Building Automation | Industrial Automation | Systems

Xtra Combinations

User Manual

EASY221-CO
CANopen Slave Interface

04/04 AWB2528-1479GB

MOELLER

Think future. Switch to green.

1st published 2002, edition date 09/02
2nd edition 04/2004
See revision protocol in the “About this manual” chapter

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

List of revisions

The following essential changes have been made since the previous version:

Edition date	Page	Description	New	Modification	Removed
04/04	gen.	easy700/800/MFD	✓		
	60	SDO protocol		✓	

Target group

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

Further manuals for this device

The following operating manuals should be followed:

- easy412 control relay, easy600 (AWB2528-1304-GB)
- easy700 control relay, (AWB2528-1508-GB)
- easy800 control relay, (AWB2528-1423-GB)
- MFD-Titan multi-function display (AWB2528-1480-GB)

All manuals are available on the Internet for download as PDF files. For a fast search enter the documentation number as the search criterion at http://www.moeller.net/support:

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

The CANopen specifies its own data types in [1] chapter 9.1 and 9.5.3. The data types listed in the following table 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 does not have to be delimited with 0_{hex}!	All ASCII characters from 20_{hex} to $7E_{hex}$ and 0_{hex} are permissible	
DOMAIN	User-specific data format		

Device designation

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

- easy412 for EASY412-..-... devices
- EASY512-..-..., EASY7..-..-...

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

- easy500 for
  - EASY512-AB...
  - EASY512-AC
  - EASY521-DA…
  - EASY512-DC
- easy600 for
  - EASY6..-AC-RC(X)
  - EASY6..-DC.-.C(X)
- easy700 for
  - EASY719-AB…
  - EASY719-AC…
  - EASY719-DA…
  - EASY719-DC…
  - EASY721-DC…
- easy800 for
  - EASY819-...
  - EASY820-...
  - EASY821-...
  - EASY822-...
- MFD-CP8.. for
  - MFD-CP8-ME
  - MFD-CP8-NT

- easy-AB for
  - EASY512-AB...
  - EASY719-AB...
- easy-AC for
  - EASY412-AC-..
  - EASY512-AC-..
  - EASY6..-AC-RC(X),
  - EASY719AC
  - EASY8..-AC-...
- easy-DC for
  - EASY412-DC-..
  - EASY512-DC-..
  - EASY6..-DC-…
  - EASY719-DC-..
  - EASY8..-.DC-...
- easy-DA for
  - EASY412-DA...
  - EASY512-DA...
  - EASY719-DA...

Abbreviations and symbols

Meaning of abbreviations and symbols used in this manual:

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	Decimal (number system with base 10)
EDS	**E**lectronic **D**ata **S**heets
EMCY	**Em**ergen**cy** Object
HEX	Hexadecimal (number system with base 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**etting

Writing conventions

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

▶ indicates actions to be taken.

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

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

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

→ Indicates interesting tips and additional information

1 The EASY221-CO

The EASY221-CO communication module was developed for automation tasks that use the CANopen fieldbus. The EASY221-CO is a gateway and can only be used in conjunction with the easy600, easy700, easy800 or MFD basic units. The system unit, consisting of the easy/MFD control unit and the CANopen gateway functions in the fieldbus system exclusively as a a slave station.

System overview

The easy CANopen slaves are integrated into a CANopen.

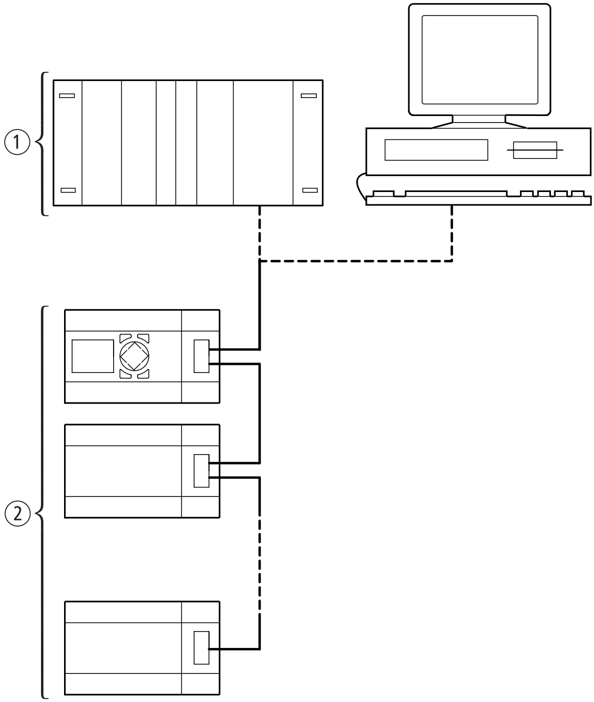


Figure 1: Integration of EASY221-CO in the CANopen network

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

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

Setup of the unit

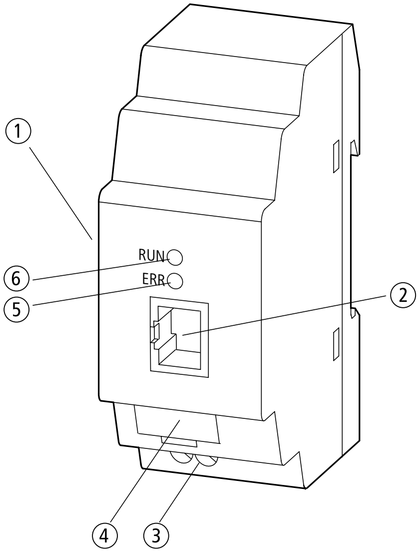


Figure 2: Setup of the EASY221-CO

① EASY-LINK socket
② CANopen terminal, 8-pin RJ45 socket
③ Power supply 24 V
④ Device designation plate
⑤ ERR LED (Error)
⑥ RUN LED

Device function description

The EASY221-CO module allows the easy and MFD series devices to be connected to a CANopen communication network. The following data can be transferred by selecting the appropriate SDO/PDO:

easy600/700/800, MFD-CP8..

- S1 to S8
  Output data of the basic unit, RUN/STOP
  (read, as viewed from CANopen master)
- R1 to R16
  Input data of the basic unit, RUN/STOP
  (write, as viewed from CANopen master)
- All function relay data
  (read, as viewed from the CANopen master)
  - Timing relays
  - Counter relays
  - Time switches
  - Analog comparators
  - Weekday, time, summer/winter time (DST)
  - All states of the easy600 contacts.

- The setpoints of the function relays
  (write, as viewed from CANopen master)
  - Timing relays
  - Counter relays
  - Time switches
  - Analog comparators
  - Weekday, time, summer/winter time (DST)

easy800/MFD-CP8..

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

Hardware and operating system requirements

The EASY221-CO expansion device operates with the easy600, easy700, easy800 and MFD basic units from the following operating system versions:

Basic unit		EASY221-CO expansion device	
Device version	OS version	Device version = 02	Device version = 03
easy600			
= 04	From 2.4	×	×
easy700			
= 01	From 1.01.xxx	–	×
easy800			
= 04	From 1.10.xxx	–	×
MFD-CP8..			
= 01	From 1.10.xxx	–	×

The device version of the appropriate basic unit or expansion device is specified on the right of the housing. Example: EASY221-CO: 03-228xxxxxxx (03 = device version)
The operating system version (OS) of the corresponding basic unit can be read using EASY-SOFT. With easy700, easy800 and MFD-CP8.. devices it is also possible to read out the information directly from the device. Read the appropriate manual for further information for this.
An overview of the modifications and new features of the different easy800 device versions is provided on Page 176.

Improper use

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

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

2 Installation

The same principles apply as for easy600, easy700, easy800 and MFD basic units with expansion devices.

Connecting the EASY221-CO to the basic unit

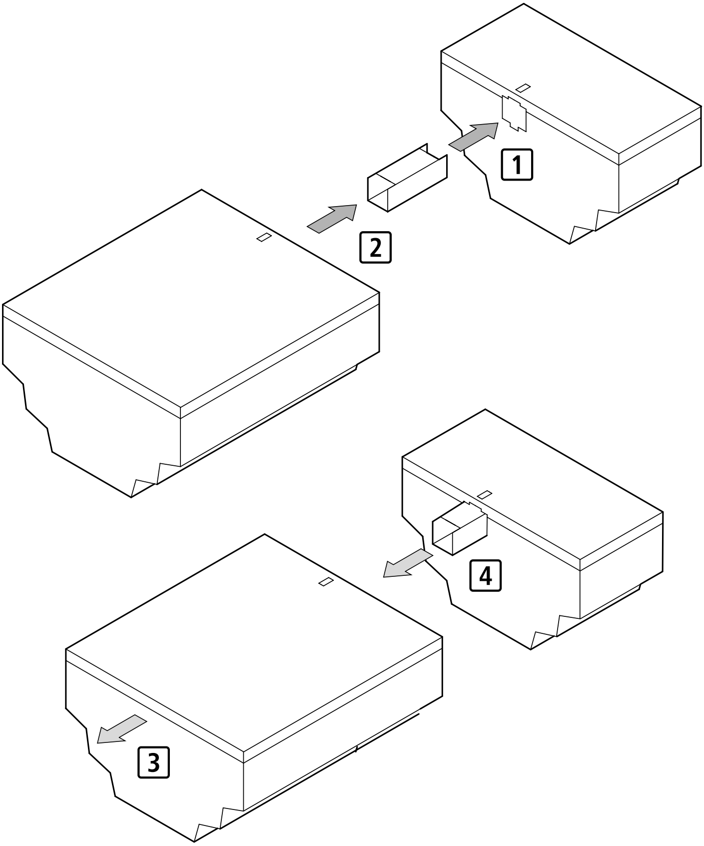


Figure 3: Fitting the EASY221-CO to the basic unit

1 + 2 Fitting

3 + 4 Removal

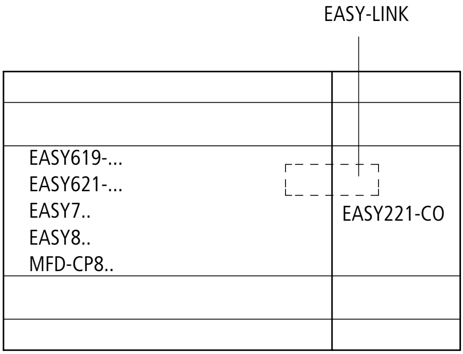
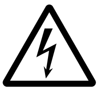


Figure 4: Connection between the basic unit and EASY221-CO

Connecting the power supply

The EASY221-CO device is run on a 24 V DC power supply (→ Technical data under "Power supply", page 269).



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

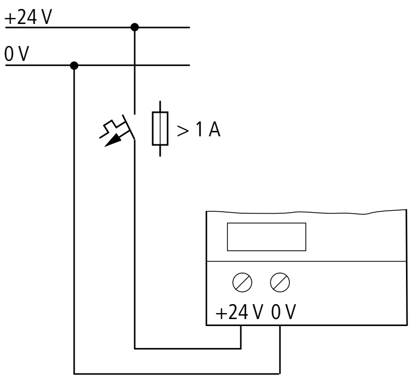


Figure 5: Power supply of the EASY221-CO

Connecting CANopen

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

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

Pin assignment of the CANopen

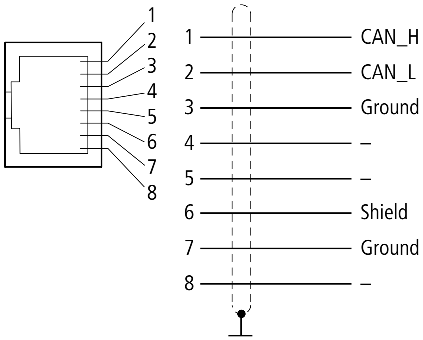


Figure 6: Pin assignment of the device socket

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

Bus terminating resistors

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

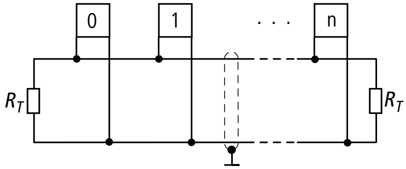


Figure 7: Terminating resistors R_T: CAN_H and CAN_L terminals

R_T = 120 Ω

EMC compliant wiring

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

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

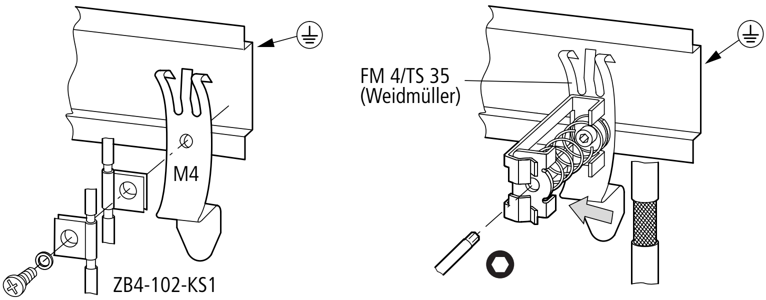


Figure 8: Shielding connection to the mounting rail

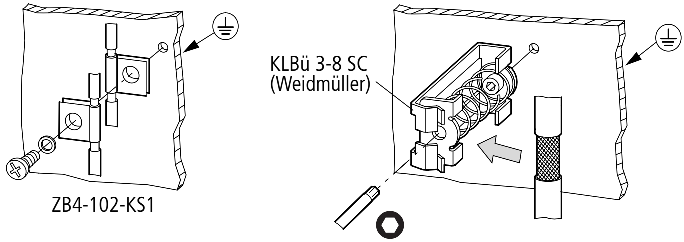


Figure 9: Shielding connection to the mounting plate

Electrical isolation

The following potential isolation must be provided for the interfaces of the EASY221-CO:

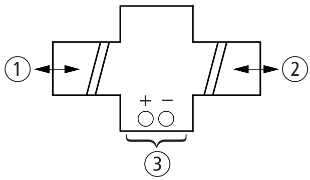


Figure 10: Potential isolation between the power supply and outputs

① Safe electrical isolation between EASY-LINK and 240 VAC

② Basic electrical isolation to the CANopen communication bus

③ Power supply 24 V DC

Transfer rates – automatic baud rate recognition

After it is switched on, the EASY221-CO module automatically detects the data transfer rate of the communication network. However, this requires that at least one station is transmitting valid telegrams in the network. The fast flashing Error and RUN LEDs of the 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 starts flashing and the ERR LED will be switched off.

The EASY221-CO supports the data transfer rates specified by CiA. The table below provides an overview of recommended bit rates and of the corresponding maximum cable lengths.

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

3 Device Operation

Initial power up

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

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

The LEDs of the EASY221-CO will flicker. The device is in the mode for determining the correct baud rate (→ section "Transfer rates – automatic baud rate recognition" on page 24). The GW message (intelligent station connected) must be displayed on the basic unit.

Basic unit	Device version	GW display
easy600	04	Static
easy700	From 01	Flashing
easy800	04	Static
	From 05	Flashing
MFD-CP8..	01	Static
	From 02	Flashing

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

If the EASY221-CO has its default settings (node ID = 127), you need to define the CANopen slave address.

Setting the CANopen slave address

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

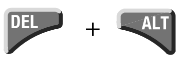
There are three ways to set the CANopen address of an EASY221-CO:

- Using the integrated display and keypad on the easy or MFD-Titan basic unit; address range: 1 to 127
- 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 on the basic unit with display

Requirements:

- The appropriate basic unit (easy600, easy700, easy800 or MFD-Titan) and EASY221-CO must be fed with power.
- The basic unit must have been unlocked (no password activated).
- The basic unit must have a valid operating system version.
- The basic unit must be in STOP mode.



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

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

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

▶ Use the cursor buttons ⌃ or ⌄ to change to CONFIGURATOR.

▶ Confirm with OK.

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

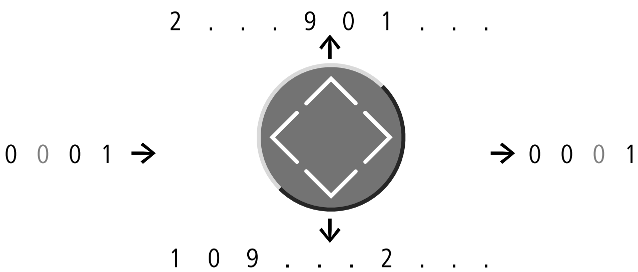
▶ With easy800/MFD devices select LINK...

▶ Confirm with OK.

The CANOPEN menu appears.

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

▶ Set the address with the cursor buttons:
  – Set the current numeric value with the ⌃ or ⌄ buttons.
  – You can change the actual numeric value via the < or > buttons.



▶ Press OK to accept the address.

▶ Press ESC to cancel address entry.

Information on the 4th display line:

xxx - xx.xx - xx

221 - 02.10 - B

- B: Hardware version, Index: b
- 02.10: Software version, OS version: 2.1
- 221: Device designation: EASY221-CO

Setting the address by means of EASY-SOFT

With EASY-SOFT, version 3.1
‹Menu   Online   Configure Expansion Devices›

With EASY-SOFT, from version 4.01
‹Menu → Communication → Configuration → Expansion Devices → EASY221-CO›.

