates from the Moeller homepage:

http://easy.moeller.net → Download → ....

Follow the link on this page.

4 CANopen Services

The functions for controlling EASY221-CO communication on the CANopen bus are defined by means of CANopen services.

Communication objects

EASY221-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/writs access to the entries of the object hierarchy.

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

EASY221-CO supports expedited transfer (of up to four data bytes) and segmented transfer (of more than four data bytes).

Block transfer is not supported.

More detailed information on the sequence is found in Section "SDO protocol", Page 102.

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

EASY221-CO does not support client SDOs.

Process data objects

Process data are exchanged in CANopen by means of PDOs (= Process Data Object). More detailed information on the sequence is found in Section "PDO protocol", Page 101.

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

PDO	Process data	Length
Receive PDO	Command or identifier for the image data R16 to R1 of easy600 (data output to "easy")	Byte 3:
Transmit PDO	Command or status for the image data S8 to S1 of easy600 (input data from "easy")	Byte 3:

→ For details on the structure of process data refer to Section "Manufacturer-specific objects", Page 44.

Receive PDO

EASY221-CO receives data via receive PDO from the CANopen network (PDO consumer) and writes these via EASY-LINK to easy600 as command or identifier for the image data R16 to R1.

Transmit PDOs

The commands or status of the S8 to S1 image data of easy600 are read vice versa via EASY-LINK and passed to the CANopen network (PDO producer) as transmit PDO of EASY221-CO .

PDO mapping

EASY221-CO supports **static PDO mapping**. The process data are here permanently assigned to the specific PDOs, with a granularity of 1 byte. The PDO mapping is permanently stored and can not be modified by the user.

The transmission types of PDOs

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

Transmit PDO:
The default setting for transmit PDOs is the "asynchronous" transmission type (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 with maximum speed.

The Inhibit Time is not enabled by default, since data transferred via EASY-LINK protocol are updated only at 180 ms intervals. The user can, however enable an Inhibit Time definition for the Transmit PDO as required.

Event Timed PDOs

The expiration of a counter can be considered as an event that triggers the transmission of a PDO. EASY221-CO does not support Event Timed PDOs by default, but can be enabled for the transmit PDOs as required.

Multiplexed PDOs

In addition to elementary process data, the multiplexed PDOs also contain an 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 unit.

EASY221-CO does not support multiplexed PDOs.

PDO mapping

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

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

Data byte	Content	Description
Data byte 1	Cyclic instruction and identifier	Data output to EASY 619 or EASY621 (index 2011, subindex 00_{hex})
Data byte 2	Image data RD16 to RD9	
Data byte 3	Image data RD8 to RD1	
Data byte 4 to 8	Not transferred	

→ For details on the structure of process data refer to Section "Output data (2011hex)", Page 45.

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

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

Data byte	Content	Description
Data byte 1	Cyclic instruction and status	Data output to EASY 619 or EASY621 (index 2012_{hex}, subindex 00_{hex})
Data byte 2	Image data SD8 to SD1	
Data byte 3	empty (00_{hex})	
Data byte 4 to 8	Not transferred	

→ For details on the structure of process data refer to Section "Input data (2012hex)", Page 48.

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

System services

Synchronisation objects

EASY221-CO as consumer supports the synchronisation object (index: 1005_{hex}) in order to enable the synchronous transfer of PDOs.

Time Stamp object

A time consumer uses the time stamp object (Index: 1012_{hex}) to provide a common time reference to all system nodes.
EASY221-CO does not support time stamp objects.

Emergency object

EASY221-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 99.

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 therefore also change their status.

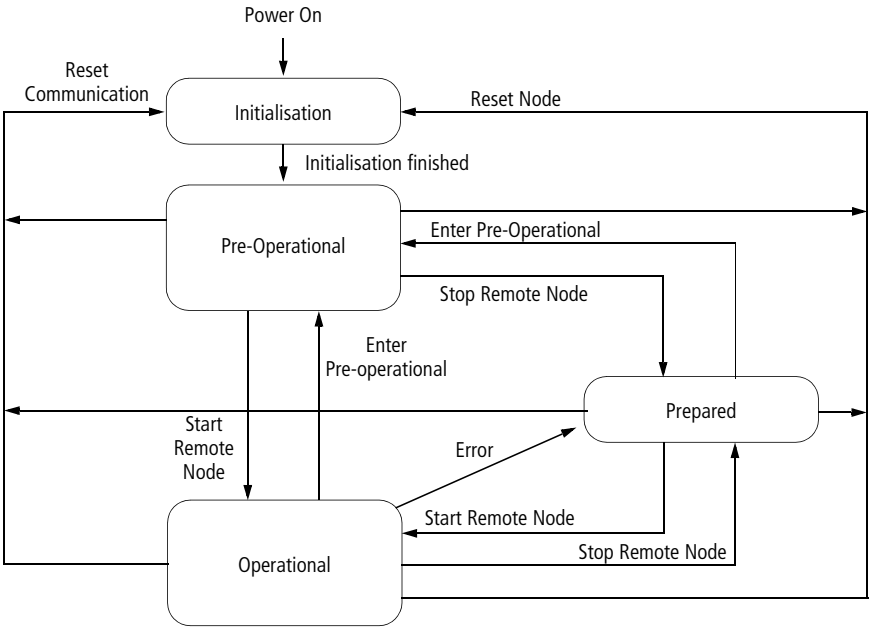


Figure 12: Network management

We distinguish between the following states:

- Initialisation,
- Pre-operational,
- Operational and
- Prepared.

Initialisation,

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

Pre-operational,

In this state, it is possible to communicate with the node via SDOs (e.g. setting the Guarding Time, Lifetime Factor). The node can neither communicate via PDO nor transmit emergency frames.

Operational

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

Prepared.

In this state, the node connection is switched to bus-off state; neither SDO nor PDO communication is possible. The network status of a node can be changed only by means of a corresponding 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 uses the identifier of the NMT error control protocol and is assigned fixed to the set device address (1792_{dec} + device address).

For information on the PDO and SDO transfer please refer to Chapter 6, Page 101.

Process data exchange by means of PDOs is enabled by setting the module to OPERATIONAL state via the "Start Remote Node" service. TxPDOs assigned the 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 no longer possible and the module only responds to the NMT services:

- Start Remote Node, transition to OPERATIONAL state; it is 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 INITIALISATION 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:

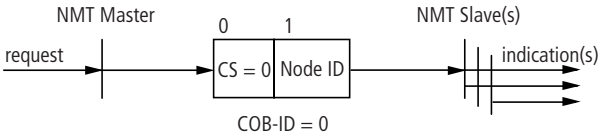


Figure 13: Structure of the NMT services, Start Remote Node

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

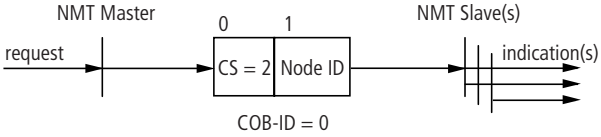


Figure 14: Structure of the NMT services, Stop Remote Node

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

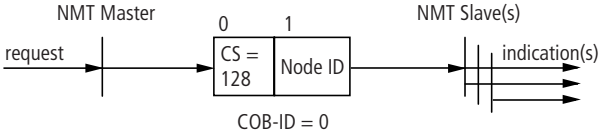


Figure 15: Structure of NMT services, PRE-OPERATIONAL state

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

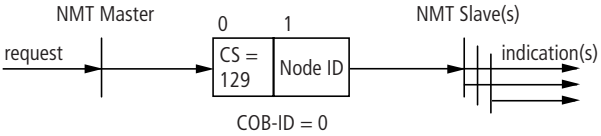


Figure 16: Structure of NMT services, Reset Node

Node ID = 0: Resets all existing nodes.

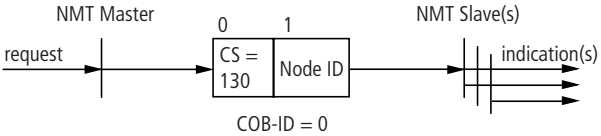


Figure 17: Structure of NMT services, Reset Communication

Node ID = 0: Resets all existing nodes.

Node monitoring

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

EASY221-CO supports Node Guarding and Heartbeat producer modes for node monitoring.

Node Guarding

The NMT master polls all NMT slaves at certain 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 his application if a node fails to respond to the RTR within the specific "Node Lifetime".

Failure of Node Guarding

Error events triggered after the Life Time has expired and a Node Guard has not received a frame from EASY221-CO will be treated as communication error.

The R data for the "easy" 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; the outputs of the "easy" basic unit can now receive PDO data.

