User Manual

09/10 MN05013007Z-EN

replaces 10/04 AWB2528-1427GB

EASY222-DN
DeviceNet Slave Interface

Powering Business Worldwide

Emergency On Call Service
Please call your local representative:
Eaton.com/aftersales
or
Hotline After Sales Service:
+49 (0) 180 5 223822 (de, en)
AfterSalesEGBonn@eaton.com

Original Operating Instructions
The German-language edition of this document is the original operating manual.

Translation of the original operating manual
All editions of this document other than those in German language are translations of the original German manual.

1st published 2002, edition date 08/02
2nd edition 2004, edition date 10/04
3rd edition 2008, edition date 02/08
4th edition 2010, edition date 09/10
See revision protocol in the "About this manual" chapter

Production: Thomas Kracht, Barbara Petrick
Translation: Terence Osborn

Danger!
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 significant amendments have been introduced since the previous issue:

Publication date	Page	Key word	New	Change	omitted
10/04	All	easy700/800/MFD	✓		
02/08	chapter8	DeviceNet access to the modified/new function blocks of the easy800 MFD-CP8/CP10 from operating system V 1.20	✓		
09/10	All	Changeover to Eaton designations	✓		

Target group

This manual is intended for automation technicians and engineers. Expert knowledge of the DeviceNet fieldbus and programming of a DeviceNet master PLC is assumed. Furthermore, you should be familiar with the handling of the easy control relay and the MFD HMI control.

Additional device manuals

The following manuals apply:

- "easy412, easy600 control relays" (MN05013004Z-EN; previous description AWB2528-1304-GB)
- "easy700 control relays" (MN05013003Z-EN; previous description AWB2528-1508GB)
- "easy800 control relays" (MN04902001Z-EN; previous description AWB2528-1423GB)
- "MFD-Titan multi-function display" (MN05002001Z-EN; previous description AWB2528-1480GB).

All manuals are available on the Internet for download as PDF files. They can be quickly located at http://www.eaton.com/moeller → Support by entering the "EASY222-DN" as the search term.

References

[1] DeviceNet Specification Volume I
Release 2.0, Errata 1 - 4
April 1, 2001

[2] DeviceNet Specification Volume II
Release 2.0, Errata 1 - 4
April 1, 2001

Device designation

The following short names for equipment types are used in this manual, as far as the description applies to all of these types:

- 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-...
- easy-AB for
  - EASY719-AB...
- easy-AC for
  - EASY6…-AC-RC(X)
  - EASY719AC
  - EASY8..-AC-...

- easy-DC for
  - easy6…-DC-…
  - EASY719-DC-…
  - EASY8…-.DC-…

- easy-DA for
  - EASY719-DA...

- MFD-CP8… for
  - MFD-CP8-ME
  - MFD-CP8-NT
  - MFD-AC-CP8-ME
  - MFD-AC-CP8-NT

- MFD-CP10… for
  - MFD-CP10-ME
  - MFD-CP10-NT

- MFD-…-CP8/CP10 for
  - MFD-CP8-ME
  - MFD-CP8-NT
  - MFD-AC-CP8-ME
  - MFD-AC-CP8-NT
  - MFD-CP10-ME
  - MFD-CP10-NT

Abbreviations and symbols

Meaning of abbreviations and symbols used in this manual:

bcd	**B**inary **C**oded **D**ecimal code
CAN	**C**ontroller **A**rea **N**etwork
dec	Decimal (number system based on 10)
hex	Hexadecimal (Number system based on 16)
len	**Len**gth
MAC ID	**M**edia **A**ccess **C**ontrol **Id**entifier
ODVA	**O**pen DeviceNet **V**endor **A**ssociation
PC	**P**ersonal **C**omputer
SELV	**S**afety **E**xtra **L**ow **V**oltage“
UCMM	**U**n**c**onnected **M**essage **M**anager

Writing Conventions

For greater clarity, the name of the current chapter is shown in the header of the left-hand page and the name of the current section in the header of the right-hand page. This does not apply to pages at the start of a chapter and empty pages at the end of a chapter.