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

→ The following applies to devices for device version 01:

After you have changed the NodeID via the basic unit you must restart EASY221-CO by switching power off and on. EASY221-CO devices with a version number > 01 accept the address automatically.

Setting the address via special configuration tools

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

Status LEDs

The EASY221-CO expansion unit is equipped with two LEDs: one green RUN LED and one red ERR LED. These indicate the current module status and allow quick error analysis.

Error LED

No.	Error LED	Status	Description
1	OFF	No error	The EASY221-CO is operating error-free. If the RUN LED is also off, 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 Controller has either reached or exceeded the Warning Limit. Too many errors have occurred on the CANopen bus.
3	Flickering	AutoBaud/LSS	Auto baud rate detection is currently busy (flickers in alternation with the RUN LED).
4	Flashes twice	Error control event	A protective Guard Event or a Heartbeat Event has occurred.
5	ON	Bus-off	The CANopen controller has changed to BUS-OFF status.

RUN LED

No.	RUN LED	Status	Description
1	OFF	Reset	The EASY221-CO is either switched off or is currently being reset.
2	Flickering	AutoBaud	Auto baud recognition is currently busy (LED flickers, in alternation with the ERR LED).
3	Single flash	STOPPED[1)]	The device is in STOPPED state.
4	Flashing	PRE-OPERATIONAL[1)]	The device is in PRE-OPERATIONAL status.
5	ON	OPERATIONAL[1)]	The device is in OPERATIONAL status.

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

Timing diagram of the ERR and RUN LEDs

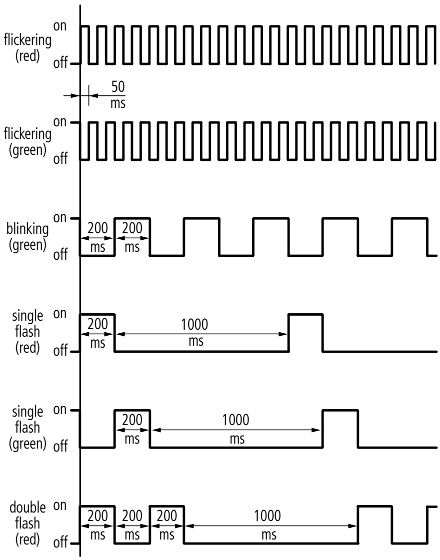


Figure 11: ERR and RUN LED

Cycle time of the easy basic unit

Communication between the basic unit and the EASY221-CO via EASY-LINK increases the cycle time of the basic unit.

The worst case value is 25 ms.

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

EDS file

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

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 the EASY221-CO on the CANopen bus are defined by the CANopen services.

Communication objects

The 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/write access to the entries of the object dictionary.

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

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

→ Block transfer is not supported!

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

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

→ The EASY221-CO does not support client SDOs.

Process data objects

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

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

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

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

Receive PDO

The EASY221-CO receives data from the CANopen network (PDO consumer) by means of receive PDOs and writes this data via EASY-LINK to the easy/MFD basic unit as a command or identifier for the image data R16 to R1.

Transmit PDOs

In the opposite direction, the commands or status of the S8 to S1 image data of easy600 are read via EASY-LINK and transmitted to the CANopen network as transmit PDOs of the EASY221-CO (PDO producer).

PDO mapping

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

Transmission types of PDOs

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

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

Inhibit Time

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

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

Event Timed PDOs

The expiration of a counter can be considered as an event which triggers the transmission of a PDO. The EASY221-CO does not support Event Timed PDOs by default, however the user can enable this function for transmit PDOs as required.

Multiplexed PDOs

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

→ The EASY221-CO does not support multiplexed PDOs.

PDO mapping

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

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

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

→ For details on the composition of the process data refer to section "Output data (2011_{hex}): operating mode, R1 – R16", page 78.

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

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

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

→ For details on the composition of the process data refer to section "Input data (2012_{hex}): operating mode, S1 – S8", page 81.

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

System services

Synchronisation object

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

Emergency object

The 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 55.

Network management

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

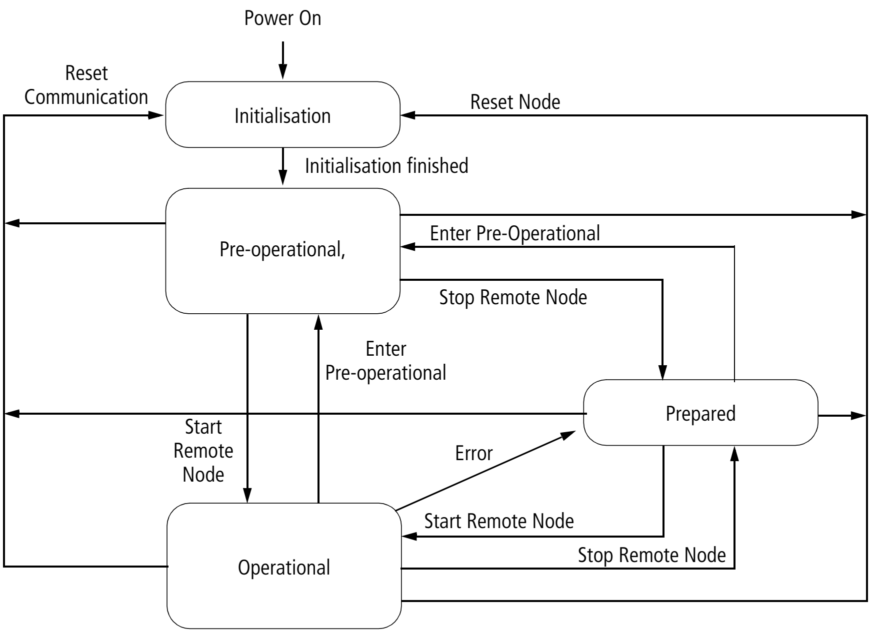


Figure 12: Network management

CANopen distinguishes between the following states:

- 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 next state, namely the Pre-operational state.

Pre-operational

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

The RUN LED of the EASY221-CO flashes to indicate this state.

From device version 03 the state is indicated also on the basic unit with the flashing GW message in the easy display. The diagnostics input I14 (on the basic unit) is set until GW is no longer flashing in the easy display. This is achieved by setting the CAN node to Operational mode.

Operational

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

The RUN LED of the EASY221-CO is static to indicate this state.

From device version 03 the status is indicated also on the basic unit by the static display of the GW status message. The diagnostics input I14 (on the basic unit) is set to zero.

Prepared

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

A Boot-Up message will be transmitted after power on of a device in order to indicate its ready state. This message frame uses the identifier of the NMT error control protocol and is permanently assigned to the set device address (1 792_{dec} + device address).

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

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

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

- Start Remote Node, transition to OPERATIONAL state; making it possible to transfer data via SDOs and PDOs.
- Enter Pre-operational, transition to PRE-OPERATIONAL state; it is possible to transfer data via SDOs.
- Reset Node and
- Reset Communication, transition to 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:

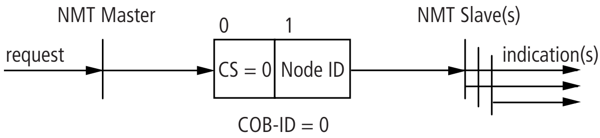


Figure 13: Structure of the NMT services, Start Remote Node

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

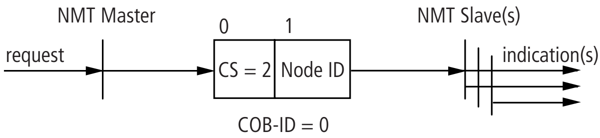


Figure 14: Structure of the NMT services, Stop Remote Node

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

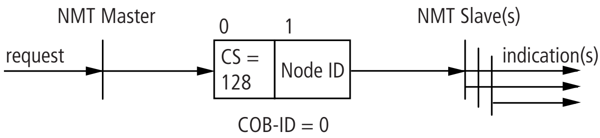


Figure 15: Structure of NMT services, PRE-OPERATIONAL state

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

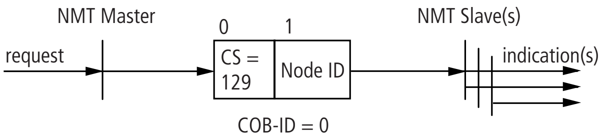


Figure 16: Structure of NMT services, Reset Node

Node ID = 0: Resets all existing nodes.

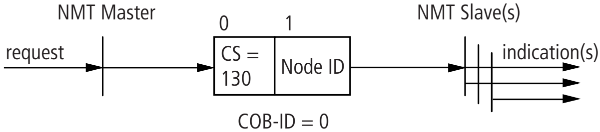


Figure 17: Structure of NMT services, Reset Communication

Node ID = 0: Resets all existing nodes.

Node monitoring

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

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

Node Guarding

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

Failure of Node Guarding

Error events triggered after the Life Time has expired and a Node Guard frame has not been received from the 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 and the outputs of the easy basic unit can now receive PDO data again.

Heartbeat Producer

The EASY221-CO broadcasts a cyclic heartbeat frame to signal its communication status. If a responsible heartbeat consumer does not receive this heartbeat frame within the Heartbeat Consuming Time, its application will report a heartbeat error. The second parameter relevant to the heartbeat protocol is the Heartbeat Producer Time, which can be set in the 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 on the EASY221-CO node, the first heartbeat frame will be transmitted during the transition from the Initialisation to the Pre-operational state. Concurrent use of both node monitoring methods is not allowed. The heartbeat protocol is used when the Heartbeat Producer Time is unequal to zero.

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

Further services

Saving and restoring entries

The EASY221-CO supports the saving and restoring of the object dictionary entries 1000_{hex} to $1FFF_{hex}$ in the non-volatile memory (EEPROM or FRAM). In the object dictionary tables, this area is named NVM-PA, while manufacturer-specific entries are stored in the NVM-RO area.

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

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

Layer Setting Service

The Layer Setting Service is used to configure the node ID via the CANopen network. The 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 on the EASY221-CO. To ensure that the correct node ID is displayed on the easy basic unit as well (Configurator menu), you must switch on the coupling module again.

Device profile

In the extension of the CiA-DS-301 communication profile which describes the communication mechanisms between nodes, the CANopen uses so-called device profiles for the essential device classes. The device profiles describe the device functions. The EASY221-CO cannot be assigned to an existing device profile.

5 Object Dictionary

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

Communication parameters

A detailed description of the communication parameters is provided 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 the 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	Pre-defined 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, device consumes the SYNC message

Index hex	Sub-index hex	Object name	Data type	Access location	FS hex	Meaning
1008	00	Manufacturer Device Name	VISIBLE_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}	Multiplier for the Guard Time, the result is equivalent to the maximum interval between the transfer of two Guarding message frames
1010	00	Store Parameters	UNSIGNED8	ro ROM	01	Max. number of storing options
	01	SAVE all Parameters	UNSIGNED32	rw RAM	wr: 65766173 rd: 00000001	→[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 device information
	01	Vendor ID	UNSIGNED32	ro NVM-RO	00000003	Manufacturer
	02	Product Code	UNSIGNED32	ro NVM-RO	0323353	Product number
	03	Revision Number	UNSIGNED32	ro NVM-RO	00010001 (Example)	Version
	04	Serial Number	UNSIGNED32	ro NVM-RO	4010016 (Example)	Serial number

1) The EASY221-CO supports up to 16 entries in the error log.

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

The 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 RxSDO. The ID is derived from the Predefined Connection Set.
	02	COB-ID Server → Client (tx)	UNSIGNED32	ro ROM	00000580 + node ID	COB-ID of the TxSDO. The ID is derived from the Predefined Connection Set.

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

Index hex	Subindex 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, according to [1]
	02	Transmission Type	UNSIGNED8	rw NVM-PA	FF	PDO transmission type: asynchronous

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

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

The 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	Subindex 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, according to [1]
	02	Transmission Type	UNSIGNED8	rw NVM-PA	FF	PDO transmission type: asynchronous
	03	Inhibit Time	UNSIGNED16	rw NVM-PA	0000	Inhibit time (min. time interval between the next transmission of a PDO) in ms 0000_{hex} = transmit now
	05	Event Timer	UNSIGNED16	rw NVM-PA	0000	Event counter 0000_{hex} = not used

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

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

Manufacturer-specific objects

In addition to the device profile objects, the object dictionary also contains the definitions of manufacturer-specific objects. The area between index 2000_{hex} and $5FFF_{hex}$ in the object dictionary of the 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 pable	FS hex	Meaning
2001[1)]	00	Coupling error	UNSIGNED8	ro easy	No	–	Error status of the EASY221-CO
2002[1)]	00	easy error	UNSIGNED8	ro easy	No	–	Error status of the easy/ MFD basic unit
2011	00	Output data	UNSIGNED24	rw easy	Yes	140000	Output data to the easy/ MFD basic unit
2012	00	Input data	UNSIGNED24	ro easy	Yes	–	Input data from the easy/MFD basic unit
2020	00	Status	UNSIGNED8	ro easy	No	FF	Status 00_{hex} = valid data, 01_{hex} = invalid data, FF_{hex} = Initial isation
2021	00	Command	DOMAIN Length = 7	rw easy	No	–	Command to the easy619/ 621
2022	00	Response	DOMAIN Length = 7	ro easy	No	–	Response from the easy619/621

Index hex	Sub-index hex	Object name	Data type	Access location	Map pable	FS hex	Meaning
3020	00	Status	UNSIGNED8	ro easy	No	FF	Status 00_{hex} = valid data, 01_{hex} = invalid data, FF_{hex} = Initial isation
3021	00	Command	DOMAIN Length = 8	rw easy/ MFD	No	–	Command to easy700/800, MFD-CP8..
3022	00	Response	DOMAIN Length = 8	ro easy/ MFD	No	–	Response from easy700/800, MFD-CP8..

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

Error messages (Emergency)

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

In the manufacturer-specific error entry (Manufacturer Specific Error Field), byte 0 outputs the error code of the 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 permanently set to 00_{hex}.

Data byte	Contents	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	Contents	Value	Description
1.	Error Code	1000_{hex}	Generic Error (→ [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: 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.

Attention
When communication between the easy600 basic unit and the EASY221-CO expansion unit 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 following protocols are used for the transfer of data via the CANopen bus:

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

Information on data contents → chapter 8 (easy600), 9 (easy700) and 10 (easy800/MFD).

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

PDO protocol

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

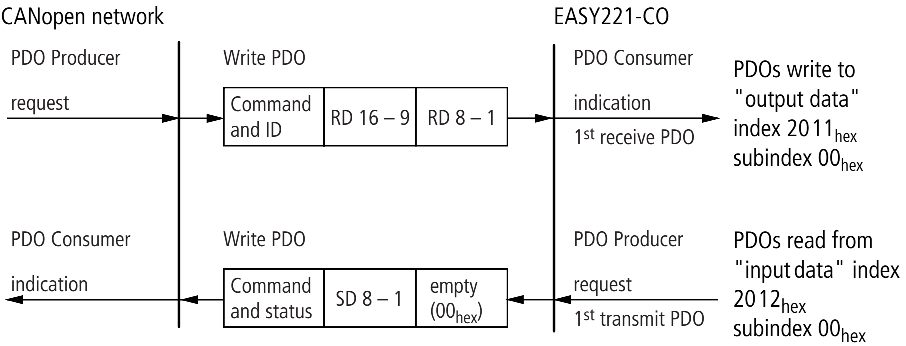


Figure 18: Write PDO Protocol

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

SDO protocol

General overview

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

Segmented transfer is required in order for data of any length to be transferred via an SDO channel since the maximum transfer capacity of a CAN telegram is only 8 bytes. This is based on the SDO protocols specified under [CIA-301].

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

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

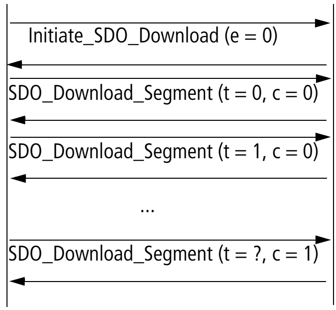


Figure 19: Segmented SDO download protocol

Expedited transfer protocol

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

Figure 20: Expedited SDO download protocol.

Control byte

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

Figure 21 shows the protocol for writing an OD entry using the expedited SDO protocol. The client control byte indicates that an Initiate Download Request is present. This byte also indicates the transfer type as "expedited transfer", as well as the number of data bytes contained in the data field. The server control byte indicates an Initiate Download Response accordingly. The logical address of the OD entry is then sent as a 16-bit index and 8-bit subdindex following the control bytes.