Heartbeat Producer

EASY221-CO signals its communication status by broadcasting a cyclic heartbeat frame. 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 EASY221-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 at the EASY221-CO node, the first heartbeat frame will be transmitted during the transition from "Initialisation" to "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.

EASY221-CO does not support the Heartbeat Consumer mode for receiving the heartbeat frames of other CANopen devices.

Further services

Saving and restoring entries

EASY221-CO support 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 under NVM-RO.

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

Factory settings (FS) in the area 1000_{hex} to $1FFF_{hex}$ can be restored via 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. EASY221-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 at the EASY221-CO. To ensure that the correct node ID is displayed at the "easy" basic unit as well (under menu item "Configurator"), you must switch on the coupling module again.

Device profile

This is an expansion to the communication profile to CiA-DS-301 that describes the communication mechanisms between nodes. CANopen uses it to define so-called device profiles for the essential device classes. The device profiles describe the device functions. EASY221-CO can not be assigned to an existing device profile.

5 Object dictionary

The object dictionary of EASY221-CO contains the entries described below.

Communication parameters

A detailed description of the communication parameters is found in the CiA specification [1] Section 9.6.3.

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 EASY221-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	Predefined Error Field	UNSIGNED8	rw RAM	00	Error history
	01 to 10[1)]	Default Error Field	UNSIGNED32	ro RAM		Error description (→[1] Page 9-65)
1005	00	COB-ID SYNC Message	UNSIGNED32	rw NVM-PA	00000080	COB-ID of the SYNC object, consumer device for the SYNC message

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1008	00	Manufacturer Device Name	VISIBLE_STRING[2)]	ro NVM-RO	454153 593232 312D43 4F	Device name of the module (EASY221-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}	Multiplicand for the Guard Time, the product 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	→[1] Page 9-70

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 (→ [1] Page 9-72)
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 information on the device
	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) EASY221-CO up to 16 entries in the error log.

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

EASY221-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 Receive SDO. 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 Transmit SDO. The ID is derived from the Predefined Connection Set.

EASY221-CO supports the first transmit SDO of the Predefined Connection Set. The transmit PDOs 2 to 4 are not supported. The table below shows the object dictionary entries 1400_{hex}: Communication parameters of the first receive PDO.

Index hex	Sub-index 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, correspondingly [1]
	02	Transmis-sion Type	UNSIGNED8	rw NVM-PA	FF	PDO transmis-sion mode: asyn-chronous

With the first receive PDO, the output data are stored to the object dictionary (index 2011_{hex}, subindex 00_{hex}) and are transferred to easy600 via EASY-LINK, by means of a standard protocol. The table below shows the object dictionary entries 1600_{hex}: Mapping parameters of the first receive PDO.

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

EASY221-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	Sub-index 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, correspondingly [1]
	02	Transmis-sion Type	UNSIGNED8	rw NVM-PA	FF	PDO transmis-sion mode: asyn-chronous
	03	Inhibit Time	UNSIGNED16	rw NVM-PA	0000	Inhibit time (min. time interval between the next transmis-sion of a PDO) in ms 0000_{hex} = send now
	05	Event Timer	UNSIGNED16	rw NVM-PA	0000	Event counter 0000_{hex} = not used

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

Index hex	Sub-index 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	1. Mapped Object	UNSIGNED32	ro ROM	20120018	index 2012_{hex}, subindex 00_{hex}, length of 24 bits

Manufacturer-specific objects

In addition to 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 EASY221-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	Map bar	FS hex	Meaning
2001	00	Coupling error	UNSIGNED8	ro EASY	No	–	Error status of EASY221-CO
2002	00	"easy" error	UNSIGNED8	ro EASY	No	–	Error status of easy600
2011	00	Output data	UNSIGNED24	rw EASY	Yes	140000	Output data easy600
2012	00	Input data	UNSIGNED24	ro EASY	Yes	–	Input data easy600
2020	00	Status	UNSIGNED8	ro EASY	No	FF	Status 00_{hex} = valid data, 01_{hex} = invalid data, FF_{hex} = Initialisation
2021	00	Instruction	DOMAIN Length = 7	rw EASY	No	–	Instruction to easy600
2022	00	Answer	DOMAIN Length = 7	ro EASY	No	–	Answer from easy600

Error states

Coupling error (2001_{hex}) and "easy" error (2002_{hex})

The objects 2001_{hex} and 2002_{hex} contain the error states of EASY221-CO and of the connected easy600. 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)", Page 99).

Direct data exchange with easy600

Data are exchanged directly with the connected easy600 basic unit via the object dictionary entries 2011_{hex} ("output data") and 2012_{hex} ("input data").

→ For details on the required PDO protocol refer to Chapter "CANopen Protocols", Page 101.

Output data (2011_{hex})

The inputs 2011_{hex} and 2012_{hex} can be mapped and transmitted via PDOs. The object 2011_{hex} contains the output data (R data), which are transferred to easy600 via the EASY221-CO gateway. The table below describes the structure of these output data in detail.

Byte	Meaning
0	Mode
1	Status of the easy inputs R9 to R16
2	Status of the easy inputs R1 to R8

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

Byte 0: Operating mode

easy operating mode	Bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Index for setting the basic unit to 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 $14_{hex} = 00010100_{bin}$:
Byte 0 must always contain this value if data are to be transferred to the easy600 basic unit via the EASY221-CO gateway.

Value $34_{hex} = 00110100_{bin}$:
This value sets the "easy" status from STOP to RUN. It is only interpreted as command and therefore does not permit an additional transfer of data. The index value 14_{hex} must be used in this situation.

Value $44_{hex} = 01000100_{bin}$:
This value sets the "easy" status from RUN to STOP. It is used only as instruction and is therefore based on the same operating principle as the RUN command.

Value $00_{hex} = 00000000_{bin}$:
If this value is set at the control byte, the gateway overwrites the R data with zero. This function is of interest only if a master is to be set to STOP mode and as resultant measure transfers zero values to all I/O in order to ensure safety state.

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

Byte 1: Status of the easy600 inputs R9 to R16

easy600 output	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 $19_{hex} = 00011001_{bin}$:
Enable R13, R12 and R9.

Byte 2: Status of the easy600 inputs R1 to R8

easy600 output	Bit							
	7	6	5	4	3	2	1	0
R1								0/1
R2							0/1	
R3						0/1		
R4					0/1			
R5				0/1				

easy600 output	Bit							
	7	6	5	4	3	2	1	0
R6			0/1					
R7		0/1						
R8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $2B_{hex} = 0010\ 1011_{bin}$:
Enable 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})

The entries 2011_{hex} and 2012_{hex} can be mapped and transferred via PDOs. The object 2012_{hex} contains the input data (S data) easy600 transfers via the EASY221-CO gateway to the CANopen bus. The tables below describes the structure of the input data in detail.

Byte	Meaning
0	Mode
1	Status of the easy outputs S1 to S8
2	Not used

Data input area (easy outputs S1 to S8, operating mode)

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

Byte 0: Operating mode

"easy" identification	Bit							
	7	6	5	4	3	2	1	0 STOP/RUN
with input delay	0	0	0	1	0	0	0	0/1
without input delay	0	0	1	0	0	0	0	0/1

0 = status "0"
1 = status "1"

Example:
Value $21_{hex} = 00100001_{bin}$:
"easy" is in RUN mode and operates with input delay

Byte 1: Status of the "easy" outputs S1 to S8

easy600 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.

Extended data exchange

The object dictionary entries **Status** (2020_{hex}), **Instruction** (2021_{hex}) and **Answer** (2022_{hex}) represent the interface for extended data exchange with easy600.

Control commands can be used to initiate data exchange for special services:

- Time-of-day easy600,
- Timing relays
- Counter relays,
- Switching timers,
- Actual/setpoint values,
- Actual/reference value of analogue value comparators,
- Status of inputs, outputs, contactor relays, function relays

Note!
The I/O data retain their previously defined state while a control command is being executed. The I/O data will not be updated until data exchange for the control command has been terminated.

Caution!
You may use only the values specified for the instruction code.
Verify data to be transferred in order to avoid unnecessary errors.

A data exchange method needs to be defined in order to ensure safe data exchange between the master and slave stations.

The operating mode of the basic unit must correspond with the LED status display when setting the various parameters.

An SDO download of a string to **Instruction** initiates the transfer of set data to easy600 by means of extended protocol. After the data have been exchanged, the answer of easy600 can be fetched from **Answer** via SDO upload, while **Status** returns the data transfer status. FF_{hex} indicates that

data transfer is still busy, and a new write access will not be executed. The **Status** 00_{hex} indicates that data transfer has been completed and the answer can be fetched from **Answer**. **Status** 01_{hex} indicates either the occurrence of a data transfer error or that invalid data were entered in **Instruction**. In this case **Answer** contains indefinite data.