▶ indicates actions to be taken.

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

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

Danger!
warns of the possibility of serious damage and slight injury or death.

→ Draws your attention to interesting tips and supplementary information.

1 The EASY222-DN

The EASY222-DN communication module has been developed for automation tasks with the DeviceNet field bus. EASY222-DN acts as a “gateway” and can only be operated in conjunction with the expanded easy600, easy700, easy800 or MFD basic units. The system unit consists of the easy/MFD control device and the EASY222-DN DeviceNet gateway and operates exclusively as a slave station on the fieldbus system.

System overview

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

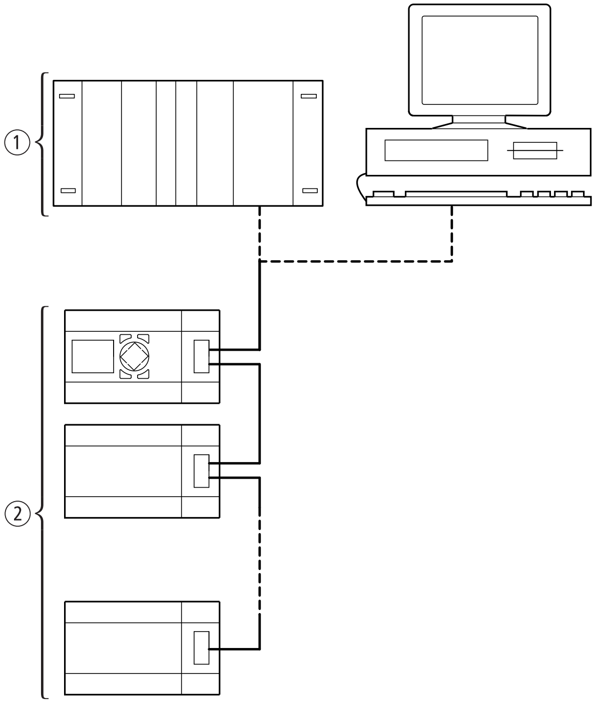


Figure 1: Implementation of EASY222-DN in the DeviceNet

1 Master area, PLC (e.g.: SLC 500) or PC with CAN card

2 Slave area, e.g.: Control relay easy/MFD with DeviceNet interface

Structure of the unit

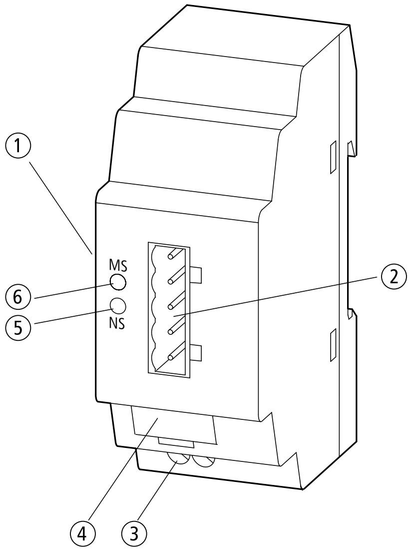


Figure 2: Surface Mounting EASY222-DN

① easyLink socket

② 5-pin DeviceNet connection to ODVA

③ Power supply 24 V ⎓

④ Device label

⑤ Network Status LED NS

⑥ Module Status LED MS

EASY222-DN Communication profile

- Predefined master/slave communication settings
  - The **I/O polling** connection is used for the transfer of 3 bytes of input data (R1 to R16) and 3 bytes of output data (S1 to S8) between the easy base unit with gateway interconnection and the DeviceNet PLC.
  - The **I/O Change of State/Cyclic** connection (acknowledged, unacknowledged) is used to transfer 2 bytes of diagnostic data from the easy control relay to DeviceNet the PLC.
  - The **explicit connection set-up** is used for read/write access to function relay parameters in the easy control relay. This type of connection set-up also supports the configuration, diagnostics and management services of the control relay.
- DeviceNet Communication adapter profile (device type 12), which has been expanded by easy requests
- Group 2 server
- UCMM-capable device
- Dynamic set-up of explicit and I/O connections are possible
- Device Heartbeat Message
- Device Shutdown Message
- Offline communication settings