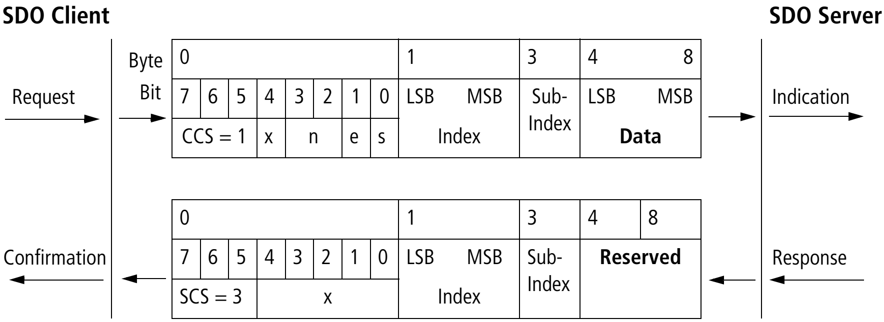


Figure 21: Writing an OD entry using the Expedited Domain Download Protocol

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

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

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

x = nc

CCS = 1 Client Command Specifier = initiate download request

SCS = 3 Server Command Specifier = initiate download response

→ The description of the CANopen protocol [CIA-301] is available on the Internet at http://www.can-cia.org.

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

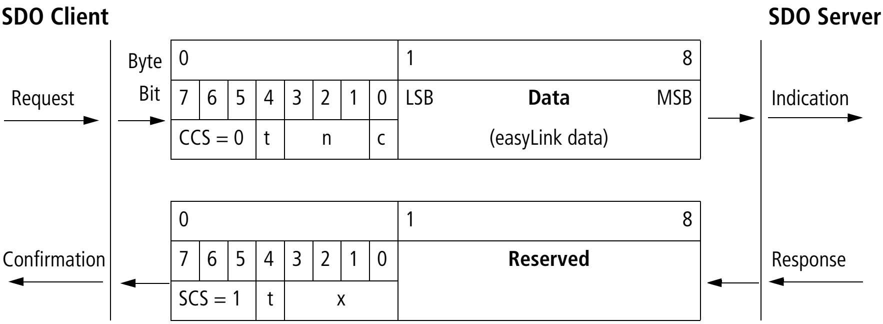


Figure 22: Download Transfer Segment after successful initialisation.

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

SDO protocol for easy/MFD

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

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

Device series	Object dictionary entries
easy600	Index 2021_{hex} Subindex 00_{hex}
easy 700/800/MFD	Index 3021_{hex} Subindex 00_{hex}

2 Two different data lengths are used for the SDO transfer with the EASY221-CO. These must be used according to the basic unit in place.

Device series	Length of the EASY-LINK data
easy600	7 bytes
easy700/800/MFD	8 bytes

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

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

Device series	Object dictionary entries
easy600	Index 2020_{hex} Subindex 00_{hex}
easy 700/800/MFD	Index 3020_{hex} Subindex 00_{hex}

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

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

Device series	Object dictionary entries
easy600	Index 2022_{hex} Subindex 00_{hex}
easy 700/800/MFD	Index 3022_{hex} Subindex 00_{hex}

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

6 Since this data also has a length of up to 7 bytes (easy600) and 8 bytes (easy700/800/MFD), a subsequent Upload SDO Segment is required in order to read the remaining data.

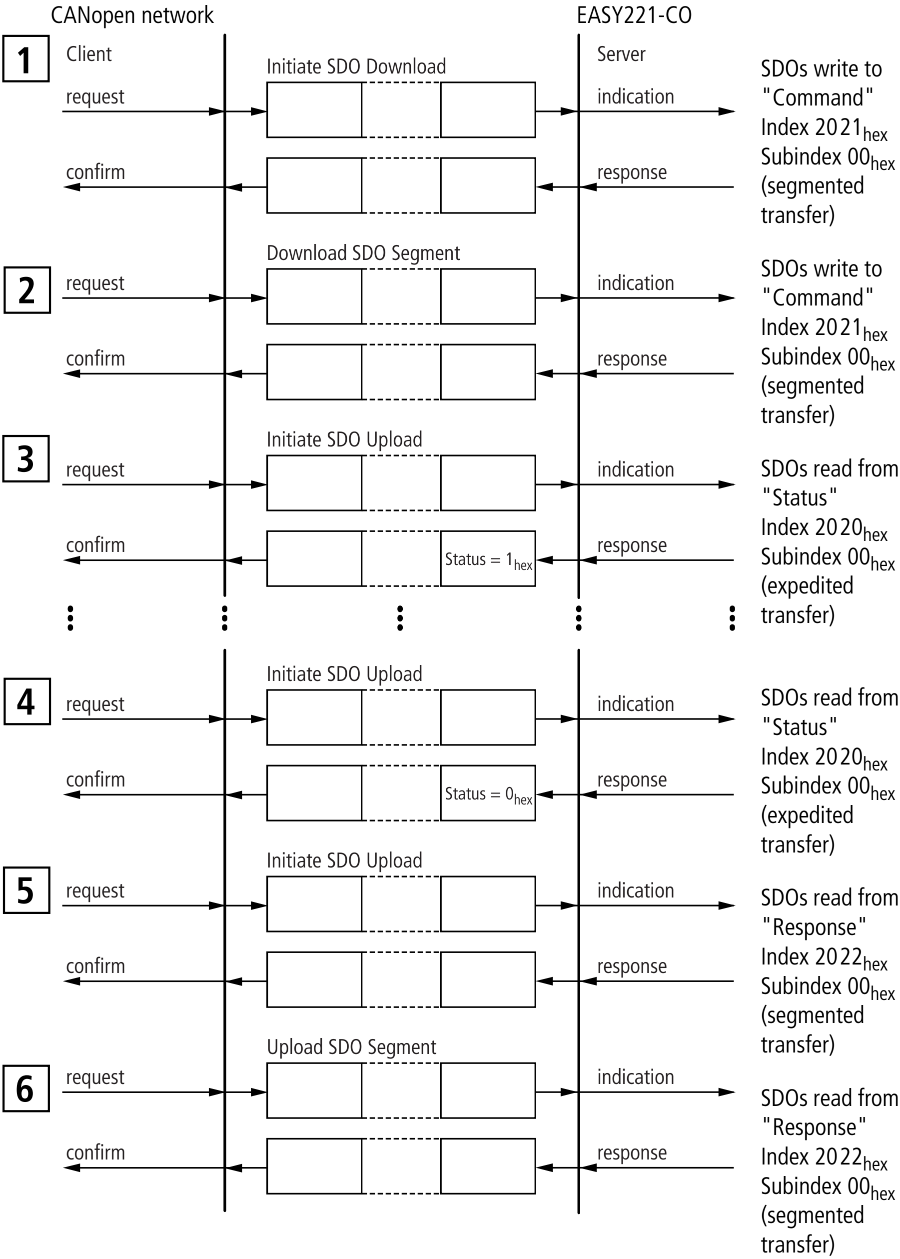


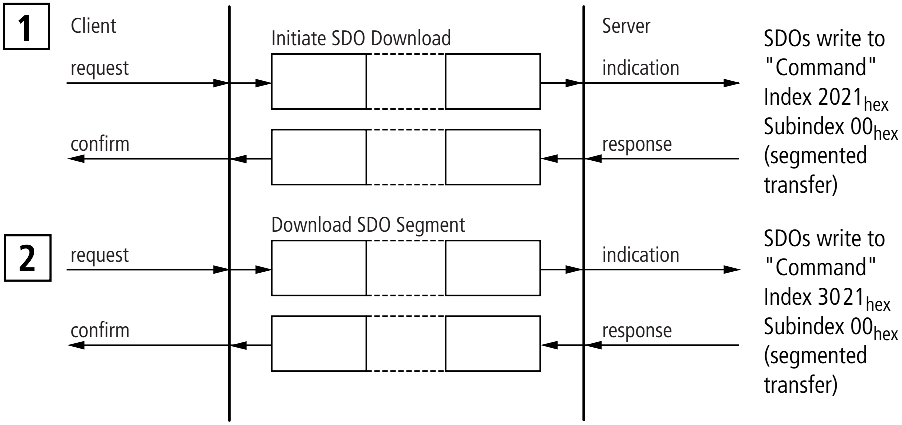
Figure 23: Sequence for extended SDO protocol (easy600)

Example of easy600: Read time (7 bytes)

The time is read from the basic unit via the SDO Transfer. The following easy telegram structure is specified in the manual (→ page 87).

Byte	Meaning	Value
0	Command: Read	5D
1	Weekday	00
2	Hour	00
3	Minute	00
4	Summer/winter time	00

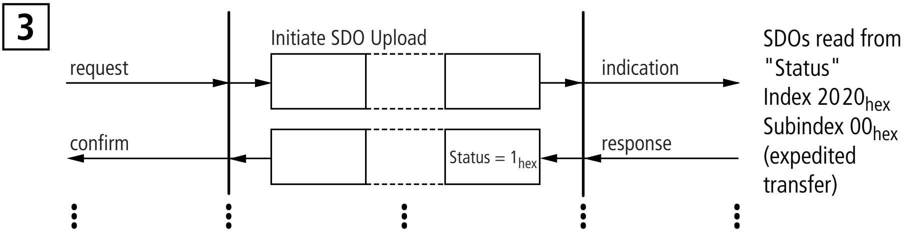
This data must be transferred with the CANopen protocol.

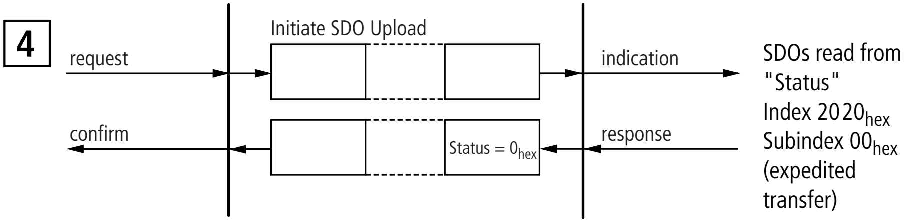


Description	ID (hex)	CAN data – byte (hex)							
Write command to EASY221-CO		0	1	2	3	4	5	6	7
Initialise the SDO download with 7 data bytes	602	21	21	20	00	**07**[1]	00	00	00
Confirmation of the SDO Block Transfer	582	60	21	20	00	00	00	00	00
Transfer of block 1 with 7 data bytes	602	01	**5D**[2]	**00**[2]	**00**[2]	**00**[2]	**00**[2]	**00**[2]	**00**[2]
Confirmation of the data block to be transferred	582	20	00	00	00	00	00	00	00

1) Number of easy-data bytes to be transferred: easy600 – 7 bytes

2) Valid data from easy basic unit

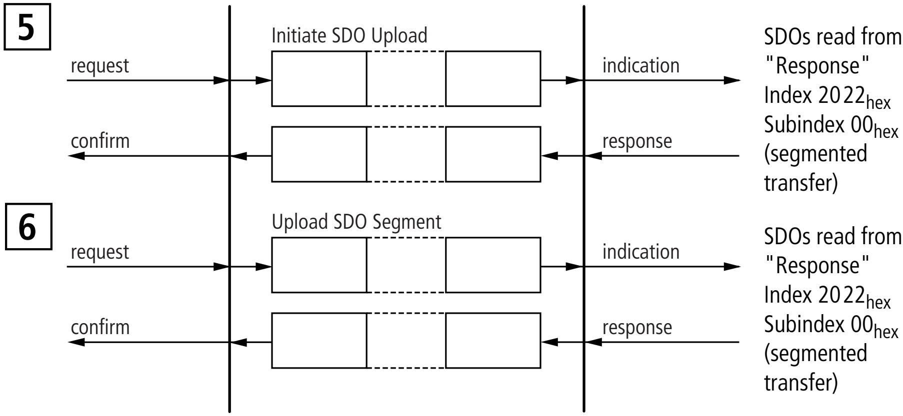




Description	ID (hex)	CAN data – byte (hex)							
Scan status		0	1	2	3	4	5	6	7
Initialise the SDO upload	602	40	20	20	00	00	00	00	00
Transfer of status byte	582	4F	20	20	00	**01**[3]	xx	xx	xx
New attempt		Data 1 is scanned via index 2020_{hex} and Subindex 00_{hex} until the value = 00_{hex}.							
Initialises the SDO upload	602	40	20	20	00	00	00	00	00
Transfer of status byte	582	4F	20	20	00	**00**[3]	xx	xx	xx

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

xx = Value has no meaning



Description	ID (hex)	CAN data – byte (hex)							
Call response		0	1	2	3	4	5	6	7
Initialise the SDO upload	602	40	22	20	00	00	00	00	00
Confirmation of the SDO Upload Block Transfer with 7 bytes	582	41	22	20	00	**07**[4]	xx	xx	xx
Scan of 1st data block	602	60	22	20	00	00	00	00	00
Transfer of the easy response byte	582	01	**42**[5]	**03**[5]	**17**[5]	**11**[5]	**00**[5]	**00**[5]	**00**[5]

4) Number of easy-data bytes to be transferred: easy600 – 7 bytes

5) Valid data from easy basic unit

xx = Value has no meaning

Evaluation of the received data

Byte	Meaning	Value
0	Response: read successful	42
1	Weekday	03
2	Hour	17
3	Minute	11
4	Summer/winter time	00

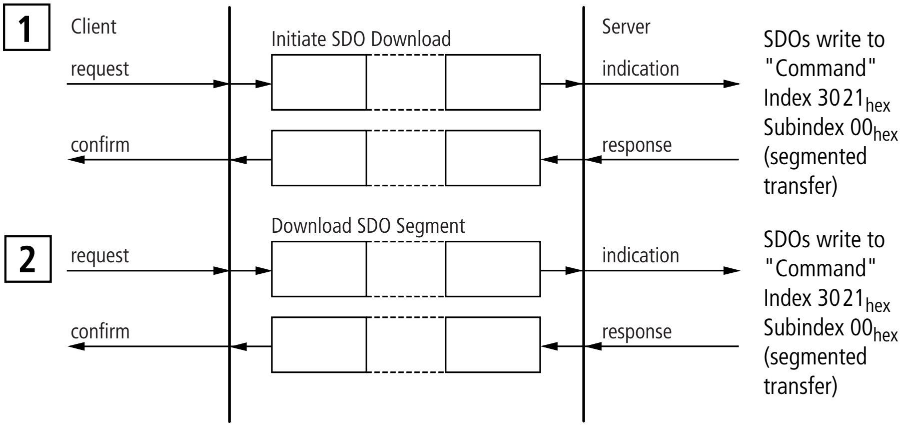
Thursday, 17:11 pm – winter time

Example of easy800: Read time (8 bytes)

The time is read from the basic unit via the SDO Transfer. The following easy telegram structure is specified for this in the manual (→ page 177).

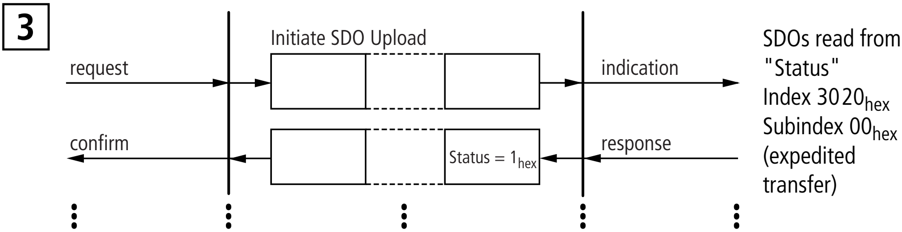
Byte	Meaning		Value
0	Command: Read		93
1	Len		05
2	Index		00
3	Data 1	Hour	00
4	Data 2	Minute	00
5	Data 3	Day	00
6	Data 4	Month	00
7	Data 5	Year	00

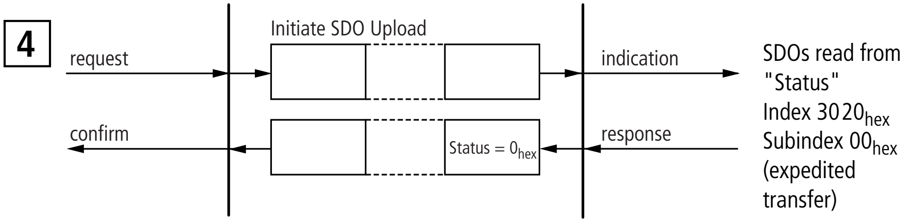
This data must be transferred with the CANopen protocol.