→ For details on the required SDO protocol refer to Chapter "CANopen Protocols", Page 101.

Overview

The first data byte of the string to be written to **Instruction** represents an instruction to easy600 and defines the significance of the remaining six data bytes. The table below lists the instruction set.

Instruction	Action	Operand	Size [byte]	Page
01_{hex} to 08_{hex}	Write	Time parameters T1 to T8	3	53
09_{hex} to 10_{hex}	Write	Counter parameters C1 to C8	3	56
11_{hex}	Write	Switching timer control bytes	4	58
12_{hex} to 21_{hex}	Write	Switching timers 1 to 4: channel A to D	5	61
22_{hex} to 29_{hex}	Write	Analogue value comparator 1 to 8	2	64
$2A_{hex}$	Write	Real-time clock	4	67
$2B_{hex}$ to 32_{hex}	Read	Timing relays 1 to 8: actual values	5	69
33_{hex} to $3A_{hex}$	Read	Timing relays 1 to 8: setpoint values	4	72
$3B_{hex}$ to 42_{hex}	Read	Counter relays 1 to 8: actual values	4	75
43_{hex} to $4A_{hex}$	Read	Counter relays 1 to 8: setpoint values	4	77
$4B_{hex}$ to $5A_{hex}$	Read	Switching timers 1 to 4: channel A to D	7	79
$5B_{hex}$	Read	Analogue value inputs	3	83
$5C_{hex}$	Read	Status of the digital inputs, P buttons and operator control keys	4	85

Instruction	Action	Operand	Size [byte]	Page
$5D_{hex}$	Read	Real-time clock	5	88
$5E_{hex}$	Read	Status of the timing relays, counter relays, switching timers and analogue value comparators	5	90
$5F_{hex}$	Read	Status of the contactor relays (marker), text display and digital outputs	5	95

Timing relay T1 to T8: Setting parameters

Instruction	Action
01_{hex}	Sets the parameters for timing relay T1
02_{hex}	Sets the parameters for timing relay T2
03_{hex}	Sets the parameters for timing relay T3
04_{hex}	Sets the parameters for timing relay T4
05_{hex}	Sets the parameters for timing relay T5
06_{hex}	Sets the parameters for timing relay T6
07_{hex}	Sets the parameters for timing relay T7
08_{hex}	Sets the parameters for timing relay T8

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	Control byte:	Low reference value	High reference value	–	–	–
Slave	Answer byte	00	00	00	00	00	00

Control byte: Setting the switching function of the timing relay

Meaning	Bit							
	7	6	5	4	3	2	1	0
On-delayed,						0	0	0
off-delayed.						0	0	1
on-delayed with random switching,						0	1	0
off-delayed with random switching,						0	1	1
pulse shaping,						1	0	0
flashing.						1	0	1
Timebase: milliseconds				0	0			
Timebase: seconds				0	1			
Timebase: minutes				1	0			
Not used			0					
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution	1							

Example:
Value $89_{hex} = 10001001_{bin}$
Timing relay operates with off-delay and seconds timebase.

Timing relay, setting the reference value (byte 2 and byte 3)

Bytes 2 and 3 determine the reference value for the timing relay. The reference value depends on the selected timebase. When the control byte is set to seconds, the low value is

based on seconds and the high value on the next higher timebase (minute). The value range for each byte in this case is 0 to 59_{dec} ($3B_{hex}$). This results in the following table:

Timebase	Low value	High value
Milliseconds	0 to 59 (10 ms)	0 to 59 s
Seconds	0 to 59 s	0 to 59 min
Minute	0 to 59 min	0 to 59 h

Example:
Low value 11_{hex}: Equivalent to 17 s, at a set timebase of seconds
high value $2D_{hex}$: Equivalent to 45 min, at a set timebase of seconds

Answer byte

The first byte received from the slave contains the acknowledgement. "easy" confirms execution of the "set time reference value X" command. Otherwise, if write access was not possible "easy" rejects this action .

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COwrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 0100001_{bin}$:
The last service was executed.

→ If a specified reference value lies out of the permissible value range, EASY221-CO will generate an error message.

Counter relays C1 to C8: Setting parameters

Instruction	Action
09_{hex}	Sets counter parameter C1
$0A_{hex}$	Sets counter parameter C2
$0B_{hex}$	Sets counter parameter C3
$0C_{hex}$	Sets counter parameter C4
$0D_{hex}$	Sets counter parameter C5
$0E_{hex}$	Sets counter parameter C6
$0F_{hex}$	Sets counter parameter C7
10_{hex}	Sets counter parameter C8

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	Control byte:	Low reference value	High reference value	–	–	–
Slave	Answer byte	00	00	00	00	00	00

Control byte:

Meaning	Bit							
	7	6	5	4	3	2	1	0
Not used			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution	1							

Example:
Value $80_{hex} = 1000000_{bin}$:
The reference value will be written to the selected timing relay and appears in the parameter menu.

Setting the reference value (byte 2 and byte 3)
These two bytes determine the reference value of the counter relay. The reference value can be set within the range from 0 to 9999_{dec}. To do so you must convert the required decimal into the equivalent hexadecimal value and then split it up into the low-byte and high-byte.

Example:
Reference value = $4318_{dec} = 10DE_{hex}$:
Low-value: DE_{hex}
High-value: 10_{hex}

Answer byte
The first byte received from the slave contains the acknowledgement. "easy" confirms execution of the "set time reference value X" command.

Otherwise, if write access was not possible easy rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COwrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 01000001_{bin}$:
The last service was executed.

→ If a specified reference value lies out of the permissible value range, EASY221-CO will generate an error message.

Switching timer: Setting the control bytes

Instruction	Action
11_{hex}	Sets the switching time

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	Control byte of switching timer 1	Control byte of switching timer 2	Control byte of switching timer 3	Control byte of switching timer 4	–	–
Slave	Answer byte	00	00	00	00	00	00

Control byte of switching timers 1 to 4

Each one of the four switching timers of the easy basic unit can be enabled by means of a specifically assigned control byte. The structure of the control byte is shown below:

Meaning	Bit							
	7	6	5	4	3	2	1	0
The status of the control byte will be changed					0	0	0	1
The status of the control byte will **not** be changed					0	0	0	0
Status: do **not** include timer relay X in the circuit diagram	0	0	0	0				
Status: include timer relay X in the circuit diagram	1	0	0	0				

It is not necessarily required to set all four control bytes. The last bit of each control bytes determines whether the switching timer is used or not.

Examples

Byte	Value		Result
	hex	bin	
0	11	0001 0001	Instruction: Set switching time
1	00	0000 0000	The status of switching timer 1 will not be not changed.
2	01	0000 0001	The status of switching timer 2 is changed to "not enabled".
3	80	10000000	The status of switching timer 3 is not changed.
4	81	10000001	Switching timer 4 is enabled in the circuit diagram.

→ All changes do not apply in the circuit until the next call of the corresponding switching timer. For example, if a switching timer is active daily between 12.00 and 13.00 p.m., status changes made within this time will not apply until the timer has become inactive again, i.e either before 12.00 o'clock midday or after 13.00 p.m.

Answer byte

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "set switching timer" command.

Otherwise, if write access was not possible easy rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COwrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 01000001_{bin}$:
The last service was executed.

At least one switching timer must be addressed by the call of this instruction. Otherwise EASY221-CO generates an error message. Furthermore, the addressed switching timer must exist in the circuit diagram of the basic unit.