Hardware and operating system requirements

The EASY222-DN expansion unit operates together with the easy600, easy700, easy800 and MFD basic units from the following operating systems:

Basic unit		**EASY222-DN expansion unit**	
Device version	OS version	Device version = 01	Device version ≧ 02
easy600			
≧ 04	from 2.4	×	×
easy700			
≧ 01	from 1.01.xxx	–	×
easy800			
≧ 04	from 1.10.xxx	–	×
MFD-CP8…/CP10…			
≧ 01	from 1.10.xxx	–	×
MFD-CP10			
≧ 01	From 1.00	–	×

The device version of the respective basic or expansion unit is stated on the right-hand side of the enclosure. Example: EASY222-DN: 02-206xxxxxxx (02 = device version)
The operating system version (OS) of the respective basic device can be read via the easySoft. On the easy700, easy800 and MFD-CP8.. devices it is possible to read out the information directly on the device. Refer to the respective manual for information.
An overview of the modifications and innovations with the different device versions of the easy800 can be found on page 147.

Improper use

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

- burner,
- Emergency switching off,
- crane controls or
- two-hand safety controls.

2 Installation

Applicable are the same guidelines as for easy/MFD basic units with expansion modules.

EASY222-DN connection to the basic unit

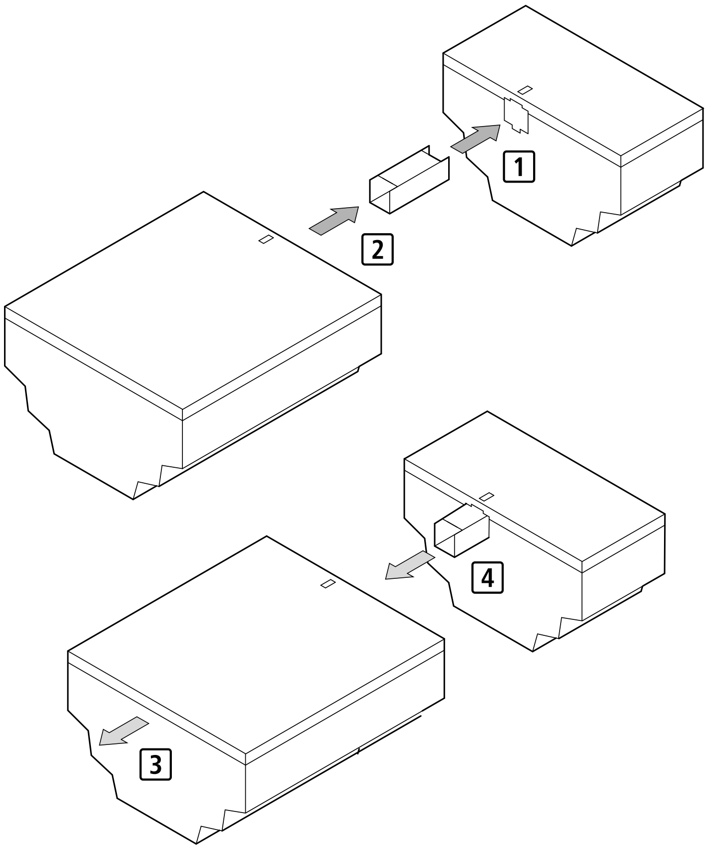


Figure 3: Mounting the EASY222-DN on the basic unit

1 + 2 Installation

3 + 4 Removal

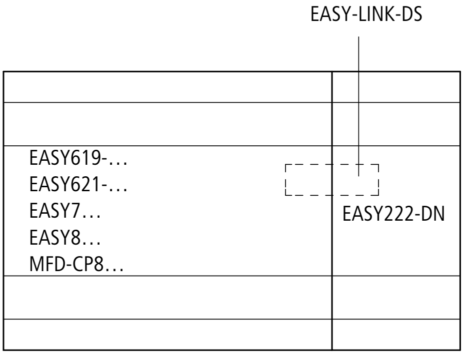
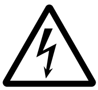


Figure 4: Connection between basic unit and EASY222-DN

Connecting the power supply

EASY222-DN operates with a 24 V DC supply voltage (→ section "Current supply", page 257).