Description	**ID (hex)**	**CAN data – byte (hex)**							
Write command to EASY221-CO		0	1	2	3	4	5	6	7
Initialise the SDO download with 7 data bytes	602	21	21	30	00	**08**[1]	00	00	00
Confirmation of the SDO Block Transfer	582	60	21	30	00	00	00	00	00
Transfer of block 1 with 7 data bytes	602	00	**93**[2]	**05**[2]	**00**[2]	**00**[2]	**00**[2]	**00**[2]	**00**[2]
Confirmation of the data block to be transferred	582	20	00	00	00	00	00	00	00
Transfer of block 2 with 8th data byte	602	1D	**00**[2]	xx	xx	xx	xx	xx	xx
Confirmation of the data block to be transferred	582	30	00	00	00	00	00	00	00

1) Number of easy data bytes to be transferred: easy700/800/MFD – 8 bytes
2) Valid data from easy basic unit



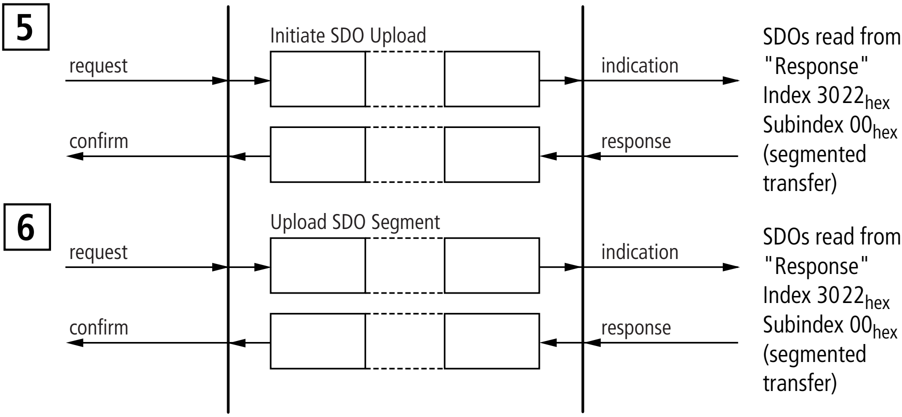


Description	**ID (hex)**	**CAN data – byte (hex)**							
Scan status		0	1	2	3	4	5	6	7
Initialise the SDO upload	602	40	20	30	00	00	00	00	00
Transfer of status byte	582	4F	20	30	00	**01**[3]	xx	xx	xx
New attempt		Data 1 is scanned via index 3020_{hex} and Subindex 00_{hex} until the value = 00_{hex}.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

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

xx = Value has no meaning

5



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

4) Number of easy data bytes to be transferred: easy700/800/MFD – 8 bytes

5) Valid data from easy basic unit

xx = Value has no meaning

Evaluation of the received data

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

22:31 pm, 01.05.2003

Emergency protocol

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

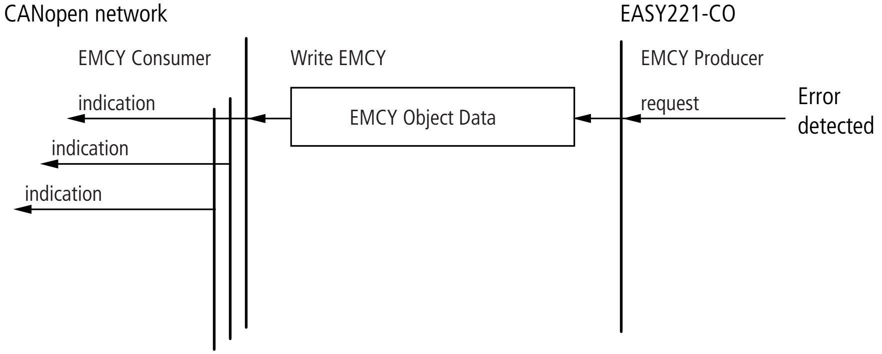


Figure 24: Emergency Object Protocol

7 PDO – Direct Data Exchange with easy/MFD

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

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

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

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

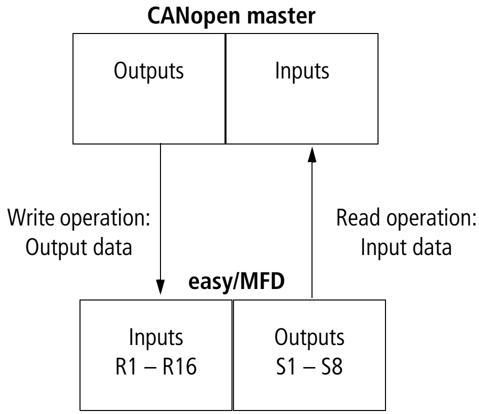


Figure 25: Input and output data as viewed from the CANopen master

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

The entries 2011_{hex} and 2012_{hex} can be mapped and can be transferred via PDOs. The object 2011_{hex} contains the output data of the CANopen master that is written via the EASY221-CO gateway to the inputs (R1 – R16) of the easy/MFD. The output data is provided in bytes 0 to 2 and is described in detail in the following tables:

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

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

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

Table 2: Byte 0: Operating mode

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

0 = status “0”, 1 = status “1”

Explanation:

Value 14_{hex} = 00010100_{bin}:
Byte 0 must always contain this value if data is to be written to the easy/MFD basic unit via the EASY221-CO gateway.

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

Value 44_{hex} = 01000100_{bin}:
This value sets the easy/MFD status from RUN to STOP. It is also used only as command and is therefore works in the same way as the RUN command.

Value 00_{hex} = 00000000_{bin}:
If this value is written to the control byte, the gateway overwrites the Rx data with zero. This function is only required if a master is to be set to STOP mode and as a resultant final measure transfers zero values in all mapped PDOs in order to ensure a safety state.

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

Table 3: Byte 1: Set/reset the easy/MFD inputs R9 to R16

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

0 = status “0”, 1 = status “1”

Example:
Value 19_{hex} = $0001\,1001_{bin}$:
Enables R13, R12 and R9.

Table 4: Byte 2: Set/reset the easy/MFD inputs R1 to R8

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

0 = status “0”, 1 = status “1”

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

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

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

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

The entries 2011_{hex} and 2012_{hex} can be mapped and can be transferred via PDOs. The object 2012_{hex} contains the output data of the easy/MFD (S data) that is transferred via the EASY221-CO gateway to the CANopen master. The tables below describe the structure of the input data in detail.

Table 5: Input data, operating mode

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

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

Table 6: Byte 0: Operating mode

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

0 = status “0” 1 = status “1”

Example:
Value 21_{hex} = $0010\,0001_{bin}$:
easy is in RUN mode and operates with debounce

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

easy/MFD 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} = $0001\,1001_{bin}$:
S5, S4 and S1 are active

Byte 2: not used

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

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

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

8 SDO – Control Commands for easy600

The object dictionary entries Status (2020_{hex}), Command (2021_{hex}) and Response (2022_{hex}) provide the interface for extended data exchange with easy600.

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

- "Date and time, Summer/winter time" (page 87)
- "Reading image data" (page 91)
- "Read/write function blocks" (page 102).

Attention!
The I/O data retains its 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 command code.
Verify data to be transferred in order to avoid unnecessary errors.

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

→ The basic unit must be in the mode of the LCD status display in order for the different parameters to be written.

The SDO download of a string for **Command** initiates the transfer of set data to easy600 with an extended protocol. After the data has been exchanged, the response of easy600 can be fetched from **Response** via the SDO upload, while **Status** returns the data transfer status. 01_{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 response can be read from **Response**. **Status** 01_{hex} indicates either the occurrence of a data transfer error or that invalid data was entered in **Command**. In this case **Response** contains undefined data.

Date and time, Summer/winter time — Telegram structure

Byte	Meaning		Value (hex), sent by		Bit							
			Master	Slave	7	6	5	4	3	2	1	0
0	Command											
		Read	5D	–	0	1	0	1	1	1	0	1
		Write	2A	–	0	0	1	0	1	0	1	0
	Response											
		Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
		Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
		Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Weekday											
		Read operation	00	→ table 8								
		Write operation	→ table 8	00								
2	Hour											
		Read operation	00	→ table 9								
		Write operation	→ table 9	00								
3	Minute											
		Read operation	00	→ table 10								
		Write operation	→ table 10	00								
4	Summer/winter time											
		Read operation	00	→ table 11								
		Write operation	→ table 11	00								

Table 8: Byte 1: Weekday (value range 00 to 06)

Weekday	**Bit**							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Monday = 00	0	0	0	0	0	0	0	0
Tuesday = 01	0	0	0	0	0	0	0	1
Wednesday = 02	0	0	0	0	0	0	1	0
Thursday = 03	0	0	0	0	0	0	1	1
Friday = 04	0	0	0	0	0	1	0	0
Saturday = 05	0	0	0	0	0	1	0	1
Sunday = 06	0	0	0	0	0	1	1	0

Table 9: Byte 2: Hour (value range 00 to 23)

Value (bcd)	**Value 10**				**Value 1**			
	Bit				**Bit**			
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
....								
9	0	0	0	0	1	0	0	1
....								
14	0	0	0	1	0	1	0	0
...								
23	0	0	1	0	0	0	1	1

Table 10: Byte 3: Minute (value range 00 to 59)

Value (bcd)	Value 10				Value 1			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
...								
10	0	0	0	1	0	0	0	0
...								
21	0	0	1	0	0	0	0	1
...								
42	0	1	0	0	0	0	1	0
...								
59	0	1	0	1	1	0	0	1

Table 11: Byte 4: Winter/summer time (value range 00 to 01)

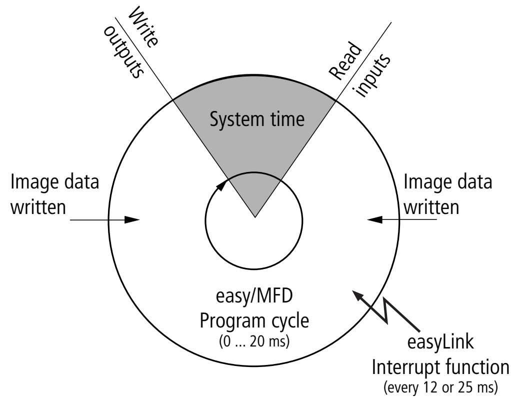
Value (bcd)	Value 10				Value 1			
	Bit				Bit			
Function	7	6	5	4	3	2	1	0
Winter time = 00	0	0	0	0	0	0	0	0
Summer time = 01	0	0	0	0	0	0	0	1

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

Byte	**Meaning**	**Value (hex), sent by**		**Bit**							
		Master	**Slave**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	Command										
	Write	2A	–	0	0	1	0	1	0	1	0
	Response										
	Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
1	Weekday	04	00								
2	Hour (14_{dec})	0E	00								
3	Minute (36_{dec})	24	00								
4	Summer/winter time	01	00								

Reading image data

General notes on working with image data



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

Overview

Operands	Meaning	Read/ write	Comm and	Page
I1 – I16, P1 – P4, ESC/OK/DEL/ALT	"Digital inputs, P and operating buttons"	Read	5C	92
I7 – I8	"Analog inputs: I7 – I8"	Read	5B	95
T1 – T8, C1 – C8, ⌚1 – ⌚4, A1 – A8	"Timing relays, counter relays, time switches and analog value comparators"	Read	5E	96
M1 – M16, Q1 – Q8, D1 – D8	"Markers, digital outputs and text display markers"	Read	5F	99

Digital inputs, P and operating buttons

The following command reads the logic state of the P1 to P4 digital button inputs as well as the logic state of the digital inputs I1 to I16.

The status of the P buttons is only displayed if

- a P button is used in the circuit diagram and
- the pushbuttons are activated on the device.

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Read	5C	–	0	1	0	1	1	1	0	0
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Status of inputs I1 to I8	00	→ table 12								
2	Status of inputs I9 to I16	00	→ table 13								
3	Status of the buttons	00	→ table 14								

Table 12: Byte 1: Status of inputs I1 to I8

Value	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							

Value 0 = switched off, Value 1 = switched on

Table 13: Byte 2: Status of inputs I9 to I16

Value	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							

Value 0 = switched off, Value 1 = switched on

Table 14: Byte 3: Status of buttons

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							

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

Analog inputs: I7 – I8

The following command is used to read the values of both analog inputs I7, I8 (only EASY...-DC-..).

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Read	5B	–	0	1	0	1	1	0	1	1
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Analog value of I7	00	See below								
2	Analog value of I8	00									

Analog inputs I7 and I8 (byte 1 and byte 2)
These two bytes contain the actual value of the analog inputs I7 and I8. Their value lies between 00 and 99, which is equivalent to a voltage 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 successfully. Data follows.
1	20_{hex}	Voltage level at input I7 = 3.2 V.
2	31_{hex}	Voltage level at input I8 = 4.9 V.

Timing relays, counter relays, time switches and analog value comparators

The following command reads the logic state of all timing relays, counters, time switches and analog value comparators.

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Read	5E	–	0	1	0	1	1	1	1	0
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Timing relay status	00	→ table 15								
2	Counter relay status	00	→ table 16								
3	Time switch status	00	→ table 17								
4	Analog value comparator status	00	→ table 18								

Table 15: Byte 1: Status of timing relays

	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							

Example:
Value $2B_{hex} = 0010\,1011_{bin}$:
T6, T4, T2 and T1 are active.

Table 16: Byte 2: Status of the counter relays

	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							

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

Table 17: Byte 3: Status of time switches

	Bit							
	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
1								0/1
2							0/1	
3						0/1		
4					0/1			
				0				
			0					
		0						
	0							

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

Table 18: Byte 4: Status of analog value comparators

	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							

Example:
Value $84_{hex} = 1000\,1000_{bin}$:
A3 and A8 are active.

Markers, digital outputs and text display markers

The following command will read the logic state of all markers M1 to M16, digital outputs Q1 to Q8, text markers D1 to D8.

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Read	5F	–	0	1	0	1	1	1	1	1
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Status of markers M1 to M8	00	→ table 19								
2	Status of markers M9 to M16	00	→ table 20								
3	Status of digital outputs Q1 to Q8	00	→ table 21								
4	Status of text display markers D1 to D8	00	→ table 22								

Table 19: Byte 1: Status of markers M1 to M8

	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							

Example:
Value $2B_{hex}$ = $0010\,1011_{bin}$:
M6, M4, M2 and M1 are active.

Table 20: Byte 2: Status of markers M9 to M16

	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							

Example:
Value 19_{hex} = $0001\,1001_{bin}$:
M13, M12 and M9 are active

Table 21: Byte 3: Status of digital outputs Q1 to Q8

	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							

Example:
Value $A8_{hex}$ = $1010\,1000_{bin}$:
Q8, Q6 and Q4 are active.

Table 22: Byte 4: Status of text display markers D1 to D8

	Bit							
	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							

Example:
Value 84_{hex} = $1000\,0100_{bin}$:
D3 and D8 are active.

Read/write function blocks

Overview

The first data byte of the string to be written for **Command** is a Command to easy600 and defines the meaning of the remaining six data bytes. The table below lists the possible commands.

Operands	Meaning	Command	Page
A1 – A8	"Analog value comparators A1 – A8: write actual values (function, comparison values)"	22_{hex} – 29_{hex}	103
C1 – C8	"Counter relays C1 – C8: read actual value"	$3B_{hex}$ – 42_{hex}	106
	"Counter relays C1 – C8: write setpoint"	09_{hex} – 10_{hex}	108
	"Counter relays C1 – C8: read setpoint"	43_{hex} – $4A_{hex}$	110
T1 – T8	"Timing relays T1 – T8: read actual value (time base, actual value, switch function)"	$2B_{hex}$ – 32_{hex}	112
	"Timing relays T1 – T8: write parameters (time base, setpoint, switch function)"	01_{hex} – 08_{hex}	116
⊙1 – ⊙4	Time switch ⊙1 – ⊙4: read actual values (channel, ON time, OFF time)	$4B_{hex}$ – $5A_{hex}$	120
	Time switch ⊙1 – ⊙4: write setpoints (channel, ON time, OFF time)	12_{hex} – 21_{hex}	124

Analog value comparators A1 – A8: write actual values (function, comparison values)

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	A1	22	–	0	0	1	0	0	0	1	0
	A2	23	–	0	0	1	0	0	0	1	1
	A3	24	–	0	0	1	0	0	1	0	0
	A4	25	–	0	0	1	0	0	1	0	1
	A5	26	–	0	0	1	0	0	1	1	0
	A6	27	–	0	0	1	0	0	1	1	1
	A7	28	–	0	0	1	0	1	0	0	0
	A8	29	–	0	0	1	0	1	0	0	1
	Response										
	Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte	→ table 23	Invalid								
2	Comparison value for comparison with constant	→ page 104	00								

→ Keep within the limits of the value range: the comparison values and the function are part of an *.eas file. If these values are changed, the original *.eas file will no longer match the file in EASY6..

Remember this characteristic when uploading, downloading or comparing easy circuit diagrams with EASY-SOFT. When downloading from the PC the latest version of the "*.eas" is overwritten. The comparison shows that the circuit diagrams are not identical.

Table 23: Byte 1: control byte

Meaning	Bit 7	6	5	4	3	2	1	0
Compare: "≧"								0
Compare: "≦"								1
I7 with I8						0	0	
I7 with constant						0	1	
I8 with constant						1	0	
Fixed			0	0	0			
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execute	1							

Example:
82_{hex} = $1000\,0010_{bin}$ means that the selected analog value comparator will be enabled in the circuit diagram of the basic unit as soon as the analog value at input I7 ≧ the defined constant (→ byte 2).