Switching timers 1 to 4: Setting channels A to D

Instruction	Action
12_{hex}	Sets channel A of switching timer 1
13_{hex}	Sets channel B of switching timer 1
14_{hex}	Sets channel C of switching timer 1
15_{hex}	Sets channel D of switching timer 1
16_{hex}	Sets channel A of switching timer 2
17_{hex}	Sets channel B of switching timer 2
18_{hex}	Sets channel C of switching timer 2
19_{hex}	Sets channel D of switching timer 2
$1A_{hex}$	Sets channel A of switching timer 3
$1B_{hex}$	Sets channel B of switching timer 3
$1C_{hex}$	Sets channel C of switching timer 3
$1D_{hex}$	Sets channel D of switching timer 3
$1E_{hex}$	Sets channel A of switching timer 4
$1F_{hex}$	Sets channel B of switching timer 4
20_{hex}	Sets channel C of switching timer 4
21_{hex}	Sets channel D of switching timer 4

sends	Data						
	Byte 0:	**Byte 1:**	**Byte 2:**	**Byte 3:**	**Byte 4:**	**Byte 5:**	**Byte 6:**
Master	Instruc-tion	**Control byte:** Start of day End of day	Minute ON	Hour ON	Minute OFF	Hour OFF	–
Slave	Answer byte	00	00	00	00	00	00

Control byte (Weekday: starting/ending, parameter menu display)

Each channel of a weekly timer is assigned a control byte that defines the start/stop conditions. The table below shows the precise structure of this control byte.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Day ON								
no day set						0	0	0
Monday						0	0	1
Tuesday						0	1	0
Wednesday						0	1	1
Thursday						1	0	0
Friday						1	0	1
Saturday						1	1	0
Sunday						1	1	1
Day OFF								
no day set			0	0	0			
Monday			0	0	1			
Tuesday			0	1	0			
Wednesday			0	1	1			
Thursday			1	0	0			
Friday			1	0	1			
Saturday			1	1	0			
Sunday			1	1	1			
Appears in the parameter menu								
No	1	0						
Yes	0	0						

Example:
Value $31_{hex} = 00110001_{bin}$:
The previously selected channel X of weekly timer Y is active Monday through Saturday.

Setting the ON and OFF time (byte 2 to byte 5)
The table below shows the bytes which determine the precise ON and OFF times of a channel. The resolution is in seconds.

ON time		**OFF time**	
Byte 2:	**Byte 3:**	**Byte 4:**	**Byte 5:**
Minute ON	Hour ON	Minute OFF	Hour OFF
00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})	00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})

→ You must convert all decimals into hexadecimal values and enter them accordingly.

Example:

Description	**Instruction/byte**	**Value**
Data of channel A of switching timer 4:	Instruction/byte0	$1E_{hex}$
Day: Monday through Saturday The channel appears in the parameter menu	Byte 1:	31_{hex} (see above)
ON 19:00	Byte 2:	00_{hex}
	Byte 3:	13_{hex}
OFF: 06:30	Byte 4:	$1E_{hex}$
	Byte 5:	06_{hex}

Answer byte

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "set switching timer" command.

Otherwise, if write access was not possible easy rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COwrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 0100001_{bin}$:
The last service was executed.

Setting the analogue value comparators 1 to 8

The required analogue value comparator 1 to 8 can be selected via the instructions from the table below.

Instruction	Action
22_{hex}	Sets analogue value comparator 1
23_{hex}	Sets analogue value comparator 2
24_{hex}	Sets analogue value comparator 3
25_{hex}	Sets analogue value comparator 4
26_{hex}	Sets analogue value comparator 5
27_{hex}	Sets analogue value comparator 6
28_{hex}	Sets analogue value comparator 7
29_{hex}	Sets analogue value comparator 8

For the analogue value comparator, you must furthermore define two additional bytes and write them to the "easy" basic unit.

sends	Data						
	Byte 0:	**Byte 1:**	**Byte 2:**	**Byte 3:**	**Byte 4:**	**Byte 5:**	**Byte 6:**
Master	Instruc-tion	Control byte:	(reference value) Constant	–	–	–	–
Slave	Answer byte	00	00	00	00	00	00

Control byte:

Meaning	Bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Compare to								
≧								0
≧								1
I7 to I8						0	0	
I7 to constant (byte 2)						0	1	
I8 to constant (byte 2)						1	0	
fixed			0	0	0			
Appears in the parameter menu								
No		1						
Yes		0						
Execution	1							

Example:
$82_{hex} = 10000010_{bin}$ means that the selected analogue value comparator will be enabled in the circuit diagram of the basic unit as soon as the analogue value at input I7 ≧ the defined constant (→ byte 2).

Reference value (byte 2)
The second byte contains the reference value as constant. Its value lies between 0 and 99 and is equivalent to a reference voltage of 0.0 to 9.9 V. This value you must also specify in hexadecimal format.

Example:
The reference value = 20_{hex} is equivalent to an analogue voltage of 3.2 V.

Answer byte
The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "set analogue value comparator" command.

Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COwrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 01000001_{bin}$:
The last service was executed.

Setting the real-time clock

The real-time clock of control relay easy600 can be set by means of the following instruction and parameters described.

Instruction	Action
$2A_{hex}$	Sets the real-time clock

The real-time clock of easy600 requires the following parameters. The seconds value can not be configured and is therefore automatically set to 00 when you set the clock.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	Weekday	Hour	Minute	Summer/ winter time	–	–
Slave	Answer byte	00	00	00	00	00	00

Weekday (byte 1)
Range of values: 00 to 06.
This byte indicates the weekday by means of a numerical value between 0 (Monday) and 6 (Sunday).

Hour (byte 2)
Range of values: 00 to 23_{dec} = 00 to 17_{hex}.
This byte is used to acquire the currently set hour value of the real-time clock in the basic unit. Here you also need to convert the range of values into hexadecimal format.

Minute (byte 3)
Range of values: 00 to 59_{dec} = 00 to $3B_{hex}$.
This byte is used to acquire the minute value which will be set at the real-time clock of "easy". Here you also need to convert the range of values into hexadecimal format.

Summer/winter time (byte 4)
Range of values: 00 to 01
00 = winter time or
01 = summer time.

Example:
It is Friday, the current time-of-day is set to CET summer time, 14:36 p.m. .

Byte	0	1	2	3	4	5	6
Value (hex)	A 2	04	0E	24	01	–	–
Meaning	Instruc-tion	Friday	14	36	Summer time	–	–

Answer byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COWrite request rejected due to error				0	0	0	0	0
Write request OK				0	0	0	0	1

Example:
Value $41_{hex} = 01000001_{bin}$:
The last service was executed.

Timing relays 1 to 8: Reading actual values

The following instructions can be used to fetch the actual values of the various timing relays.

Instruction	Action
$2B_{hex}$	Reads the actual value of timing relay 1
$2C_{hex}$	Reads the actual value of timing relay 2
$2D_{hex}$	Reads the actual value of timing relay 3
$2E_{hex}$	Reads the actual value of timing relay 4
$2F_{hex}$	Reads the actual value of timing relay 5
30_{hex}	Reads the actual value of timing relay 6
31_{hex}	Reads the actual value of timing relay 7
32_{hex}	Reads the actual value of timing relay 8

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	–	–	–	–	–	–
Slave	Answer byte	Control byte:	Actual value Low value	Actual value High value	Random value	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if read access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Reading the switching function (control byte: byte 1)

	7	6	5	4	3	2	1	0
On-delayed,						0	0	0
off-delayed.						0	0	1
on-delayed with random switching,						0	1	0
off-delayed with random switching,						0	1	1
pulse shaping,						1	0	0
flashing.						1	0	1
Timebase: milliseconds				0	0			
Timebase: seconds				0	1			
Timebase: minutes				1	0			
Not used			0					
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution	1							

Example:
Value $89_{hex} = 10001001_{bin}$:
Timing relay operates with off-delay and seconds timebase.

Actual value (byte 2 and byte 3)
These two bytes determine the actual value of the timing relay. The actual value depends on the set timebase. When the control byte is set to a seconds timebase, the low-value represents the SECONDS and the high-value the MINUTES. The maximum range of return values for each byte is 0 to 59_{dec} ($3B_{hex}$). The table below results:

Timebase	Low value	High value
millisecond	0 to 59 (10 ms)	0 to 59 s
Seconds	0 to 59 s	0 to 59 min
Minute	0 to 59 min	0 to 59 h

Example:
Low-value 11_{hex}: Equivalent to 17 s, at a set timebase of seconds.
High-value $2D_{hex}$: Equivalent to 45 min, at a set timebase of seconds

Random value (byte 4)
"easy" sets a random delay time between zero and the set reference time for relays operating with random switching characteristics. This reference time is specified at this byte in hexadecimal format.

Timing relays 1 to 8: Reading the reference value values

The following instructions can be used to fetch the reference values of the various timing relays.

Instruction	Action
33_{hex}	Reads the reference value of timing relay 1
34_{hex}	Reads the reference value of timing relay 2
35_{hex}	Reads the reference value of timing relay 3
36_{hex}	Reads the reference value of timing relay 4
37_{hex}	Reads the reference value of timing relay 5
38_{hex}	Reads the reference value of timing relay 6
39_{hex}	Reads the reference value of timing relay 7
$3A_{hex}$	Reads the reference value of timing relay 8

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Control byte:	Reference value Low value	Reference value High value	00	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Control byte (byte 1)

Meaning	Bit							
	7	6	5	4	3	2	1	0
On-delayed,						0	0	0
off-delayed.						0	0	1
on-delayed with random switching,						0	1	0
off-delayed with random switching,						0	1	1
pulse shaping,						1	0	0
flashing.						1	0	1
Timebase: milliseconds				0	0			
Timebase: seconds				0	1			
Timebase: minutes				1	0			
Not used			0					

Meaning	Bit							
	7	6	5	4	3	2	1	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution	1							

Example:
Value $89_{hex} = 10001001_{bin}$:
Timing relay operates with off-delay and seconds timebase.