Danger!
Ensure a reliable electrical isolation of the low voltage (SELV) for the 24 V supply.

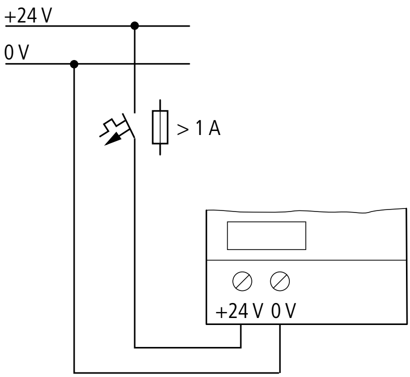


Figure 5: Supply voltage EASY222-DN

Connecting DeviceNet

A 5 pole DeviceNet plug connects the DeviceNet interface of the device to the DeviceNet field bus.

Please use a special DeviceNet plug and DeviceNet cable for this connection. Both are specified in the ODVA. The type of cable has an influence on the maximum available length of the bus line and thus on the data transfer rate.

Terminal assignment DeviceNet

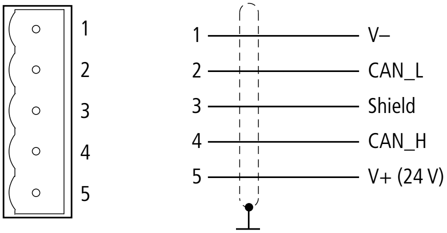


Figure 6: Pin assignment of the equipment socket

1	GND	black
2	CAN_L	blue
3	screen	clear
4	CAN_H	white
5	24 V	red

All pins of the plug must be connected to ensure safe communication of the EASY222-DN on the fieldbus DeviceNet. This also applies to the 24-V bus voltage.

→ The gateway therefore does not participate in communication on the bus if the bus voltage is not available.
The Network status LED indicates OFF mode in this situation.

Terminal resistors

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

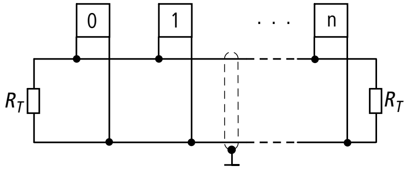


Figure 7: Terminating resistors R_T: CAN_H and CAN_L terminals

$R_T = 120\ \Omega$

EMC-conformant wiring of the network

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

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

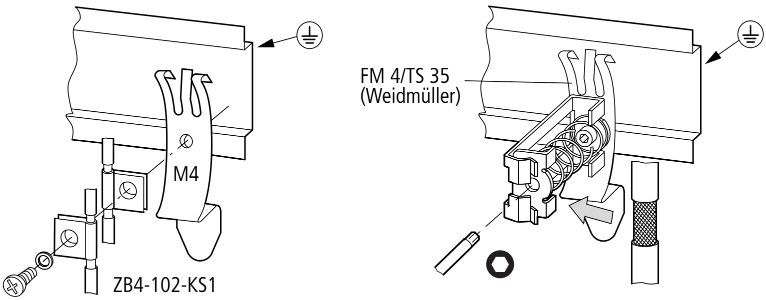


Figure 8: Shielding connection to the mounting rail

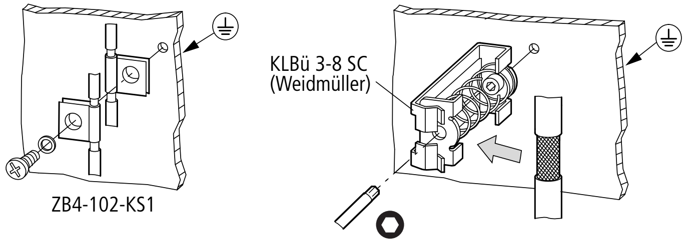


Figure 9: Shielding connection to the mounting plate

Potential isolation

The following potential isolation specifications apply to EASY222-DN interfaces:

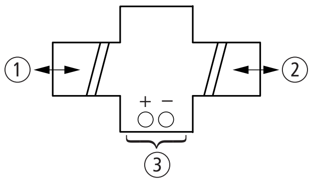


Figure 10: Potential isolation between supply voltage and outputs

1. Safe electrical isolation between easyLink and the 240 V AC mains
2. Simple electrical isolation to the DeviceNet communication bus
3. 24 V DC supply voltage

Data transfer rates – automatic baud rate detection

After it is switched on, the EASY222-DN module automatically detects the data transfer rate of the communication network. However, this is possible only if at least one network node transmits valid message frames. The device supports the following data transfer rates according to ODVA:

- 125 Kbit/s,
- 250 Kbit/s,
- 500 Kbit/s,

Maximum distances and bus cable lengths

The max. bus length is not determined by the data transfer rate, but rather by the cable used. The following cables are permitted:

- A so-called "Thin Cable",
- a "Thick Cable"
- or a "Flat Cable".

The data cable requirements are specified by the ODVA.

Baud rate [Kbit/s]	max. bus length in m		
	"Thick Cable"	"Thin Cable"	"Flat Cable"
125	500	100	420
250	250	100	200
500	100	100	100

3 Device operation

Initial starting

- ▶ Before you switch on the device, verify that it is properly connected to the power supply, to the bus connectors and to the basic unit.
- ▶ Switch on the power supply for the basic unit and the EASY222-DN.

The LEDs of the EASY222-DN flicker.
The device is in the mode for detection of the correct baud rate (→ section "Data transfer rates – automatic baud rate detection" on page 22).
The GW information (intelligent station connected) is 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
MFD-CP10…	01	Flashing

As soon as the device in the network management is switched to the "Operational" status, the state of the GW changes to static even on the devices with a flashing GW, → section "Network status LED" on page 28).

If the EASY222-DN has factory settings (node ID = 127), you need to define the DeviceNet slave address.

DeviceNet setting the slave address

Each DeviceNet slave requires a unique address (MAC ID) in the DeviceNet structure. Within a DeviceNet structure, you can assign a maximum of 64 addresses (0 to 63).
Each MAC ID must be unique within the entire bus structure.

There are three ways to set the DeviceNet address of an EASY222-DN:

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

Setting the address at the basic unit with display:

Basic requirements:

- The respective basic devices (easy600, easy700, easy800 or MFD-Titan) and EASY222-DN are supplied with voltage.
- The basic unit is accessible (password protection not activated).
- The basic unit has a valid operating system version.
- The basic unit is in STOP mode.

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

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

▶ Use the cursor keys ˄ or ˅ to change to the CONFIGURATOR.

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

▶ Confirm with OK.

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

▶ Select the LINK.... menu with the easy800/MFD units

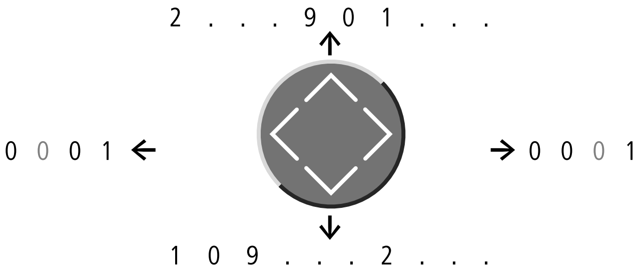
▶ Confirm with OK.

The DEVICENET menu appears.

```
DEVICENET

MAC ID 0026
222-01.20- D
```

▶ Set the address by means of the cursor buttons:
  – Set the current numeric value via the ⌃ or ⌄ keys.
  – You can change the current numeric value via ‹ or ›.