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

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

Example

The analog value comparator A8 has the following settings:

- Compare I7 < 4.7 V

The master initiates the command to reduce the comparison value to 4.2 V.

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Command: A8	29	0	0	1	0	1	0	0	1
	Response: Write successful	–	0	1	0	0	0	0	0	1
1	Control byte	→	1	0	0	0	0	0	1	1
2	Comparison value for comparison with constant	2A	0	0	1	0	1	0	1	0

The slave responds with the following telegram:

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Response: Write successful	41	0	1	0	0	0	0	0	1
1	Comparators	Invalid								
2	Comparison value for comparison with constant	00								

Counter relays C1 – C8: read actual value

Telegram structure

Byte	Meaning		Value (hex), sent by		Bit							
			Master	Slave	7	6	5	4	3	2	1	0
0	Command: Read											
		C1	3B	–	0	0	1	1	1	0	1	1
		C2	3C	–	0	0	1	1	1	1	0	0
		C3	3D	–	0	0	1	1	1	1	0	1
		C4	3E	–	0	0	1	1	1	1	1	0
		C5	3F	–	0	0	1	1	1	1	1	1
		C6	40	–	0	1	0	0	0	0	0	0
		C7	41	–	0	1	0	0	0	0	0	1
		C8	42	–	0	1	0	0	0	0	1	0
	Response											
		Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
		Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte		00	→ table 24	x	x	x	x	x	x	x	x
2	Counter relay actual value (low byte)		00	→ page 107								
3	Counter relay actual value (high byte)		00									

Table 24: Byte 1: control byte

Meaning	Bit 7	6	5	4	3	2	1	0
n.c.			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execute (processed in the circuit diagram)	1							

Example:
Value 80_{hex} = $1000\,0000_{bin}$:
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 C1 – C8: write setpoint

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	C1	09	–	1/0	0	0	0	1	0	0	1
	C2	0A	–	1/0	0	0	0	1	0	1	0
	C3	0B	–	1/0	0	0	0	1	0	1	1
	C4	0C	–	1/0	0	0	0	1	1	0	0
	C5	0D	–	1/0	0	0	0	1	1	0	1
	C6	0E	–	1/0	0	0	0	1	1	1	0
	C7	0F	–	1/0	0	0	0	1	1	1	1
	C8	10	–	1/0	0	0	1	0	0	0	0
	Response										
	Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte	→ table 25	00								
2	Setpoint value (low byte)	→ page 109	00								
3	Setpoint value (high byte)		00								

Value range of the counter values: 0000 to 9999

→ Keep within the value range.

The value is part of an "*.eas file" (EASY-SOFT file). If these values are changed, the original *.eas file will no longer match the file in EASY6..

Remember this characteristic when uploading, downloading or comparing easy circuit diagrams with EASY-SOFT.

When downloading from the PC the latest version of the "*.eas" is overwritten.

The comparison shows that the circuit diagrams are not identical.

Table 25: Byte 1: control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
n.c.			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execute	1							

Example:
Value 80_{hex} = 1000000_{bin}:
The setpoint is written to the selected counter relay and appears in the parameter menu.

Writing the setpoint (byte 2 and byte 3)
These two bytes determine the setpoint of the counter relay. The setpoint 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:
Setpoint = 4318_{dec} = $10DE_{hex}$:
Low value: DE_{hex}
High value: 10_{hex}

Counter relays C1 – C8: read setpoint

Telegram structure

Byte	Meaning		Value (hex), sent by		Bit							
			Master	Slave	7	6	5	4	3	2	1	0
0	Command											
		C1	43	–	0	1	0	0	0	0	1	1
		C2	44	–	0	1	0	0	0	0	1	0
		C3	45	–	0	1	0	0	0	1	0	1
		C4	46	–	0	1	0	0	0	1	1	0
		C5	47	–	0	1	0	0	0	1	1	1
		C6	48	–	0	1	0	0	1	0	0	0
		C7	49	–	0	1	0	0	1	0	0	1
		C8	4A	–	0	1	0	0	1	0	1	0
	Response											
		Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
		Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte		00	→ table 26								
2	Counter relay setpoint (low byte)		00	→ page 111								
3	Counter relay setpoint (high byte)		00									

Table 26: Byte 1: control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
n.c.			0	0	0	0	0	0
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execute (is being processed in the circuit diagram)	1							

Example:
Value $80_{hex} = 1000\,0000_{bin}$:
The setpoint of the counter relay is set and appears in the parameter menu.

Setpoint value (byte 2 and byte 3)
These two bytes determine the setpoint of the counter relay. The setpoint can lie within the value range 0 to 9999_{dec}. In order to determine the setpoint, 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}

Timing relays T1 – T8: read actual value (time base, actual value, switch function)

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command: Read										
	T1	2B	–	0	0	1	0	1	0	1	1
	T2	2C	–	0	0	1	0	1	1	0	0
	T3	2D	–	0	0	1	0	1	1	0	1
	T4	2E	–	0	0	1	0	1	1	1	0
	T5	2F	–	0	0	1	0	1	1	1	1
	T6	30	–	0	0	1	1	0	0	0	0
	T7	31	–	0	0	1	1	0	0	0	1
	T8	32	–	0	0	1	1	0	0	1	0
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte	00	→ table 27								
2	Time actual value (low byte)	00	→ page 114								
3	Time actual value (high byte)	00									
4	Random value	00	→ page 114								
5 – 6		00	00								

Table 27: Byte 1: control byte

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
Single pulse						1	0	0
Flashing						1	0	1
Time base "s"				0	0			
Time base "M:S"				0	1			
Time base "H:M"				1	0			
n.c.			0					
Appears in the parameter menu		0						
Does not appear in the parameter menu		1						
Timing relay not processed by operating system	0							
Timing relay processed by operating system	1							

Actual value (byte 2 and byte 3)

These two bytes determine the actual value of the timing relay. The actual value also depends on the set time base. When the control byte is set to a seconds time base, 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 produces the following table:

Table 28: Byte 2 to 3: time actual value

Time base	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, time base in [s].
High value $2D_{hex}$: Equivalent to 45 min, time base in [s]

Random value (byte 4)

easy sets a random delay time between zero and the defined setpoint for relays operating with random switching characteristics. This setpoint time is specified at this byte in hexadecimal format.

Example

The master initiates the command for reading timing relay T1:

Byte	Meaning	Value (hex)	Bit 7	6	5	4	3	2	1	0
0	Command: T1	2B	0	0	1	0	1	0	1	1
1 – 3		00								

The slave responds with the following values:

Byte	Meaning	Value (hex)	Bit 7	6	5	4	3	2	1	0
0	Response: Read successful	C2	1	1	0	0	0	0	1	0
1	Trigger coil activated, M:S time base, on-delayed, Parameter display +	→	1	0	0	0	1	0	0	0
2	Time actual value (low byte)	10	0	0	0	1	0	0	0	0
3	Time actual value (high byte)	0E	0	0	0	0	1	1	1	0

Value Set time = $0E10_{hex}$ = 3600

3600 s = 60:00 M:S

Timing relays T1 – T8: write parameters (time base, setpoint, switch function)

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command: Write										
	T1	01	–	0	0	0	0	0	0	0	1
	T2	02	–	0	0	0	0	0	0	1	0
	T3	03	–	0	0	0	0	0	0	1	1
	T4	04	–	0	0	0	0	0	1	0	0
	T5	05	–	0	0	0	0	0	1	0	1
	T6	06	–	0	0	0	0	0	1	1	0
	T7	07	–	0	0	0	0	0	1	1	1
	T8	08	–	0	0	0	0	1	0	0	0
	Response										
	Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte	→ table 29	Invalid								
2	Low setpoint value	→ page 119	00								
3	High setpoint value										
4 – 6		00	00								

→ Time values over 60s are converted to minutes.
Time values over 60 min. are converted to hours.
Time values over 24 h are converted to days.

The value range of the times and the timing relay setpoint are part of an "*.eas file". If these values are changed, the original *.eas file will no longer match the file in EASY6..

Remember this characteristic when uploading, downloading or comparing "easy" circuit diagrams with EASY-SOFT.

When downloading from the PC the latest version of the "*.eas" is overwritten.

The comparison shows that the circuit diagrams are not identical.

Value range of the time values

- "S" 00.00 to 99.99
- "M:S" 00:00 to 99:59 (M = 00 to 99, S = 00 to 59)
- "H:M" 00:00 to 99:59 (H = 00 to 99, M = 00 to 59)

→ Only the bytes reserved for the required time base should be used.

Table 29: Byte 1: control byte

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
Single pulse						1	0	0
Flashing						1	0	1
Time base “s”				0	0			
Time base “M:S”				0	1			
Time base “H:M”				1	0			
n.c.			0					
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Execute	1							

Example:
Value 89_{hex} = $10001001_{bin.}$
The timing relay operates with off-delay and time base in [s].

Timing relay, writing the setpoint (byte 2 and byte 3)
Bytes 2 and 3 determine the setpoint for the timing relay. The setpoint depends on the selected time base. When the time base of the control byte is set to seconds, the low value is based on seconds and the high value on the next higher time base (minute). The value range for each byte in this case is 0 to 59_{dec} ($3B_{hex}$). This produces the following table:

Time base	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 time base of seconds
High value $2D_{hex}$: Equivalent to 45 min, at a set time base of seconds

Time switch Ö1 – Ö4: read actual values (channel, ON time, OFF time)

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Ö1 channel A	4B	–	0	1	0	0	1	0	1	1
	Ö1 channel B	4C	–	0	1	0	0	1	1	0	0
	Ö1 channel C	4D	–	0	1	0	0	1	1	0	1
	Ö1 channel D	4E	–	0	1	0	0	1	1	1	0
	Ö2 channel A	4F	–	0	1	0	0	1	1	1	1
	Ö2 channel B	50	–	0	1	0	1	0	0	0	0
	Ö2 channel C	51	–	0	1	0	1	0	0	0	1
	Ö2 channel D	52	–	0	1	0	1	0	0	1	0
	Ö3 channel A	53	–	0	1	0	1	0	0	1	1
	Ö3 channel B	54	–	0	1	0	1	0	1	0	0
	Ö3 channel C	55	–	0	1	0	1	0	1	0	1
	Ö3 channel D	56	–	0	1	0	1	0	1	1	0
	Ö4 channel A	57	–	0	1	0	1	0	1	1	1
	Ö4 channel B	58	–	0	1	0	1	1	0	0	0
	Ö4 channel C	59	–	0	1	0	1	1	0	0	1
	Ö4 channel D	5A	–	0	1	0	1	1	0	1	0
	Response										
	Read successful	–	C2/42	1/0	1	0	0	0	0	1	0
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Time switch control byte	00	→ table 30								
2	Channel control byte	00	→ table 31								
3	Minute (switch point ON)	00	→ page 123								
4	Hour (switch point ON)	00									
5	Minute (switch point OFF)	00									
6	Hour (switch point OFF)	00									

Table 30: Byte 1: time switch control byte

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

Example:
Value 80_{hex} = $1000\,0000_{bin}$:
The addressed time switch is used in the circuit diagram.

Channel control byte
(Weekday: starting/ending, parameter menu display)
Each channel of a 7-day time switch is assigned a control byte that defines the start/stop conditions. The table below shows the precise structure of this control byte.

Table 31: Byte 2: channel control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
ON day								
None 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
OFF day								
None 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 7-day time switch Y is active Monday through Saturday.

Switching times (byte 3 to byte 6)

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 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 follows.
1	80_{hex}	The addressed time switch is used in the circuit diagram.
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}	

Time switch Ö1 – Ö4: write setpoints (channel, ON time, OFF time)

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command										
	Ö1 channel A	12	–	0	0	0	1	0	0	1	0
	Ö1 channel B	13	–	0	0	0	1	0	0	1	1
	Ö1 channel C	14	–	0	0	0	1	0	1	0	0
	Ö1 channel D	15	–	0	0	0	1	0	1	0	1
	Ö2 channel A	16	–	0	0	0	1	0	1	1	0
	Ö2 channel B	17	–	0	0	0	1	0	1	1	1
	Ö2 channel C	18	–	0	0	0	1	1	0	0	0
	Ö2 channel D	19	–	0	0	0	1	1	0	0	1
	Ö3 channel A	1A	–	0	0	0	1	1	0	1	0
	Ö3 channel B	1B	–	0	0	0	1	1	0	1	1
	Ö3 channel C	1C	–	0	0	0	1	1	1	0	0
	Ö3 channel D	1D	–	0	0	0	1	1	1	0	1
	Ö4 channel A	1E	–	0	0	0	1	1	1	1	0
	Ö4 channel B	1F	–	0	0	0	1	1	1	1	1
	Ö4 channel C	20	–	0	0	1	0	0	0	0	0
	Ö4 channel D	21	–	0	0	1	0	0	0	0	1
	Response										
	Write successful	–	C1/41	1/0	1	0	0	0	0	0	1
	Command rejected	–	C0/40	1/0	1	0	0	0	0	0	0
1	Control byte (Day beginning/end)	→ page 125	00								
2	Minute (switch point ON)	→ page 127	00								
3	Hour (switch point ON)		00								
4	Minute (switch point OFF)		00								
5	Hour (switch point OFF)		00								
6	Not used										

→ Keep within the value range: the values of minute and hour of the switch points are part of an EASY-SOFT file (*.eas). If these values are changed, the original "*.eas file" will no longer match the file in EASY6.. the file in EASY6..

Remember this characteristic when uploading, downloading or comparing "easy" circuit diagrams with EASY-SOFT. When downloading from the PC the latest version of the "*.eas" is overwritten. The comparison shows that the circuit diagrams are not identical.

Control byte (Weekday: starting/ending, parameter menu display)
Each channel of a 7-day time switch is assigned a control byte that defines the start/stop conditions. The table below shows the precise structure of this control byte.

Table 32: Byte 1: Control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
ON day								
None 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
OFF day								
None 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 7-day time switch Y is active Monday through Saturday.

Writing 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	**Command/byte**	**Value**
Data of channel A of time switch 4:	Command/byte 0	$1E_{hex}$
Day: Monday through Saturday 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}

Example

The master initiates the command to write the following data to channel C of ◷2:

- Day: Tuesday (010) to Saturday (110)
- ON: 10:00
- OFF: 17:30
- Switch point ON < OFF (0)
- Channel does not appear in the Parameters menu (1)

Byte	Meaning	Value	Bit 7	6	5	4	3	2	1	0
0	Command: ◷2 channel C	18_{hex}	0	0	0	1	1	0	0	0
1	Weekday, Parameter menu display	$B2_{hex}$	1	0	1	1	0	0	1	0
2	Minute (switch point ON)	00_{bcd}	0	0	0	0	0	0	0	0
3	Hour (switch point ON)	10_{bcd}	0	0	0	1	0	0	0	0
4	Minute (switch point OFF)	30_{bcd}	0	0	1	1	0	0	0	0
5	Hour (switch point OFF)	17_{bcd}	0	0	0	1	0	1	1	1
6	Not used									

The slave responds with the following telegram:

Byte	Meaning	Value	Bit 7	6	5	4	3	2	1	0
0	Response: Write successful	41_{hex}	0	1	0	0	0	0	0	1
1 – 6		00								

9 SDO – Control Commands for easy700

The object dictionary entries Status (3020_{hex}), Command (3021_{hex}) and Response (3022_{hex}) represent the interface for extended data exchange with easy700 on the CANopen communication bus. This allows you to transfer services from the following areas:

- "Read/write date and time" (page 130)
- "Read/write image data" (page 134)
- "Read/write function block data" (page 155).

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

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

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

Read/write date and time

→ Please also note the relevant description of the real-time clock provided in the easy700 manual (AWB2528-1508GB).

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command			
		Read	93	–
		Write	B3	–
	Response			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Len		05	05
2	Index		0 – 2[1]	0 – 2[1]
3 – 7	Data 1 – 5		depending on index, → table 33	

1) 0 = Time/date, → table 33
1 = Summer time, → table 34
2 = Winter time, → table 35

Table 33: Index 0 – date and time of real-time clock

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

Table 34: Index 1 – Summer time

Byte	Contents		Value (hex)
3	Data 1	Area	
		None	00
		Rule	01
		Automatic EU	02
		Automatic GB	03
		Automatic US	04
for "Area" = "Rule":			
4	Data 2	Summer time switching rule	→ table 36
5	Data 3		
6	Data 4		
7	Data 5		

Table 35: Index 2 – Winter time
(only valid if Area = "Rule" selected)

Byte	Contents		Value (hex)
3	Data 1	Area = Rule	01
4 – 7	Data 2 – 5	Winter time switching rule	→ table 36

Switching rule bit array

→ Please also read the detailed description in the easy700 manual (AWB2528-1508GB).