Reference value (byte 2 and byte 3)
These two bytes determine the reference value of the timing relay. This reference value depends on the selected timebase. When the control byte is set to a seconds timebase, the low value represents the SECONDS and the high value the MINUTES. The maximum range of return values for each byte is 0 to 59_{dec} ($3B_{hex}$). This results in the following table:

Timebase	Low value	High value
Milliseconds	0 to 59 (10 ms)	0 to 59 s
Seconds	0 to 59 s	0 to 59 min
Minute	0 to 59 min	0 to 59 h

Example:
The low $value_{hex}$ is equivalent to 17 s at a seconds timebase

The high value $2D_{hex}$ is equivalent to 45 min at a seconds timebase

Counter relays 1 to 8: Reading the actual values

The following instructions can be used to fetch the actual values of the various counting relays.

Instruction	Action
$3B_{hex}$	Reads the actual value of counter relay 1
$3C_{hex}$	Reads the actual value of counter relay 2
$3D_{hex}$	Reads the actual value of counter relay 3
$3E_{hex}$	Reads the actual value of counter relay 4
$3F_{hex}$	Reads the actual value of counter relay 5
40_{hex}	Reads the actual value of counter relay 6
41_{hex}	Reads the actual value of counter relay 7
42_{hex}	Reads the actual value of counter relay 8

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	–	–	–	–	–	–
Slave	Answer byte	Control byte:	Actual value Low value	Actual value High value	00	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $40_{hex} = 01000000_{bin}$:
Read request error, no data follow.

Control byte (byte 1)

Meaning	Bit							
	7	6	5	4	3	2	1	0
Not used			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution (will be processed in the circuit diagram)	1							

Example:
Value $80_{hex} = 10000000_{bin}$:
The actual value of the counter relay is set and appears in the parameter menu.

Actual value (byte 2 and byte 3)

These two bytes determine the actual value of the counter relay. The actual value can lie within the value range 0 to 9999_{dec}. In order to determine the corresponding actual value, you need to convert the 16-bit hexadecimal low and high value into the decimal format.

Example:
High value: 10_{hex}
Low value: DE_{hex}
$10DE_{hex} = 4318_{dec}$

Counter relays 1 to 8: Reading the reference values

The following instructions can be used to fetch the reference values of the various counting relays.

Instruction	Action
43_{hex}	Reads the reference value of counter relay 1
44_{hex}	Reads the reference value of counter relay 2
45_{hex}	Reads the reference value of counter relay 3
46_{hex}	Reads the reference value of counter relay 4
47_{hex}	Reads the reference value of counter relay 5
48_{hex}	Reads the reference value of counter relay 6
49_{hex}	Reads the reference value of counter relay 7
$4A_{hex}$	Reads the reference value of counter relay 8

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Control byte	Reference value Low value	Reference value High value	00	00	00

Answer byte (byte 0)
The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
fixed		1	0					
EASY221-COReadrequest rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $40_{hex} = 01000000_{bin}$:
Read request error, no data follow.

Control byte (byte 1)

Meaning	Bit							
	7	6	5	4	3	2	1	0
Not used			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execution (is being processed in the circuit diagram)	1							

Example:
Value $80_{hex} = 10000000_{bin}$:
The reference value of the counter relay is set and appears in the parameter menu.

Reference value (byte 2 and byte 3)
These two bytes determine the reference value of the counter relay. The reference value can lie within the value range 0 to 9999_{dec}. In order to determine the corresponding reference value, you need to convert the 16-bit hexadecimal low and high value into the decimal format.

Example:
High value: 10_{hex}
Low value: DE_{hex}
$10DE_{hex} = 4318_{dec}$

Switching timers 1 to 4: Reading channels A to D

The following instructions can be used to fetch the configuration data of specific switching timers and their channels.

Instruction	Action
$4B_{hex}$	Reads channel A of switching timer 1
$4C_{hex}$	Reads channel B of switching timer 1
$4D_{hex}$	Reads channel C of switching timer 1
$4E_{hex}$	Reads channel D of switching timer 1
$4F_{hex}$	Reads channel A of switching timer 2
50_{hex}	Reads channel B of switching timer 2
51_{hex}	Reads channel C of switching timer 2
52_{hex}	Reads channel D of switching timer 2
53_{hex}	Reads channel A of switching timer 3
54_{hex}	Reads channel B of switching timer 3
55_{hex}	Reads channel C of switching timer 3
56_{hex}	Reads channel D of switching timer 3
57_{hex}	Reads channel A of switching timer 4
58_{hex}	Reads channel B of switching timer 4
59_{hex}	Reads channel C of switching timer 4
$5A_{hex}$	Reads channel D of switching timer 4

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Control byte: Time switches	Control byte: Channel	Minute ON	Hour ON	Minute OFF	Hour OFF

Answer byte (byte 0)
The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Control byte switching timer

Meaning	Bit							
	7	6	5	4	3	2	1	0
Not being processed	0	0	0	0	0	0	0	0
Execution (is being processed in the circuit diagram)	1	0	0	0	0	0	0	0

Example:
Value $80_{hex} = 10000000_{bin}$:
The addressed switching timer is used in the circuit diagram.

Control byte channel

(Weekday: starting/ending, parameter menu display)
Each channel of a weekly timer is assigned a control byte that defines the start/stop conditions. The table below shows the precise structure of this control byte.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Day ON								
no day set						0	0	0
Monday						0	0	1
Tuesday						0	1	0
Wednesday						0	1	1
Thursday						1	0	0
Friday						1	0	1
Saturday						1	1	0
Sunday						1	1	1
Day OFF								
no day set			0	0	0			
Monday			0	0	1			
Tuesday			0	1	0			
Wednesday			0	1	1			
Thursday			1	0	0			
Friday			1	0	1			
Saturday			1	1	0			
Sunday			1	1	1			

Meaning	Bit							
	7	6	5	4	3	2	1	0
Appears in the parameter menu								
No	1	0						
Yes	0	0						

Example:
Value $31_{hex} = 00110001_{bin}$:
The previously selected channel X of weekly timer Y is active Monday through Saturday.

Switching times (byte 3 to byte 6)
The table below shows which bytes precisely determine the ON and OFF times of a channel. The resolution is in seconds.

ON time		OFF time	
Byte 3:	**Byte 4:**	**Byte 5:**	**Byte 6:**
Minute ON	Hour ON	Minute OFF	Hour OFF
00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})	00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})

→ "easy" returns hexadecimal values. You may have to convert the corresponding values into decimal format.

Example:

Byte	Value	Description
0	42_{hex}	The read request has been executed. Data follow.
1	80_{hex}	The addressed switching timer is used in the circuit.
2	31_{hex} (see above)	Day: Monday through Saturday The channel appears in the parameter menu
3	00_{hex}	ON 19:00
4	13_{hex}	
5	$1E_{hex}$	OFF: 06:30
6	06_{hex}	

Reading analogue inputs

Instruction	Action
$5B_{hex}$	Reads analogue inputs

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Analogue input I7	Analogue input I8	00	00	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CO Read request request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Analogue inputs I7 and I8 (byte 1 and byte 2)

These two bytes contain the actual value of the analogue inputs I7 and I8. Their value lies between 00 and 99, which is equivalent to a voltage level of 0 to 9.9 V at the inputs. The corresponding values are returned in hexadecimal format.

Example:

Byte	Value	Description
0	42_{hex}	The read request has been executed. Data follow.
1	20_{hex}	Voltage level at input I7 = 3.2 V.
2	31_{hex}	Voltage level at input I8 = 4.9 V.

Reading the status of digital inputs, P buttons and operator control keys

Instruction	Action
$5C_{hex}$	Reads the status of the digital inputs, P buttons and operator control keys

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Inputs I8 to I1	Inputs I16 to I9	P buttons and operator control keys	00	00	00

Answer byte (byte 0)
The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COread request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Status at inputs I1 to I8 (byte 1)

easy600 input	Bit							
	7	6	5	4	3	2	1	0
I1								0/1
I2							0/1	
I3						0/1		
I4					0/1			
I5				0/1				
I6			0/1					
I7		0/1						
I8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $2B_{hex} = 00101011_{bin}$:
I6, I4, I2 and I1 are active.