▶ Accept the address with OK.

▶ Cancel address input with ESC.

Information about the 4th display line:

xxx-xx.xx-xx

222-02.10- b

Hardware version, Index: b

Software version, OS version: 2.1

Device identity: EASY222-DN

Setting the address by means of easySoft

With easySoft, version 3.1
‹Menu → Online → Configuration of expansion units›

With easySoft, from version 4.01
‹Menu → Communication → Configuration → Expansion units → EASY222-DN›.

→ The menu is only available in the communication view; therefore please activate the "Communication" tab.

→ The following applies for device version identity 01:

After you have changed the MAC ID via the basic device you must restart EASY222-DN. To do this switch the power supply off and on again. EASY222-DN devices with a version ID > 01 take on the address automatically.

Setting the address via the master PLC

The configuration software supplied with your master PLC offers a further option of setting or modifying the MAC ID of the gateway. For more information, refer to the included PLC documentation.

You can also use various other software packages to modify the MAC ID, e.g. by sending an explicit message. Do so by using the corresponding service of the DeviceNet object (Section “DeviceNet object”, page 37).

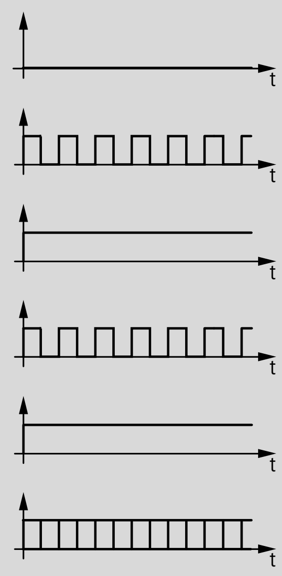
LED status indication

The EASY222-DN expansion device is provided with two LEDs. These provide fast support for troubleshooting. EASY222-DN monitors itself as well as the DeviceNet communication bus.

Module status LED

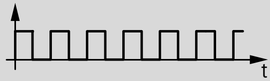
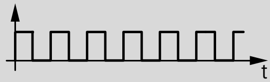
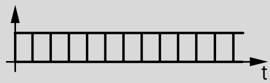
The dual-color LED (GREEN/RED) indicates the status of EASY222-DN. It monitors whether the device is fully functional and operates without fault.

OFF	No power supply at the EASY222-DN.
GREEN flashing	EASY222-DN is in standby mode. The configuration is faulty or incomplete, or a configuration does not exist.
GREEN	EASY222-DN is in normal operational state.
RED flashing	An error has occurred. There is no need to replace the EASY222-DN.
RED	A fatal error has occurred EASY222-DN. EASY222-DN must be replaced.
GREEN-RED flashing	EASY222-DN is performing a self-test.



Network status LED

The dual-color LED (GREEN/RED) indicates the status of the DeviceNet communication bus. This function monitors operability and correct operation of the EASY222-DN.

OFF	EASY222-DN is offline. Either it is performing a DUP_MAC_ID test or power is missing at the device or bus.	
GREEN	EASY222-DN is online and the connection is active.	
GREEN flashing	EASY222-DN is online. Communication has not yet been established.	
RED flashing	Time-out of at least one I/O connection (time-out state).	
RED	A fatal network error has occurred. EASY222-DN has shut down communication.	
GREEN-RED flashing	EASY222-DN has detected a network access error and is now in communication error state.	

Cycle time of the "easy" basic unit

Network traffic between the easy/MFD basic unit and the EASY222-DN via easyLink extends the cycle scan time of the basic device

In the worst case, this time can be extended by 25 ms.

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

EDS file

You can implement EASY222-DN into the DeviceNet structure by means of a standardized EDS file (Electronic Data Sheet).

This EDS file primarily defines the polled I/O connection, the COS I/O connection and the cyclic I/O connection of the gateway. It does not contain data or parameters (easy object) for functions of the easy basic unit. These functions are accessed by means of explicit messages.