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

Table 36: Switching rule bit array

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

Read/write image data

→ Please also observe the relevant description of possible image data provided in the easy700 manual (AWB2528-1508GB) or in the EASY-SOFT Help. The section "General notes on working with image data" on page 91 also applies to easy700.

Overview

Operands	Meaning	Read/ write	Type (hex)	Page
A1 – A16	"Analog value comparators/threshold comparators: A1 – A16"	Read	8C	135
C1 – C16	"Counters: C1 – C16"	Read	EE	136
D1 – D16	"Text function blocks: D1 – D16"	Read	94	137
I1 – I16	"Local inputs: I1 – I16"	Read	84	138
IA1 – IA4	"Local analog inputs: IA1 – IA4"	Read	8C	139
M1 – M16, N1 – N16	"Markers: M1 – M16/N1 – N16"	Write	86/87	141
M1 – M16, N1 – N16	"Markers: M1 – M16/N1 – N16"	Read	86/87	143
O1 – O4	"Operating hours counters: O1 – O4"	Read	EF	145
P1 – P4	"Local P buttons: P1 – P4"	Read	8A	146
Q1 – Q8	"Local outputs: Q1 – Q8"	Read	85	148
R1 – R16/ S1 – S8	"Inputs/outputs of EASY-LINK: R1 – R16/ S1 – S8"	Read	88/89	149
T1 – T16	"Timing relays: T1 – T16"	Read	ED	151
Y1 – Y4	"Year time switch: Y1 – Y8"	Read	91	152
Z1 – Z3	"Master reset: Z1 – Z3"	Read	93	153
H1 – H4	7-day time switch: ⌚1 – ⌚8	Read	90	154

Analog value comparators/threshold comparators: A1 – A16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	8C	8C
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 37
5	Data 2 (Low Byte)	00	→ table 37
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 37: Byte 4 to 5: Data 1 to 2

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

Counters: C1 – C16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	EE	EE
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 47
5	Data 2 (Low Byte)	00	→ table 47
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 38: Byte 4 to 5: Data 1 to 2

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

Text function blocks: D1 – D16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	94	94
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 39
5	Data 2 (High Byte)	00	→ table 39
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 39: Byte 4 to 5: Data 1 to 2

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

Local inputs: I1 – I16

This command string enables you to read the local inputs of the easy700 basic unit. The relevant input word is stored in Intel format.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1)]
1	Len	02	02
2	Type	84	84
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 40
5	Data 2 (High Byte)	00	→ table 40
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 40: Byte 4 to 5: Data 1 to 2

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

Local analog inputs: IA1 – IA4

The analog inputs on the easy700 basic unit (I7, I8, I11, I12) can be read directly via CANopen. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	02	02
2	Type	8C	8C
3	Index	00 – 03[2]	00 – 03[2]
4	Data 1 (Low Byte)	00	→ table 41
5	Data 2 (High Byte)	00	→ table 41
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

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

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

Table 41: Example telegram for reading the value at the analog input

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

1) 02 = Analog input I11

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

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

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

Write markers: M1 – M16/N1 – N16

Telegram structure

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

1) Possible causes → page 173

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

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

Example:
Marker M13 is set.

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

1) Possible causes → page 173

Read markers: M1 – M16/N1 – N16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type		
	M marker	86	86
	N marker	87	87
3	Index[2]	00	00
4	Data 1 (Low Byte)	00	→ table 42
5	Data 2 (Low Byte)	00	→ table 42
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

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

Table 42: Byte 4 to 5: Data 1 to 2

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

Example:
The N markers are read:

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

1) Possible causes → page 173

The markers N3, N11 and N16 are set.

Operating hours counters: O1 – O4

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	EF	EF
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 43
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 43: Byte 4 : Data 1

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

Local P buttons: P1 – P4

The local P buttons are the display cursor buttons of the easy700 basic unit. You can scan the buttons in both RUN and STOP mode.

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

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	8A	8A
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 44
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 44: Byte 4: Data 1

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

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

Local outputs: Q1 – Q8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 1)
1	Len	01	01
2	Type	85	85
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 45
6 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 45: Byte 4 : Data 1

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

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

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

This service allows you to read the local R and S data and the data of the NET stations (1 – 8) transferred via EASYLINK, again from the relevant easy700 image.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1)]
1	Len	01	01
2	Type		
	for R data	88	88
	for S data	89	89
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 46
5	Data 2 (Low Byte)	00	→ table 46
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 46: Byte 5 to 6: Data 1 to 2

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

Timing relays: T1 – T16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	ED	ED
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 47
5	Data 2 (Low Byte)	00	→ table 47
6 – 7	Data 3 – 4	00	00

1) Possible causes → page 173

Table 47: Byte 4 to 5: Data 1 to 2

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

Year time switch: Y1 – Y8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	91	91
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 48
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 48: Byte 4: Data 1

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

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

Master reset: Z1 – Z3

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	93	93
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 49
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 49: Byte 4: Data 1

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

7-day time switch: Ö1 – Ö8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0[1]
1	Len	01	01
2	Type	90	90
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 50
5 – 7	Data 2 – 4	00	00

1) Possible causes → page 173

Table 50: Byte 4: Data 1

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

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

Read/write function block data

→ Please also observe the relevant description of the function blocks provided in the easy700 manual (AWB2528-1508GB) or in the EASY-SOFT Help.

General notes

Always note the following when working with function blocks:

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

Overview

Operands	Meaning	Read/ write	Type (hex)	Page
A1 – A16	"Analog value comparator/threshold comparator: A1 – A16"	Read/write	8D	156
C1 – C16	"Counter relays: C1 – C16"	Read/write	8F	159
O1 – O4	"Operating hours counters: O1 – O4"	Read/write	92	162
T1 – T16	"Timing relays: T1 – T16"	Read/write	8E	164
Y1 – Y8	"Year time switch: Y1 – Y8"	Read/write	A2	167
⌚1 – ⌚8	7-day time switch: ⌚1 – ⌚8	Read/write	A1	170

Analog value comparator/threshold comparator: A1 – A16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8D	8D
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 51	
4 – 7	Data 1 – 4	depending on index, → table 52	

1) Possible causes → page 173

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

Table 51: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 52		×	
01	Control byte → table 53		×	
02	Comparison value 1	I1 2	×	c 1
03	Comparison value 2	I2 2	×	c 1
04	Gain factor for I1 (I1 = F1 × I1)	F1 2	×	c 1
05	Gain factor for I2 (I2 = F2 × I2)	F2 2	×	c 1
06	Offset for value I1 (I1 = OS + actual value at I1)	OS 2	×	c 1
07	Switching hysteresis for value I2	HY 2	×	c 1

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

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

Table 52: Index 00 – Parameters

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

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

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

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

Table 53: Index 01 – Control byte

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

1) Status 1 if comparison condition is fulfilled.

Counter relays: C1 – C16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8F	8F
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 54	
4 – 7	Data 1 – 4	depending on index, → table 55	

1) Possible causes → page 173

2) easy provides 16 counters C1 to C16 for use as required. These can be addressed using the instance (0 – F).

Table 54: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 55		×	
01	Control byte → table 56		×	
02	Actual value	S1[2]	×	c[1]
03	Counter setpoint 2	S2[2]	×	c[1]

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

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

Table 55: Index 00 – Parameters

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

Example:
Data 1 (Byte 4) = 0x07

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

Table 56: Index 01 – Control byte

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

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

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

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

Explanation:

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

Counter status = 786

Operating hours counters: O1 – O4

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	92	92
2	Instance[2]	00 – 03	00 – 03
3	Index	→ table 57	
4 – 7	Data 1 – 4	depending on index, → table 58	

1) Possible causes → page 173

2) easy provides 4 operating hours counters O1 to O4. These can be addressed using the instance (0 – 3).

Table 57: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 58		×	
01	Control byte → table 59		×	
02	Actual value	S1[2]	×	c[1]
03	Counter setpoint 2	S2[2]	×	c[1]

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

2) A 32-bit value is transferred in data bytes Data 1 – Data 4. It should be remembered that the Data 1 is the low byte and Data 4 the high byte.

Table 58: Index 00 – Parameters

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

Example:
Data 1 (Byte 4) = 0x01

Meaning:
The values appear in the Parameter menu.

Table 59: Index 01 – Control byte

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

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

Example:
Index 02/03

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

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

Timing relays: T1 – T16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	8E	8E
2	Instance[2]	00 – 0F	00 – 0F
3	Index	→ table 60	
4 – 7	Data 1 – 4	depending on index, → table 61	

1) Possible causes → page 173
2) easy provides 16 timing relays T1 to T16 for use as required. These can be addressed using the instance (0 – F).

Table 60: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 61		×	
01	Control byte → table 62		×	
02	Actual value 1	T	×	c[1]
03	Time setpoint 1	S1[2]	×	c[1]
04	Time setpoint 2	S2[2]	×	c[1]

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

Table 61: Index 00 – Parameters

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

Example:
Data 1 (Byte 4) = 0xAC

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

Table 62: Index 01 – Control byte

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

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

Example:
The time setpoint 1 is to be read:

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

Explanation:

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

Meaning depending on set time base:

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

Year time switch: Y1 – Y8

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	A2	A2
2	Instance[2]	00 – 07	00 – 07
3	Index	→ table 63	
4 – 7	Data 1 – 4	depending on index, → table 64	

1) Possible causes → page 173

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

Table 63: Operand overview

Index (hex)	Operand	Read	Write
00	Parameters → table 64	×	
01	Control byte → table 65	×	
	Channel A	×	c[1]
11	Time point ON	×	c[1]
12	Time point OFF	×	c[1]
	Channel B	×	c[1]
21	Time point ON	×	c[1]
22	Time point OFF	×	c[1]
	Channel C	×	c[1]
31	Time point ON	×	c[1]
32	Time point OFF	×	c[1]
	Channel D	×	c[1]
41	Time point ON	×	c[1]
42	Time point OFF	×	c[1]

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

2) The switch points are transferred in data bytes Data 1 – Data 3.

Table 64: Index 00 – Parameters

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

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

Table 65: Index 01 – Control byte

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

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

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

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

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

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

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

7-day time switch: Ö1 – Ö8

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0[1]
1	Type	A1	A1
2	Instance[2]	00 – 07	00 – 07
3	Index	→ table 66	→ table 66
4 – 7	Data 1 – 4	depending on index, → table 67	

1) Possible causes → page 173
2) easy provides 8 7-day time switches Ö1 to Ö8 for use as required. These can be addressed using the instance (0 – 7).

Table 66: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 67		×	
01	Control byte → table 68		×	
11	Channel A	Day on/off	×	c[1]
12		On time	×	c[1]
13		Off time	×	c[1]
21	Channel B	Day on/off	×	c[1]
22		On time	×	c[1]
23		Off time	×	c[1]
31	Channel C	Day on/off	×	c[1]
32		On time	×	c[1]
33		Off time	×	c[1]
41	Channel D	Day on/off	×	c[1]
42		On time	×	c[1]
43		Off time	×	c[1]

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

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

Table 67: Index 00 – Parameters

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

Example:
Data 1 (Byte 4) = 0x03

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

Table 68: Index 01 – Control byte

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

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

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

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

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

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

Analysis – error codes via EASY-LINK

The easy700 basic unit will return a defined error code in the event of an incorrectly selected operating mode or an invalid telegram. The error code transferred has the following structure:

Telegram structure

Byte	Meaning	Slave transmits (value hex)
0	Response	
	Command rejected	C0
1	Type	00
2	Instance	00
3	Index	00
4	Error code	→ table 69

Table 69: Error codes

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

10 SDO – Control Commands for easy800/MFD

The OD entries Status (3020_{hex}), Command (3021_{hex}) and Response (3022_{hex}) provide the interface for extended data exchange with easy800 and MFD on the CANopen communication bus. This allows you to transfer services from the following areas:

- Read/write date and time (page 177)
- Read/write image data (page 182)
- Read/write function block data (page 202).

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

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

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

Version history

The following table provides an overview of modifications and new features of the different easy800 device versions:

Effect on easy-Link	easy800, device version		
	From 02	From 04	From 05
Support for complete PDO access			
R data writable	✓	✓	✓
S data readable	✓	✓	✓
Support for complete SDO access			
Function blocks	–	MR, A, AR, BV, C, CF, CH, CI, CP, D, DB, GT, HW, HY, OT, PT, SC, T, BC, BT, DC, FT, LS, NC, PW, ST, VC	
Image data			
Read	–	IW, IA, ID, QW, QA, P, RW, SW, M, MB, MW, MD	
Write	–	QW, QA, M, MB, MW, MD	M, MB, MW, MD
Clock functions	–	✓	✓
Rule option for winter/summer (DST) time change	–	–	✓

Read/write date and time

→ Please also note the relevant description of the real-time clock provided in the easy800 manual (AWB2528-1423GB).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	05	05
2	Index	00	00
3 – 7	Data 1 – 5		
	Read operation	00	→ table 70
	Write operation	→ table 70	00

Table 70: Byte 3 to 7: Data 1 to 5

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

Winter/summer time, DST

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command			
		Read	93	–
		Write	B3	–
	Response			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Len		05	05
2	Index			
		Summer/winter time	01 → table 71	→ table 71
		Winter time (according to Area = Rule)[1]	02 → table 72	02 → table 72
3 – 7	Data 1 – 5			
		Read operation	00	depending on index, → table 71, 72
		Write operation	depending on index, → table 71, 72	00

1) Detailed setting options for easy800/MFD from version 05

Table 71: Index 01 – Summer / winter time change

Byte	Contents			Value (hex)
3	Data 1	Area		
			None	00
			Manual	01
			Automatic EU	02
			Automatic GB	03
			Automatic US	04
			Rule[1]	05
For Area = Manual				
4	Data 2	Set summer time day 1 to 28, 29, 30, 31 (depending on month and year)		00 – 3B
5	Data 3	Set Summer time month (1 – 12)		01 – 1F
6	Data 4	Set winter time day 1 to 28, 29, 30, 31 (depending on month and year)		01 – 0C
7	Data 5	Set winter time month (1 – 12)		00 – 63
For Area = Rule[1]:				
4 – 7	Data 2 – 5	Summer time switching rule		→ table 73

1) Detailed setting options for easy800/MFD from version 05

Table 72: Index 02 – Winter time
(only valid if Area = Rule selected)

Byte	Contents		Value (hex)
3	Data 1	Area = Rule	01
4 – 7	Data 2 – 5	Winter time switching rule	→ table 73

Switching rule bit array

→ Please also read the detailed description in the easy800 manual (AWB2528-1423GB). The following table shows the composition of the corresponding data bytes.

Table 73: Switching rule bit array

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

Example

The real-time clock of the easy800 is to be set to Friday 23.05.2003, 14:36.

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

→ All values must be transferred as hexadecimal values.

Read/write image data

→ Please also observe the relevant description of possible image data provided in the easy800 manual (AWB2528-1423GB) or in the EASY-SOFT Help. The information provided in section "General notes on working with image data" on page 91 also applies to easy700.

Overview

Operands	Meaning	Read/ write	Command (hex)	Page
IA1 – IA4	"Local analog inputs: IA1 – IA4"	Read	02	183
ID1 – ID16	"Local diagnostics: ID1 – ID16"	Read	03	185
IW0	"Local inputs: IW0"	Read	01	187
IW1 – IW8	"Inputs of the stations: IW1 to IW8"	Read	01	189
M...	"Markers: M.."	Read/write	0B – 0E	190
P1 – P4	"Local P buttons: P1 – P4"	Read	06	193
QA1	"Local analog output: QA1"	Read/write	05	195
QW0, QW1 – QW8	"Local outputs: QW0/ outputs of the stations QW1 – QW8"	Read/write	04	196
R1 – R16 S1 – S8	"Inputs/outputs of EASY-LINK: RW/SW"	Read	07/09	198
RN1 – RN32 SN1 – SN32	"Receive Data Network: RN1 – RN32/ Transmit Data Network: SN1 – SN32"	Read	08/0A	200

Local analog inputs: IA1 – IA4

The analog inputs on the easy800 and MFD basic units can be read directly via CANopen. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

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

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

Example

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

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

1) 01 = Analog input 1

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

Byte 5 – Data 2 (High Byte): 02_{hex}

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

Local diagnostics: ID1 – ID16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	03	03
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 74
5	Data 2 (High Byte)	00	→ table 74
6 – 7	Data 3 – 4	00	00

Table 74: Byte 4 to 5: Data 1 to 2

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

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

Example

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

Local inputs: IW0

This command string enables you to read the local inputs of the easy800/MFD. The relevant input word is stored in Intel format.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	01	01
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 75
5	Data 2 (High Byte)	00	→ table 75
6 – 7	Data 3 – 4	00	00

Table 75: Byte 4 to 5: Data 1 to 2

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

Example: Read local inputs IW0

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

→ All values must be transferred as hexadecimal values.