Status at inputs I9 to I16 (byte 2)

easy600 input	Bit							
	7	6	5	4	3	2	1	0
I9								0/1
I10							0/1	
I11						0/1		
I12					0/1			
I13				0/1				
I14			0/1					
I15		0/1						
I16	0/1							

0 = status "0"
1 = status "1"

Example:
Value 19hex = $0001\,1001_{bin}$:
I13, I12 and I9 are active

Status of P buttons and operator control keys (byte 3)

Meaning	Bit							
	7	6	5	4	3	2	1	0
Status P1								0/1
Status P2							0/1	
Status P3						0/1		
Status P4					0/1			
ESC not actuated/actuated				0/1				
OK not actuated/actuated			0/1					
DEL not actuated/actuated		0/1						
ALT not actuated/actuated	0/1							

0 = status "0"
1 = status "1"

Example:
Value 01hex = $0000\,0001_{bin}$:
P1 active – or cursor > is actuated.

Reading the real-time clock

Instruction	Action
$5D_{hex}$	Reads the real-time clock

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	3	–	–	–	–	–
Slave	Answer byte	Weekday	Hour	Minute	Time-of-Year	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Weekday (byte 1)
Range of values: 00 to 06.
This byte indicates the weekday by means of a numerical value between 0 (Monday) and 6 (Sunday).

Hour (byte 2)
Range of values: 00 to 23_{dec} = 00 to 17_{hex}.
This byte returns the current hour value of the basic unit's real-time clock. This is a hexadecimal value.

Minute (byte 3)
Range of values: 00 to 59_{dec} = 00 to $3B_{hex}$.
This byte returns the current minute value of the basic unit's real-time clock. This is a hexadecimal value.

Summer/winter time (byte 4)
Range of values: 00 to 01
00 = winter time or
01 = summer time.

Example:

Value	Byte						
	0	1	2	3	4	5	6
hex	42	04	0E	24	01	–	–
dec	66	04	14	36	01	–	–

It is Friday, the current time-of-day is 14:36 p.m. summer time CET.

Status of the timing relays, counter relays, switching timers and analogue value comparators

Instruction	Action
$5E_{hex}$	Reads the status of: • Timing relays • Counter relays, • Switching timers and • analogue value comparators.

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

sends	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruc-tion	–	–	–	–	–	–
Slave	Answer byte	Image of timing relays T8 to T1	Image of counter relays C8 to C1	Image of switching timers W4 to W1	Image of analogue value compara-tors A8 to A1	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. "easy" confirms the execution of a "read easy parameter" request. Otherwise, if write access was not possible "easy" rejects this action.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-COread request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Timing relay image: Status T1 to T8 (byte 1)

Meaning	Bit							
	7	6	5	4	3	2	1	0
T1								0/1
T2							0/1	
T3						0/1		
T4					0/1			
T5				0/1				
T6			0/1					
T7		0/1						
T8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $2B_{hex} = 00101011_{bin}$:
T6, T4, T2 and T1 are active.

Timing relay image: Status C1 to C8 (byte 2)

Meaning	Bit							
	7	6	5	4	3	2	1	0
C1								0/1
C2							0/1	
C3						0/1		
C4					0/1			
C5				0/1				
C6			0/1					
C7		0/1						
C8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $19_{hex} = 0001\,1001_{bin}$:
C5, C4 and C1 are active

Switching timer image: Status W1 to W4 (byte 3)

Meaning	Bit							
	7	6	5	4	3	2	1	0
W1								0/1
W2							0/1	
W3						0/1		
W4					0/1			

0 = status "0"
1 = status "1"

Example:
Value $08_{hex} = 0000\,1000_{bin}$:
W3 is active.

Analogue value comparator image: Status A1 to A8 (byte 4)

Meaning	Bit							
	7	6	5	4	3	2	1	0
A1								0/1
A2							0/1	
A3						0/1		
A4					0/1			
A5				0/1				
A6			0/1					
A7		0/1						
A8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $84_{hex} = 10001000_{bin}$:
A3 and A8 are active.

Reading the status of contactor relays (markers), text displays and digital outputs

Instruction	Action
$5F_{hex}$	Reads the status of: • Contactor relay (marker) • Text displays and • Digital outputs

After the read request has been triggered by sending only the respective instruction, "easy" returns the corresponding data.

Transmitter	Data						
	Byte 0:	Byte 1:	Byte 2:	Byte 3:	Byte 4:	Byte 5:	Byte 6:
Master	Instruction	–	–	–	–	–	–
Slave	Answer byte	Image of contactor relays M8 to M1	Image of contactor relays M16 to M9	Image of outputs Q8 to Q1	Image of text markers D8 to D1	00	00

Answer byte (byte 0)

The first byte received from the slave contains the acknowledgement. easy confirms the execution of a "read easy parameter" request. Otherwise, easy rejects this action if read access was not possible.

Meaning	Bit							
	7	6	5	4	3	2	1	0
Answer bit	0/1							
fixed		1	0					
EASY221-CORead request rejected due to error				0	0	0	0	0
Read request OK, data follow				0	0	0	1	0

Example:
Value $42_{hex} = 01000010_{bin}$:
Read request OK, data follow.

Contactor relay image: Status M1 to M8 (byte 1)

Meaning	Bit							
	7	6	5	4	3	2	1	0
M1								0/1
M2							0/1	
M3						0/1		
M4					0/1			
M5				0/1				
M6			0/1					
M7		0/1						
M8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $2B_{hex} = 00101011_{bin}$:
M6, M4, M2 and M1 are active.

Contactor relay image: Status M9 to M16 (byte 2)

Meaning	Bit							
	7	6	5	4	3	2	1	0
M9								0/1
M10							0/1	
M11						0/1		
M12					0/1			
M13				0/1				
M14			0/1					
M15		0/1						
M16	0/1							

0 = status "0"
1 = status "1"

Example:
Value $19_{hex} = 00011001_{bin}$:
M13, M12 and M9 are active

Digital outputs image: Status Q1 to Q8 (byte 3)

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

0 = status "0"
1 = status "1"

Example:
Value $A8_{hex} = 10101000_{bin}$:
Q8, Q6 and Q4 are active.

Text marker image: Status D1 to D8 (byte 4)

	7	6	5	4	3	2	1	0
D1								0/1
D2							0/1	
D3						0/1		
D4					0/1			
D5				0/1				
D6			0/1					
D7		0/1						
D8	0/1							

0 = status "0"
1 = status "1"

Example:
Value $84_{hex} = 10000100_{bin}$:
D3 and D8 are active.

Error messages (Emergency)

EASY221-CO supports the defined generic error (1000_{hex}) described in the table below. It is triggered when bit 0 is set for Generic Error in the error register (index 1001_{hex}, subindex 00_{hex}).

In the manufacturer-specific error field (Manufacturer Specific Error Field), byte 0 outputs the error code of the EASY221-CO (index 2001_{hex}, subindex 00_{hex}), and byte 1 outputs the error code of the connected easy600 (index 2002_{hex}, subindex 00_{hex}). This value is currently set fixed to 00_{hex}.

Data byte	Content	Value	Description
1.	Generic Error Code	1000_{hex}	Generic Error (→ [1] Section 9.2.5.1)
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}	"easy" error (index 2002_{hex}, subindex 00_{hex})
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 fetched via server SDO. Format of entries in the Standard Error Fields (Subindex 01_{hex} to 10_{hex}):

Data byte	Content	Value	Description
1.	Error Code	1000_{hex}	Generic error see also [1] Section 9.2.5.1
2.			
3.	Additional Information	xx_{hex}	Coupling error (index 2001_{hex}, subindex 00_{hex})
4.		00_{hex}	EASY error (index 2002_{hex}, subindex 00_{hex})

Third data byte of the coupling module status

Value 00_{hex}
The "easy" basic unit is connected to the EASY221-CO gateway via EASY-LINK.

Value 04_{hex}
The "easy" basic unit is either switched off or is not connected to the EASY221-CO gateway via EASY-LINK.

Note!
When communication between the basic unit easy600 and the expansion unit EASY221-CO goes down, a corresponding error code will be generated in the third data byte. Furthermore, the Rx/Tx data of the gateway will be transferred with the value 00_{hex}.

6 CANopen Protocols

The chapter below briefly describes the CANopen protocols used for EASY221-CO.

PDO protocol

The EASY221-CO by default uses the Write PDO Protocol as shown in the figure below. The Read PDO Protocol (not displayed) can be called as required.