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

http://www.eaton.com/moeller → Support → Search term „EASY222-DN"

Follow the "Link" on this page.

A printed version of the EDS file can be found in the annex (→ section "EDS file", page 259).

→ Note on the EDS file:

The Identity Object entry - Major Revision defines the current operating system state of the EASY222-DN communication module. As the device with a newer operating system version can deviate from the EDS description in this point, this entry must be modified accordingly, → section "Identity object" on page 35.

4 DeviceNet functions

Object model

EASY222-DN is based on the Communications Adapter Profile according to ODVA specifications (Release V2.0).

The DeviceNet object model can be used to describe all EASY222-DN functions. The object model reflects the principle of communication at the application layer. This manual deals in the following only with objects relevant for your application. Primary topic is the manufacturer-specific class easy object.

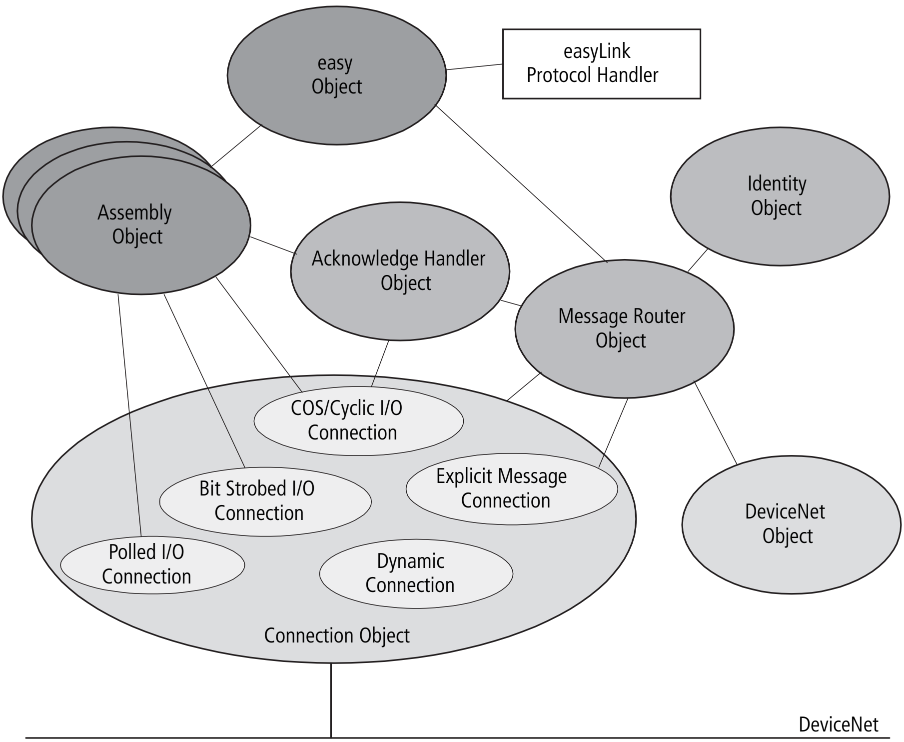


Figure 11: DeviceNet objects

The DeviceNet objects in the illustration can be compiled again as “Management objects”, “Connection objects” and “Manufacturer-specific objects”. Their tasks will be briefly explained after the following.

	Object address		Service address	Function
	Class ID [hex]	**Instance ID [hex]**	**[hex]**	**Attribute ID [hex]**
① **Management objects**				
Identity object	01	01		→ page 33
Message Router	02	01		
② **Connection objects**				
DeviceNet object	03	01		→ page 33
Connection Object	05	01 – 04, 04 – 0F		
③ **Manufacturer-specific objects**				
easy Object	64	01		→ page 38
Direct access: Inputs/outputs, operating mode				
Read			0E	→ chapter 5
Write			10	
Extended access: time, image data, function blocks			32	
easy600				→ chapter 6
easy700				→ chapter 7
easy800/MFD				→ chapter 8
Assembly Object	04	64 – 66		

① Management objects

These define DeviceNet-specific data and functions and must be supported by all DeviceNet devices:

- Identity Object

The Identity Object (Class ID 01_{hex}) contains all data for unique identification of a network node, e.g. the Vendor ID, Device Type and Product Code. It also comprises the actual status of a device, the serial number and the product name.