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

Inputs of the stations: IW1 to IW8

The easy800 and MFD devices can be remotely expanded very simply using the EASYNET. The service offered here makes it possible to implement read access to the inputs of individual NET stations.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	01	01
3	Index	01 – 08[1]	01 – 08[1]
4	Data 1 (Low Byte)	00	→ table 75 on page 187.
5	Data 2 (High Byte)	00	
6 – 7	Data 3 – 4	00	00

1) Corresponds to address of network station

Markers: M..

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command			
		Read	91	–
		Write	B1	–
	Response			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Len		→ table 76	→ table 76
2	Type			
3	Index			
4 – 7	Data 1 – 4			
		Read operation	00	→ "Example 1: setting/resetting a marker bit" on page 192
		Write operation	→ "Example 2: write marker word" on page 192	00

Table 76: Byte 1 to 3: Len, Type, Index

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

If required, refer to the more detailed description of the marker allocation in the easy800 manual. Only a small extract of this manual is shown at this point in order to illustrate the allocation principle.

Attention!
The function blocks and DW markers (32-bit values) of easy800/MFD operate with signed values.

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

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

Example 1: setting/resetting a marker bit

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

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

1) 01 = set, 00 = reset

Example 2: write marker word

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

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

Local P buttons: P1 – P4

The local P buttons are the display cursor buttons of the easy800/MFD basic unit. You can scan the buttons in both RUN and STOP mode.

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

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	06	06
3	Index	00	00
4	Data 1 (Low Byte)	00	→ table 77
5 – 7	Data 2 – 4	00	00

Table 77: Byte 4: Data

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

Local analog output: QA1

The commands provided can be used to access the local analog output of the easy800 or MFD basic unit. When writing to the analog output (only possible with easy800 version 04 or higher), however, the value will only be output externally if the device concerned is in RUN mode and the image concerned has not been overwritten by the actual program. → section "Read/write image data" on page 182.

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

1) Write operations are only possible with easy800 Version 04 or higher, → section "Version history" on page 176.

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

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

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

The local outputs can be read directly via CANopen and also written to with easy800 version 04 or higher. However, the outputs are only switched externally if the device is in RUN mode and the addressed output is not being used in the circuit diagram. → section "Read/write image data" on page 182.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command		
	Read	91	–
	Write[1)]	B1	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Len	02	02
2	Type	04	04
3	Index[2)]	00/01 – 08	00/01 – 08
4	Data 1		
	Read operation	00	→ table 74
	Write operation	→ table 78	00
5 – 7	Data 2 – 4	00	00

1) Write operations are only possible with easy800 Version 04 or higher, → section "Version history" on page 176.

2) 00 = Local output
01 – 08 = Outputs of network stations 1 – 8

Table 78: Byte 4: Data

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

Inputs/outputs of EASY-LINK: RW/SW

This service allows you to read the local R and S data and the data of the NET stations (1 – 8) transferred via EASY-LINK, again from the relevant easy800/MFD image.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	02	02
2	Type	For RW: 07	For RW: 07
		For SW: 09	For SW: 09
3	Index	00/01 – 08[1]	00/01 – 08[1]
4	Data 1 (Low Byte)	00	→ table 79
5	Data 2 (High Byte)	00	→ table 79
6 – 7	Data 3 – 4	00	00

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

Table 79: Byte 4 to 5: Data 1 to 2

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

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

EASYNET allows a point-to-point connection to be implemented between the individual NET stations. The RN and SN data are used for the data exchange (see the easy800 manual).

→ The RN SN data of the local device (Index = 0) to which the EASY221-CO is fitted cannot be scanned. In this case the command would be denied with the $0C_{hex}$ signal.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Len	04	04
2	Type	For RN1 – RN32: 08	
		For SN1 – SN32: 0A	
3	Index	01 – 08[1]	01 – 08[1]
4 – 7	Data 1 – 4	00	→ table 80

1) Corresponds to NET-ID

Table 80: Byte 4 to 7: Data 1 to 4

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

Read/write function block data

→ Please also note the relevant description of the function blocks provided in the easy800 manual.

General notes

Always note the following when working with function blocks:

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

Overview

Operands	Meaning	Read/write	Type(hex)	Page
A01 – A32	"Analog value comparators: A01 – A32"	Read/write	11	204
AR01 – AR32	"Arithmetic function block: AR01 – AR32"	Read/write	12	206
BC01 – BC32	"Block compare: BC01 – BC32"	Read/write	25	208
BT01 – BT32	"Block transfer: BT01 – BT32"	Read/write	26	210
BV01 – BV32	"Boolean operation: BV01 – BV32"	Read/write	13	212
C01 – C32	"Counters: C01 – C32"	Read/write	14	214
CF01 – CF04	"Frequency counters: CF01 – CF04"	Read/write	15	216
CH01 – CH04	"High-speed counters: CH01 – CH04"	Read/write	16	218
CI01 – CI02	"Incremental counters: CI01 – CI02"	Read/write	17	220
CP01 – CP32	"Comparators: CP01 – CP32"	Read/write	18	222
D01 – D32	"Text output function blocks: D01 – D32"	Read/write	19	224
DB01 – DB32	"Data function blocks: DB01 – DB32"	Read/write	1A	227
DC01 – DC32	"PID controllers: DC01 – DC32"	Read/write	27	229
FT01 – FT32	"Signal smoothing filters: FT01 – FT32"	Read/write	28	232
GT01 – GT32	"Receive network data: GT01 – GT32"	Read	1B	234
HW01 – HW32	"7-day time switches: HW01 – HW32"	Read	1C	236
HY01 – HY32	"Year time switches: HY01 – HY32"	Read	1D	239
LS01 – LS32	"Value scaling: LS01 – LS32"	Read/write	29	242
MR01 – MR32	"Master reset: MR01 – MR32"	Read	0F	244
NC01 – NC32	"Numerical converters : NC01 – NC32"	Read/write	2A	246
OT01 – OT04	"Operating hours counters: OT01 – OT04"	Read/write	1E	248
PT01 – PT32	"Transmit network data: PT01 – PT32"	Read	1F	250
PW01 – PW02	"Pulse width modulation: PW01 – PW02"	Read/write	2B	252
SC01	"Synchronize clock: SC01"	Read	20	254
ST01	"Set cycle time: ST01"	Read/write	2C	255
T01 – T32	"Timing relays: T01 – T32"	Read/write	21	257
VC01 – VC32	"Value limitation: VC01 – VC32"	Read/write	2D	260

Analog value comparators: A01 – A32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	11	11
2	Instance	01 – 20	01 – 20
3	Index	→ table 81	→ table 81
4 – 7	Data 1 – 4	00	depending on index, → table 82, 83

Table 81: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 82		×	
01	Mode, → table 83		×	
02	Comparison value 1	I1	×	c[1]
03	Gain factor for I1 (I1 = F1 × value)	F1	×	c[1]
04	Comparison value 2	I2	×	c[1]
05	Gain factor for I2 (I2 = F2 × value)	F2	×	c[1]
06	Offset for value I1	OS	×	c[1]
07	Switching hysteresis for value I2 (the value of HY is for both positive and negative hysteresis.)	HY	×	c[1]

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

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

Table 82: Index 0 – Bit IO

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

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

Table 83: Index 1 - Mode

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

Arithmetic function block: AR01 – AR32

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command:			
		Read	92	–
		Write	B2	–
	Response:			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Type		12	12
2	Instance		01 – 20	01 – 20
3	Index		→ table 84	→ table 84
4 – 7	Data 1 – 4			
		Read operation	00	depending on index, → table 85, 86
		Write operation	depending on index, → table 85, 86	00

Table 84: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 85		×	
01	Mode, → table 86		×	
02	First operand	I1	×	c[1]
03	Second operand	I2	×	c[1]
04	Result	QV	×	

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

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

Table 85: Index 0 – Bit IO

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

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

Table 86: Index 1 - Mode

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

Block compare: BC01 – BC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	25	25
2	Instance	01 – 20	01 – 20
3	Index	→ table 87	→ table 87
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 88, 89
	Write operation	depending on index, → table 88, 89	00

Table 87: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 88		×	
01	Mode, → table 89		×	
02	Source range 1	I1	×	c[1]
03	Target range 2	I2	×	c[1]
04	Number of elements to compare: 8 (max. 192 bytes)	NO	×	c[1]

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

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

Table 88: Index 0 – Bit IO

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

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

Error outputs

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

Table 89: Index 1 - Mode

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

Block transfer: BT01 – BT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	26	26
2	Instance	01 – 20	01 – 20
3	Index	→ table 90	→ table 90
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 91, 92
	Write operation	depending on index, → table 91, 92	00

Table 90: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 91		×	
01	Mode, → table 92		×	
02	Source range 1	I1	×	c[1]
03	Target range 2	I2	×	c[1]
04	Number of elements to compare: max. 192 bytes	NO	×	c[1]

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

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

Table 91: Index 0 – Bit IO

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

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

Error outputs

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

Table 92: Index 1 - Mode

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

Boolean operation: BV01 – BV32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	13	13
2	Instance	01 – 20	01 – 20
3	Index	→ table 93	→ table 93
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 94, 95
	Write operation	depending on index, → table 94, 95	00

Table 93: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 94		×	
01	Mode, → table 95		×	
02	First operand	I1	×	c[1]
03	Second operand	I2	×	c[1]
04	Operation result	QV	×	

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

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

Table 94: Index 0 – Bit IO

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

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

Table 95: Index 1 - Mode

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

Counters: C01 – C32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	14	14
2	Instance	01 – 20	01 – 20
3	Index	→ table 96	→ table 96
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 97
	Write operation	depending on index, → table 97	00

Table 96: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 97	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2 147 483 648 to +2 147 483 647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

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

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

Table 97: Index 0 – Bit IO

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

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

Frequency counters: CF01 – CF04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	15	15
3	Instance	01 – 04	01 – 04
4	Index	→ table 98	→ table 98
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 99
	Write operation	depending on index, → table 99	00

Table 98: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 99		×	
01	Mode/Parameters		–	–
02	Upper setpoint	SH	×	c[1]
03	Lower setpoint	SL	×	c[1]
04	Actual value in RUN mode	QV	×	

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

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

Table 99: Index 0 – Bit IO

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

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

High-speed counters: CH01 – CH04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	16	16
2	Instance	01 – 04	01 – 04
3	Index	→ table 100	→ table 100
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 101
	Write operation	depending on index, → table 101	00

Table 100: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 101	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2 147 483 648 to +2 147 483 647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

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

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

Table 101: Index 0 – Bit IO

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

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

Incremental counters: CI01 – CI02

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	17	17
2	Instance	01 – 02	01 – 02
3	Index	→ table 102	→ table 102
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 103
	Write operation	depending on index, → table 103	00

Table 102: Operand overview

Index (hex)	Operand		Value	Read	Write
00	Bit IO		→ table 103	×	
01	Mode/Parameters		–	–	–
02	Upper setpoint	SH	In integer range from –2147483648 to +2147483647	×	c[1]
03	Lower setpoint	SL		×	c[1]
04	Preset actual value	SV		×	c[1]
05	Actual value in RUN mode	QV		×	

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

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

Table 103: Index 0 – Bit IO

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

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

Comparators: CP01 – CP32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	18	18
2	Instance	01 – 20	01 – 20
3	Index	→ table 104	→ table 104
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 105
	Write operation	depending on index, → table 105	00

Table 104: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 105		×	
01	Mode/Parameters		–	–
02	Comparison value	I1	×	c[1]
03	Comparison value	I2	×	c[1]

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

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

Table 105: Index 0 – Bit IO

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

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

Text output function blocks: D01 – D32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	19	19
2	Instance	01 – 20	01 – 20
3	Index	→ table 106	→ table 106
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 107
	Write operation	depending on index, → table 107	00

Table 106: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 107	×	
01	Mode/Parameters	–	–
02	Text line 1, column 1 - 4	×	
03	Text line 1, column 5 - 8	×	
04	Text line 1, column 9 - 12	×	
05	Text line 1, column 13 - 16	×	
06	Text line 2, column 1 - 4	×	
07	Text line 2, column 5 - 8	×	
08	Text line 2, column 9 - 12	×	
09	Text line 2, column 13 - 16	×	
10	Text line 3, column 1 - 4	×	
11	Text line 3, column 5 - 8	×	
12	Text line 3, column 9 - 12	×	
13	Text line 3, column 13 - 16	×	
14	Text line 4, column 1 - 4	×	
15	Text line 4, column 5 - 8	×	
16	Text line 4, column 9 - 12	×	
17	Text line 4, column 13 - 16	×	
18	Variable 1	×	c[1]
19	Variable 2	×	c[1]
20	Variable 3	×	c[1]
21	Variable 4	×	c[1]
22	Scaling minimum value 1	×	
23	Scaling minimum value 2	×	
24	Scaling minimum value 3	×	
25	Scaling minimum value 4	×	
26	Scaling maximum value 1	×	

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

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

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

Table 107: Index 0 – Bit IO

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

1) Text function block enable

2) Status 1, text function block is active

Data function blocks: DB01 – DB32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	1A	1A
2	Instance	01 – 20	01 – 20
3	Index	→ table 108	→ table 108
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 109
	Write operation	depending on index, → table 109	00

Table 108: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 109		×	
01	Mode/Parameters		–	–
02	Input value: value that is transferred to the QV output when the FB is triggered.	I1	×	c[1]
03	Output value	QV	×	

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

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

Table 109: Index 0 – Bit IO

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

1) Transfer of the value present at I1 on rising edge.

2) Status 1 if the trigger signal is 1.

PID controllers: DC01 – DC32

Telegram structure

Byte	Meaning		Value (hex), sent by	
			Master	Slave
0	Command:			
		Read	92	–
		Write	B2	–
	Response:			
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
1	Type		27	27
2	Instance		01 – 20	01 – 20
3	Index		→ table 110	→ table 110
4 – 7	Data 1 – 4			
		Read operation	00	depending on index, → table 111, 112
		Write operation	depending on index, → table 111, 112	

Table 110: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 111		×	
01	Mode, → table 112		×	
02	Setpoint: –32 768 to +32 767	I1	×	c[1]
03	Actual value: –32 768 to +32 767	I2	×	c[1]
04	Proportional gain [%], Value range: 0 to 65 535	KP	×	c[1]
05	Reset time [0.1 s], Value range: 0 to 65 535	TN	×	c[1]
06	Rate time [0.1 s], Value range: 0 to 65 535	TV	×	c[1]
07	Scan time = Time between function block calls. Value range: 0.1s to 6553.5s. If 0 is entered as the value, the scan time will be determined by the program cycle time.	TC	×	c[1]
08	Manual manipulated variable, value range: –4096 to +4095	MV	×	c[1]
09	Manipulated variable • Mode: UNI, value range: 0 to +4095 (12 bit) • Mode: BIP, value range: –4096 to +4095 (13 bit)	QV	×	

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

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

Table 111: Index 0 – Bit IO

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

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

Table 112: Index 1 - Mode

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

Signal smoothing filters: FT01 – FT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	28	28
2	Instance	01 – 20	01 – 20
3	Index	→ table 113	→ table 113
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 114
	Write operation	depending on index, → table 114	00

Table 113: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 114		×	
01	Mode/Parameters		–	–
02	Input value, value range: –32768 to +32767	I1	×	c[1]
03	Recovery time [0.1 s], Value range: 0 to 65535	TG	×	c[1]
04	Proportional gain [%], Value range: 0 to 65535	KP	×	c[1]
05	Delayed output value, value range: –32768 to +32767	QV	×	

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

Table 114: Index 0 – Bit IO

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

1) Activates the function block on status 1.

Receive network data: GT01 – GT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1B	1B
2	Instance	01 – 20	01 – 20
3	Index	→ table 115	
4 – 7	Data 1 – 4	00	depending on index, → table 116, 117

Table 115: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 116		×	
01	Mode/Parameters, → table 117		×	–
02	Output value: actual value from the network	QV	×	

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

Table 116: Index 0 – Bit IO

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

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

Table 117: Index 1 – Mode/Parameters (designation of PUT FB with data to be received)

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

1) Number of station transmitting the value. Possible station numbers: 01 to 08

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

7-day time switches: HW01 – HW32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1C	1C
2	Instance	01 – 20	01 – 20
3	Index	→ table 118	
4 – 7	Data 1 – 4	00	depending on index, → table 119

Table 118: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 119	×	
01	Mode/Parameters	–	–
02	Parameters → table 120 Channel A	×	
03	Channel B		
04	Channel C		
05	Channel D		

Table 119: Index 0 – Bit IO

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

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

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

Table 120: Index 2 to 5, Parameter channels A to D

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

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

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

Example

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

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

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

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

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

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

Year time switches: HY01 – HY32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1D	1D
2	Instance	01 – 20	01 – 20
3	Index	→ table 121	
4 – 7	Data 1 – 4	00	depending on index, → table 122

Table 121: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 122	×	
01	Mode/Parameters	–	–
02	Parameters → table 123 Channel A	×	
03	Channel B		
04	Channel C		
05	Channel D		

Table 122: Index 0 – Bit IO

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

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

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

Table 123: Index 2 to 5, parameter channels A to D

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

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

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

Example

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

Index 2 – 5, Parameter channels A – D

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

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

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

Index 2 – 5, Parameter channels A – D

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

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

Resulting telegram:

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

Value scaling: LS01 – LS32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	29	29
2	Instance	01 – 20	01 – 20
3	Index	→ table 124	→ table 124
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 125
	Write operation	depending on index, → table 125	

Table 124: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 125		×	
01	Mode/Parameters		–	–
02	Input value, value range: 32 bit	I1	×	c[1]
03	Interpolation point 1, X coordinate, value range: 32 bit	X1	×	c[1]
04	Interpolation point 1, Y coordinate, value range: 32 bit	Y1	×	c[1]
05	Interpolation point 2, X coordinate, value range: 32 bit	X2	×	c[1]
06	Interpolation point 2, Y coordinate, value range: 32 bit	Y2	×	c[1]
07	Output value: contains the scaled input value	QV	×	

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

Table 125: Index 0 – Bit IO

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

1) Activates the function block on status 1.