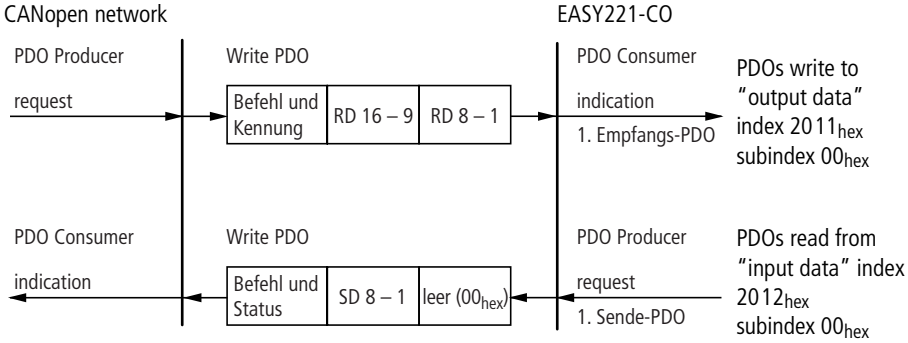


Figure 18: Write PDO Protocol

An indication informs the application that new data can be received via the first RxPDO and stored in the "output data" field of the object dictionary (index 2011_{hex}, subindex 00_{hex}). It then initiates via request the transmission of data in"input data" field of the object dictionary (index 2012_{hex}, subindex 00_{hex}) via the first Tx PDO.

SDO protocol

The SDO protocol initiates general read/write access to the object dictionaries (not explained in the following sections, see also [1]) as well as access to data in the easy600 via the extended protocol of EASY-LINK. EASY221-CO operates as server in this case, using the first server SDO. The following figure outlines the sequence.

With Initiate SDO Download, the client first initiates SDO write access to the object dictionary field "Instruction" (index 2021_{hex}, subindex 00_{hex}) of the server. Since these data can have a length of up to seven bytes, a subsequent Download SDO Segment is required in order to conclude the Segmented Transfer. The "easy" Protocol Handler then downloads the received data via the extended protocol to easy600.

The client then checks with Initiate SDO Upload whether the transfer has been completed; this is indicated by the status in the object dictionary (index 2020_{hex}, subindex 00_{hex}). Since only one byte is transferred at this stage, this is performed by means of Expedited Transfer.

The client polls this status cyclically (at intervals of approx. 50 to 100 ms), until the content is 00_{hex}. The answer of easy600 is then available in the object dictionary (index 2022_{hex}, subindex 00_{hex}).

The client then reads this answer, i.e. the read request is first initiated with Initiate SDO Upload and, since the maximum data length can amount to seven bytes, the remaining data must be read via Upload SDO Segment.

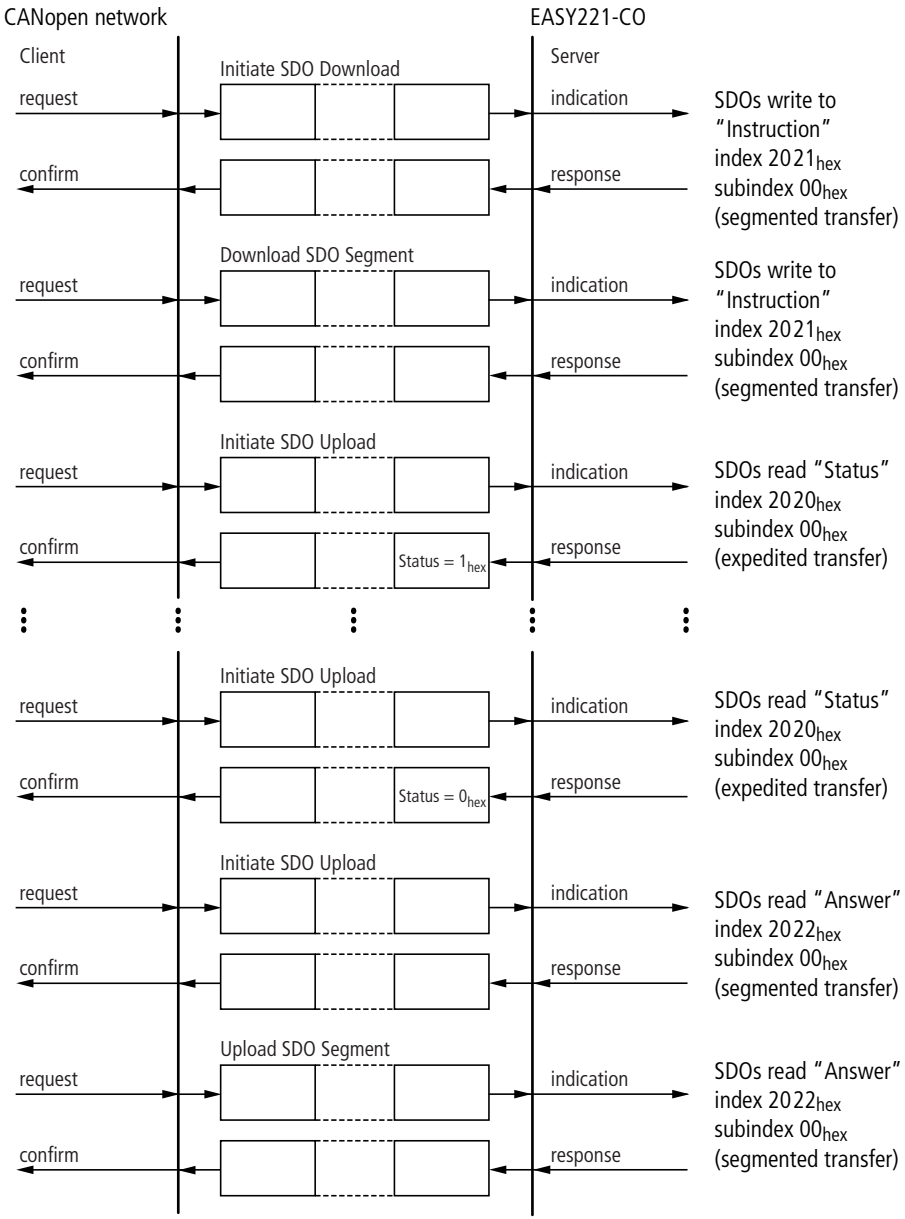


Figure 19: SDO Protocol – Sequence of the extended protocol

Emergency protocol

The Write EMCY Protocol is used for the EASY221-CO, as shown in the figure below. The emergency protocol does not require confirmation.

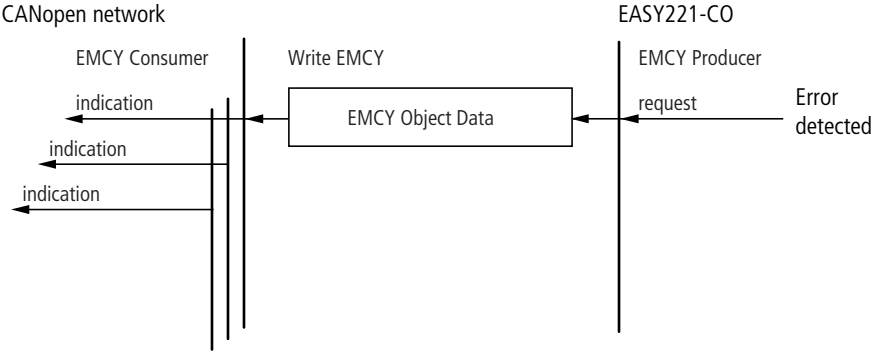


Figure 20: Emergency Object Protocol

7 What happens if...

RUN LED

Status of the RUN LED	Possible cause	To correct or avoid error
OFF	EASY221-CO is either switched off or is currently being reset.	Switch on the EASY221-CO and supply with mains voltage.
Flickering	Auto baud rate recognition is currently busy (flickers alternating with the ERR LED).	Check the communication status of the master PLC or of 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	EASY221-CO operates error-free. When the RUN LED is switched off also, the EASY221-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 recognition is currently busy (flickers alternating with the RUN LED).	Check the communication status at the master PLC or on the bus.
Flashes twice	A protective event has occurred, i.e. a "Guard" or "Heartbeat" event.	Check configuration data.
ON	The CANopen PLC has changed to BUS-OFF state.	Verify the correct setting of the NODE ID.