Master reset: MR01 – MR32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	0F	0F
2	Instance	01 – 20	01 – 20
3	Index		
	Bit IO	00	00
	Mode	01	01
4 – 7	Data 1 – 4	00	depending on index, → table 126, 127

Table 126: Index 0 – Bit IO

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

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

Table 127: Index 1 - Mode

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

Numerical converters : NC01 – NC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2A	2A
2	Instance	01 – 20	01 – 20
3	Index	→ table 128	→ table 128
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 129, 130
	Write operation	depending on index, → table 129, 130	00

Table 128: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 129	×	
01	Mode, → table 130	×	
02	Input value: I1 operand to be converted	×	c[1]
03	Output value: QV contains the conversion result	×	

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

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

Table 129: Index 0 – Bit IO

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

1) Activates the function block on status 1.

Table 130: Index 1 - Mode

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

Operating hours counters: OT01 – OT04

→ Further information is available in the S40 Application Note AN27K21g.exe “EASY800/MFD-DP Data Handling Function Block for PS416 and PS4-341”.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	1E	1E
2	Instance	01 – 04	01 – 04
3	Index	→ table 131	→ table 131
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 132
	Write operation	depending on index, → table 132	00

Table 131: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 132		×	
01	Mode/Parameters		–	–
02	Upper threshold value	I1	×	c[1]
03	Actual value of operating hours counter	QV	×	

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

Table 132: Index 0 – Bit IO

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

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

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

Transmit network data: PT01 – PT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	1F	1F
2	Instance	01 – 20	01 – 20
3	Index	→ table 133	
4 – 7	Data 1 – 4	00	depending on index, → table 134

Table 133: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 134	×	
01	Mode/Parameters	–	–
02	Input value: Setpoint I1 that it transmitted to the NET network	×	

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

Table 134: Index 0 – Bit IO

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

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

Pulse width modulation: PW01 – PW02

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2B	2B
2	Instance	01 – 02	01 – 02
3	Index	→ table 135	→ table 135
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 136
	Write operation	depending on index, → table 136	00

Table 135: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 136		×	
01	Mode/Parameters		–	–
02	Manipulated variable, value range: 0 to 4095 (12 bit)	SV	×	c[1]
03	Period duration [ms], Value range: 0 to 65535	PD	×	c[1]
04	Minimum on duration [ms], Value range: 0 to 65535	ME	×	c[1]

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

Table 136: Index 0 – Bit IO

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

1) Activates the function block on status 1.

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

Synchronize clock: SC01

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
1	Type	20	20
2	Instance	01	01
3	Index	→ table 137	
4 – 7	Data 1 – 4	00	depending on index, → table 138

Table 137: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 138	×	
01	Mode/Parameters	–	–

Table 138: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T1)
FB output Data 3		–	–	–	–	–	–	–	Q12)

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

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

Set cycle time: ST01

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2C	2C
2	Instance	01	01
3	Index	→ table 139	→ table 139
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 140
	Write operation	depending on index, → table 140	00

Table 139: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 140		×	
01	Mode/Parameters		–	–
02	Cycle time in ms, value range: 0 – 1000	I1	×	c[1]

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

Table 140: Index 0 – Bit IO

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

1) Activates the function block on status 1.

Timing relays: T01 – T32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	21	21
2	Instance	01 – 20	01 – 20
3	Index	→ table 141	→ table 141
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 142, 143
	Write operation	depending on index, → table 142, 143	

Table 141: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 142		×	
01	Mode/Parameters, → table 143		×	
02	Setpoint 1: Time setpoint 1	I1	×	c[1]
03	Setpoint 2: Time setpoint 2 (with timing relay with 2 setpoints)	I2	×	c[1]
04	Actual value: Time elapsed in RUN mode	QV	×	

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

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

Table 142: Index 0 – Bit IO

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

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

Table 143: Index 1 - Mode/Parameters

Mode	**Data 1**	**Operating mode**
	0	On-delayed
	1	On-delayed with random setpoint
	2	Off-delayed
	3	Off-delayed with random setpoint
	4	On and off delayed (two time setpoints)
	5	On and off delayed each with random setpoint (two time setpoints)
	6	Impulse transmitters
	7	Flashing relay (two time setpoints)
	8	Off-delayed, retriggerable (easy600 Mode)
	9	Off-delayed with random setpoint, retriggerable (easy600 Mode)
Para-meters	**Data 3**	**Operating mode**
	0	S (milliseconds)
	1	M:S (seconds)
	2	H:M (minutes)

Value limitation: VC01 – VC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
1	Type	2D	2D
2	Instance	01 – 20	01 – 20
3	Index	→ table 144	→ table 144
4 – 7	Data 1 – 4		
	Read operation	00	depending on index, → table 145
	Write operation	depending on index, → table 145	00

Table 144: Operand overview

Index (hex)	Operand		Read	Write
00	Bit IO, → table 145		×	
01	Mode/Parameters		–	–
02	Input value	I1	×	c[1]
03	Upper limit value	SH	×	c[1]
04	Lower limit value	SL	×	c[1]
05	Output value: outputs the value present at input I1 within the set limits.	QV	×	

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

Table 145: Index 0 – Bit IO

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

1) Activates the function block on status 1.

Analysis – error codes via EASY-LINK

The easy800/MFD basic unit will return a defined error code in the event of an incorrectly selected operating mode or an invalid telegram. The error code transferred has the following structure:

Telegram structure

Byte	Meaning	Slave transmits (value hex)
0	Response	
	Command rejected	C0
1	Type	
2	Instance	
3	Index	
4	Error code	→ table 146
5 – 7	Data 2 – 4	

Table 146: Error codes

Error code	Description
0x00	No error
0x03	Formal error in the response related to type, instance or index
0x04	No communication possible (Timeout)
0x05	DP module has only transmitted 0xC0 (Easy800 Basic II, MFD Version I).
0x45	The value selected by Type and Index must not be overwritten (Bit IO, Mode/Parameters or output value).
0x46	The value selected by Type and Index is not assigned to a constant and cannot therefore be written.
0x9E	Access to the FB data not possible (program download active).
0x9F	Type is invalid (no defined FB, depending also on the version of the addressed device).
0xA0	FB selected by Type and Index does not exist in the program.
0xA1	Index related to the specified FB type is invalid.

11 What happens if...?

RUN LED

Status of the RUN LED	Possible cause	To correct or avoid error
OFF	The EASY221-CO is either switched off or is currently being reset.	Switch on the EASY221-CO and supply with mains voltage.
Flickering	Auto baud recognition is currently busy (LED flickers, alternating with the ERR LED).	Check the communication of the master PLC or the bus.
Single flash	The device is in STOPPED state.	Change the status of NMT (network management), see Section 4.3
Flashing	The device is in PRE-OPERATIONAL state.	
ON	The device is in OPERATIONAL state.	

Error LED

Status of the error LED	Possible cause	To correct or avoid error
OFF	The EASY221-CO is operating error-free. If the RUN LED is also off, 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 rate recognition is currently busy (flickers alternating with the RUN LED).	Check the communication of the master PLC or the bus.
Flashes twice	A protective Guard Event or a Heartbeat Event has occurred.	Check configuration data.
ON	The CANopen PLC has changed to BUS-OFF state.	Verify the correct setting of the NODE ID.

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
Vibration (IEC 60068-2-6)		
constant amplitude 0.15 mm	Hz	10 to 57
Constant acceleration 2 g	Hz	57 to 150
Shocks (IEC 60068-2-27) semi-sinusoidal 15 g/11 ms	Shocks	18
Drop (IEC 60068-2-31) height	mm	50
Free fall, when packed (IEC 60068-2-32)	m	1

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

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

Dimensions

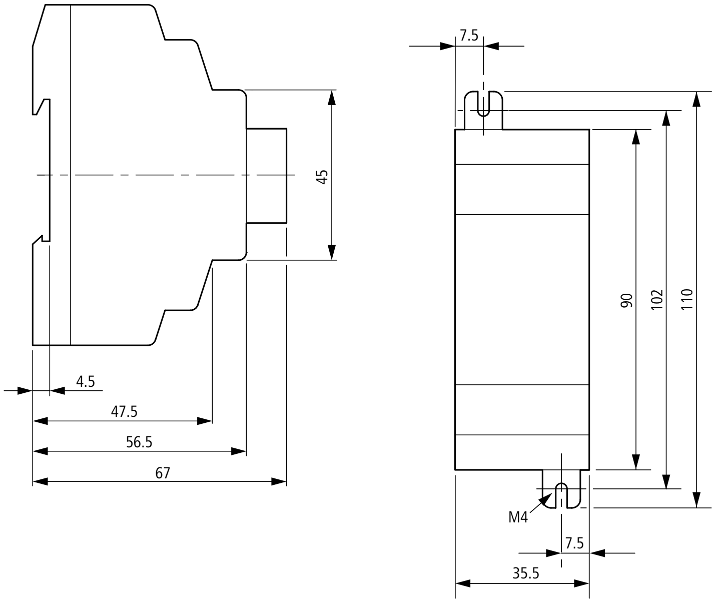


Figure 26: EASY221-CO dimensions in [mm]

Glossary

This glossary refers to the topics related to CANopen.

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

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

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

Const	Constant object. The value is read-only and does not change during runtime. Example: Device Software Version.
CPU	Abbreviation for "Central Processing Unit". Central unit for data processing, which represents the core element of a computer.
CRC	Cyclic Redundancy Check: CAN data integrity check routine with low residual error probability. Also used in other areas of data transfer.
CSA certification	Canadian certification (Canadian Standards Association)
CSMA	Carrier Sense Multiple Access. Bus access routine used in CAN. Each node can independently access the bus as soon as the bus is free.
Data Frame	CAN message frame used by a transmitter to broadcast data to several receivers.
Data request telegram	CAN remote transmission request frame, which a network node transmits to another node.
DBT	Distributor. One of the services of the application layer in the CAN Reference Model. Used for the configuration of layers in the CAN Reference Model. The assignment of COB-IDs to the COBs used by CMS represents a DBT task.
DBT master	Special CAN node. Its task is to assign and manage the COB-IDs in a CAL or CANopen network.
DBT slave	All CAN nodes assigned a COB-ID by the DBT master.
Device Profile	here: CANopen Device Profile. Described in CiA Draft Standards CiA-DS-401 ff.
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.
Earthing strip	Flexible conductor, mostly braided. Interconnects inactive parts of equipment, e.g. the doors of a control panel and the switch cabinet body.
EDS	Electronic Data Sheet: File containing device-specific parameter definitions (provided by the manufacturer of CANopen or DeviceNet devices)
EEPROM	Abbreviation for "Electrically Erasable Programmable Read-only .
EIA	Abbreviation for "Electronic Industries , USA.
Electrical equipment	Comprises all equipment used for the generation, conversion, transfer, distribution and application of electrical energy, e.g. power lines, cables, machines, controllers.
EMC	Abbreviation for "Electromagnetic Compatibility". The ability of electrical equipment to function trouble-free within a particular environment without a negative effect on the environment concerned.
EN	Abbreviation for "European Norm".
Equipotential bonding	Adaptation of the electrical level of the body of electrical equipment and auxiliary conductive bodies by means of an electrical connection.
ESD	Abbreviation for "Electrostatic Discharge".
Fault Mode	Determines the mode of reaction to errors. When this bit is set for an output, this output will be set to the value declared in its fault state parameter.
Field supply	Voltage supply to field devices as well as signal voltage.
Fieldbus	Data network on the sensor/actuator level. The fieldbus interconnects the devices at field level. Characteristic feature of the fieldbus is their highly reliable transfer of signals and real-time response.

Galvanic coupling	A galvanic coupling occurs when two circuits use the same cable. Typical sources of interference are, for example, starting motors, static discharges, clocked devices, and a potential difference between the housing of components and the common power supply.
GND	Abbreviation for "GROUND" (0 potential).
Ground	In electrical engineering the name for conductive grounding with an electrical potential at any point equal to zero. In the environment of grounding devices, the electrical ground potential may not equal zero. This is called a "reference ground".
ground (verb)	Represents the connection of an electrically conductive component to the equipotential earth via a grounding device.
Grounding device	One or several components that have a direct and good contact with the ground.
Guard Identifier	Guarding protocol identifier used for node monitoring. The NMT master here transmits an RTR to the monitored slaves, requesting it to return its current status.
Guard Time	Node monitoring time. Configurable time 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	Number system with base 16. Counting from 0 to 9 and then with the letters A, B, C, D, E and F.
I/O	Abbreviation for "Input/Output".
Identifier	Frame identifier. Standard CAN uses 11-bit, Extended CAN 29-bit identifiers.
Impedance	Apparent resistance that a component or circuit of several components has for an alternating current at a particular frequency.

Inactive metal parts	Conductive parts that cannot be touched and which are insulated from active metal parts. They can, however, carry voltage in the event of a fault.
Index	The index (in arrays and records) and the subindex specify an object address that conforms with CANopen standard. This address represents an index in the object dictionary. Only an index is output for simple variables. Array structures have subindexes which are appended comma-separated to the index. Example: [1800,01] = index 1800, subindex 1.
Inductive coupling	Inductive (magnetic) coupling occurs between two current carrying conductors. The magnetism produced by the currents induces an interference voltage. Typical interference sources are, for example transformers, motors, mains cables installed parallel and RF signal cables.
Inhibit Time	Time interval during which a PDO may not be transmitted again, in order to avoid excess load on the network.
Life Time	Life Time/node monitoring time. Configurable time 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. Described in the CiA Draft Standard CiA-DS-205. It contains the so-called layer-specific management functions. These include in particular the module name and ID as well as the timing parameters of the physical transmission layer, i.e. the baud rate of the CAN nodes.
LMT master	In the LMT model, this CAN node is assigned the task of configuring the LMT parameters of the other CAN nodes.
LMT slave	CAN node that that communicates in the LMT model with the LMT master in a master/slave model.
Low impedance connection	Connection with low alternating-current resistance.
LSB	Abbreviation for “Least Significant Bit”. Bit with the lowest value.

Mapping	All connection data, i.e. the assignment of network variables to PDOs. A PDO can transmit one or multiple network variables (see CiA DS-301). The assignment of variables to PDOs is defined in the Mapping tables. These can be addressed via the object dictionary.
Master	Station or node in a bus system that controls communication between the other stations of the bus system.
Master-slave mode	Operating mode in which a station or node of the system acts as master that controls communication on the bus.
Mode	Operating mode.
Module bus	Represents the internal bus of an XI/ON station. Used by the XI/ON modules for communication with the gateway. Independent of the fieldbus.
MSB	Abbreviation for "Most Significant Bit". Bit with the most significant value.
Multimaster Mode	Operating mode in which all stations or nodes of a system have equal rights for communicating on the bus.
Namur	Abbreviation for "Normen-Arbeitsgemeinschaft für Mess- und Regeltechnik" (Standards Committee for Measurement and Control Technology). 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 electronic memory for electronic counters and for data backup during power loss.

Object Dictionary	Object dictionary. The object dictionary contains all objects accessible via the network in a defined sequence. These objects are accessed via a 16-bit index.
Operational	Active status of a CANopen node. In this state the node can transmit and receive PDOs, depending on the type and configuration. SDO communication is still possible.
Overhead	System management time required in the system in each transmission cycle.
Parameter assignment	Definition of parameters for individual bus slaves or their modules in the configuration software of the 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" from this state.
Priorities	The CAN frame identifiers also determine the priorities for bus access. This allows fast bus access according to the significance of messages.
Protected against short-circuit	Property of electrical equipment. Short-circuit-proof equipment has the ability to withstand the thermal and dynamic loads that may occur at the location of installation on account of a short-circuit.
Protective conductor	A conductor required for the protection against dangerous currents, designated by the letters PE (abbreviation of "Protective Earth").

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

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

Index

D

E

O

P