Annex

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 ZB4-101-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
Oscillations (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) easy-AC (IEC/EN 61000-4-5), power cable symmetrical	kV	1
High energy pulses (Surge) easy-DC (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 loss at 24 V DC, typical	W	4,8
LED displays		
Module Status LED MS	Colour	Green/red
Network Status LED NS	Colour	Green/red
CANopen		
Device connection		8-pin RJ45 socket
Electrical isolation		Bus to power supply (simple) Bus and power supply to easy basic unit (safety isolation)
Function		CANopenSlave
INTERFACE		CANopen (CAN)
Bus protocol		CANopen
Baud rate, automatic detection up to	kbps	1000
Bus termination resistors		Separate installation at the bus possible
Bus addresses, accessible via easy basic unit with display or EASY-SOFT		1 to 127
Services		
Module inputs		all data S1 to S8 (easy600)
Module outputs		all data R1 to R16 (easy600)
Module control commands		Read/Write Weekday, time-of-day, summer/winter time All parameters of the "easy" functions

Dimensions

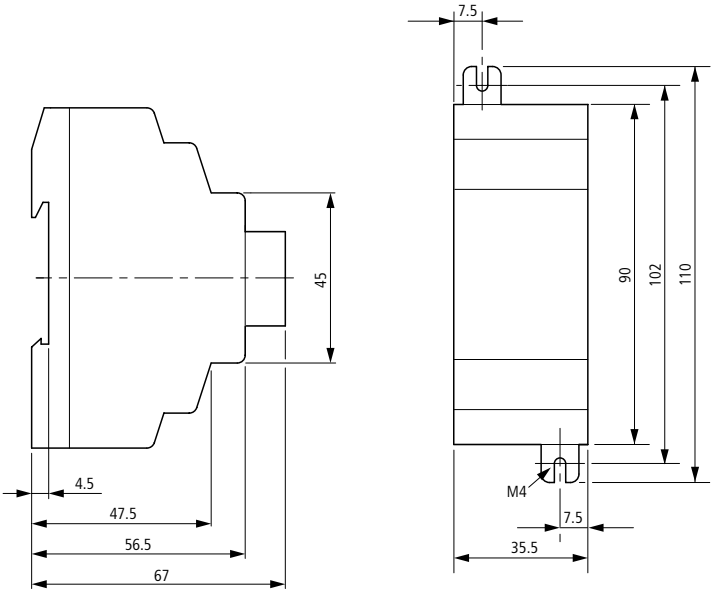


Figure 21: Dimensions EASY221-CO in mm

Glossary

This glossary refers to 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 assembly parts carrying live voltage during operation.
Address	Number that identifies a memory area, systems or module within a network, for example.
Addressing	Assignment or setting of an address for a module in the network, for example.
Analogue	Infinitely proportional value, e.g. of a voltage. Analogue 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 to the communication bus.
Automation product	I/O controlling device that is interconnected to a system process. PLCs represent a special group of automation products.
Basic CAN	Concept for the implementation of a CAN controller. All CAN messages are stored to an intermediate Tx and Rx buffer, that is, without causing high load on the host controller that has to evaluate all messages.
Baud	Measure for the data transfer rate. One baud is equivalent to the transmission of one bit per second (bps).
Baud rate	Measuring unit for the data transmission speed in bps.
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	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	The bridge connects the CANopen network to the electronic modules which represent the network slaves.
Bus	Bus line system for data exchange, for example between the CPU, memory and I/O. A bus can consist of several parallel segments for data transfer, addressing, control and power supply.
Bus cycle time	Time interval in which a master provides services to all slaves or nodes of a bus system, i.e. sets their outputs and reads inputs.
Bus line	Smallest unit connected to the bus. Consists of the PLC, a module and a bus interface for the module.
Bus system	All units as a whole which communicate across a bus.
Byte	A sequence of 8 bits
CAL	CAN Application Layer. Standardised 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	In CAN the controllers are interconnected to the bus medium by means of an interface to ISO/DIS 11898. The structure of this interface usually is not formed by a discrete circuit, but rather by a CAN Transceiver chip.
CANopen	Profile families based on CAN for high-speed data exchange. CiA standardises the communication profile as 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	with CAN: The producer automatically and immediately sends his data when the position changes.
Chassis ground	All interconnected inactive equipment parts which are not subject to hazardous fault voltage.
CiA	CiA e. V./CAN in Automation. International CAN manufacturer and user organisation.
CiA DS	CAN in Automation Draft Standard, communication profile
CiA DSP	CAN in Automation Draft Standard Proposal
CMS	CAN Based Message Specification. On 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. Unique 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-piece element for unique assignment of the electronic and basic modules.
Command modules	Represents modules with an internal memory set, capable of executing specific instructions (e.g. the output of substitute values).
common potential	Electrical connection 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.
CONFIGURE...	Systematic arrangement of the I/O modules of a station.
Const	Constant object. The value is read-only and does not change during . Example: Device Software Version.

CPU	Abbreviation for "Central Processing Unit". Central unit for data processing. 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 to transfer data from a transmitter to several receivers.
DBT	Distributor. One of the services of the Application Layer in the CAN Reference Model. Used for the configuration of the 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 whose 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.
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.".
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.

Earth	Defines in electrical engineering the conductive earth whose electrical potential is equal to zero at any point. The electrical potential in the area of earthing devices may not be equal to zero. In this case, one refers to "Reference ground".
Earth electrode	One or several components with direct and good contact to earth.
Earthing	Represents the connection of an electrically conductive component to the equipotential earth via a grounding device.
Earthing tape	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 Memory".
EIA	Abbreviation for "Electronic Industries Association", 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". Defines the ability of electrical equipment to operate error-free and without causing a negative influence within a certain environment.
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 power supply	Power supply for the field devices and signal voltage.
Fieldbus	Data network on the sensor/actuator level. The fieldbus interconnects the devices at field level. Characteristic feature of the fieldbus is the highly reliable transfer of signals and real-time response.
Galvanic coupling	Galvanic coupling generally develops between two circuits using a common cable. Typical interference sources are starting motors, static discharge, clocked devices and potential difference between the component enclosure and their common power supply.
GND	Abbreviation for "GROUND" (zero potential).
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 utilised 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	Numerical system with the base 16. The count starts at 0 to 9 a continues 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	Alternating current-resistance of a component or of a circuit consisting of several components at a specific frequency.
Inactive metallic parts	Touch-protected conductive components, isolated electrically from active metallic parts by means of an insulation, but subject to fault-voltage.

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 develops between two current-carrying conductors. The magnetic effect generated 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 utilised 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. Defined in CiA Draft Standard CiA-DS-205. Contains layer-specific management functions. These include in particular the module name and ID as well as the timing parameter 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 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".

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".
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". 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 initialisation, 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: non-volatile electronic memory for electronic counters and for value 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 can transmit and receive PDOs, depending on the type and configuration. SDO communication is still possible.
Overhead	System management time. Required once for each data transfer cycle.
Parameter assignment	Definition of parameters for individual bus slaves or their modules in the configuration software of the DeviceNet 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 EASY221-CO after power on and automatic initialisation. The node can be addressed by means of SDO, and can be set to "Operational" state in this state.
Priorities	The CAN frame identifiers also determine the priorities for bus access assignment. This allows fast bus access according to the significance of messages.
Protected against short-circuit	Property of electrical equipment. A short-circuit proof component withstands thermal and dynamic stress, which may develop at its installation location as a result of a short-circuit.
Protective conductor	Conductor required for human body protection against hazardous currents. Abbreviation: PE ("Protective Earth").

Radiation coupling	Radiation coupling develops when an electromagnetic wave meets a conductor structure, thus inducing 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	Earth potential in the area of grounding devices. May have a potential other than the zero of "earth" potential.
Reference potential	Represents a reference point for measuring and/or visualising the voltage of any connected electrical circuits.
Remote frame	A CAN remote transmission request frame a network node transmits to another node.
Repeater	Amplifier for signals transferred across a bus.
Response time	In a bus system this represents the time interval between the transmission of a read request and receiving the answer. 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 are transferred via PDOs (as network variable).
RWW	Read/Write/Write. Object assigned read/write attributes. It can only be written, however when data are 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.
Screen	Term that describes the conductive covering of cables, cubicles and cabinets.
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 are transferred in a bit-stream across the cables.
Shielding	Refers to all measures and equipment used to connect system parts to the screen.
Slave	Station or node 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).
Terminating resistor	Terminating resistor at the start and end of a bus cable. Prevents interference due to signal reflexion and is used for the adaptation of bus cables. Terminating resistors must always be the last unit at the end of a bus segment.
Topology	Geometrical network structure, or circuit arrangement.
Transmission Type	Transmission characteristics of a PDO.
UART	Abbreviation for "Universal Asynchronous Receiver/ Transmitter". A "UART" represents a logical circuit used to convert an asynchronous serial data stream into a parallel bit stream and vice versa.
unidirectional	Operating in one direction.
WO	Write Only. Object with write access only.

Index