Detailed information → page 35.

- Message Router Object

The Message Router Object (Class ID 02_{hex}) provides access to all classes and instances in the device by means of explicit messages.

② Connection objects

Define messages exchanged via DeviceNet:

- DeviceNet object

The DeviceNet object (Class ID: 03_{hex}) must be supported by each device. It defines the physical connection of a device to the DeviceNet network. This means it includes the device address (MAC ID) as well as the currently set baud rate.

Detailed information → page 37.

- Connection Object

The Connection Object (Class ID: 05_{hex}) is supported by all DeviceNet devices in at least one instance. It defines the access to data via I/O messages or explicit messages, the path and length of producer/consumer data, the CAN connection identifier, the watchdog and the error response.

③ **Manufacturer-specific objects**
Define device-specific data and functions (Application Objects, Parameter Object, Assembly Object).

- Application Objects – easy Object

Application objects (Class ID: 64_{hex}) describe simple applications for automation engineering. They are either predefined in the DeviceNet object library or by the user.

Detailed information → page 38.

- Assembly Objects

The Assembly Object (Class ID: 04_{hex}) provides the user with mapping options, i.e. attribute data of different instances in different classes can be grouped together to form a single attribute of an instance in an assembly object.

Identity object

Object address		Function	Access
Class ID	Instance ID	Attribute ID	Service code
01_{hex}	01_{hex}	→ table 1	→ table 2

Table 1: Attribute IDs of the Identity Object instance

Attri-bute ID	Access	Name	Description	Size [byte]
1	Read	Vendor ID	The vendor ID is issued by the ODVA. This is 248_{dec} for Eaton GmbH.	2
2	Read	Device type	The EASY222-DN belongs to the communication adapters category. The value for this is 12_{dec}.	2
3	Read	Product code	The product code is defined by Eaton: 650_{dec}. It describes the model number.	2
4	Read	Device version	Two bytes are returned when the device version is read.	
		Hardware version,	The low byte defines the hardware version, the high byte the operating system version.	1
		Operating system version		1
5	Read	Status	This attribute describes the global status of the device.	2
6	Read	Serial number	The serial number of the device can be read with this attribute.	4
7	Read	Product name	The product name EASY222-DN is stored as an ASCII value (hex).	12
9	Read	Configuration consistency value	This attribute returns a counter value that monitors the number of modifications in non-volatile memory (E2PROM).	2
10	Read/ Write	Heartbeat Interval	Defines an interval between heartbeat messages in [s].	2

Service code

The Identity Object Instance and also the following instances support the services listed in the table below.

Table 2: Service code

Service code value	Service name	Description
05_{hex}	Reset	Calls the Reset function of the EASY222-DN communication module.
$0E_{hex}$	Get_Attribute_Single	This service can be used to fetch the value of a selected attribute from the communication module.
10_{hex}	Set_Attribute_Single	This service can be used to set a selected attribute in the device.

DeviceNet object

Object address		Function	Access
Class ID	Instance ID	Attribute ID	Service code
03_{hex}	01_{hex}	→ table 3	→ table 2

The DeviceNet object instance is used to configure the EASY222-DN communication module and to define the physical environment. The same Service Codes are used as for the Identity Object.

Table 3: Attribute IDs of the DeviceNet Object instance

Attribute ID	Access	Name	Description	Size [byte]
1	Read/Write	MAC ID	The MAC ID represents the network address of a network node. It can be read and set for EASY222-DN with this attribute via the fieldbus. Value range: 0 to 63_{dec}. (→ section “DeviceNet setting the slave address”, page 24)	1
2	Read/Write	Baud rate	This attribute can be used to read/set the data transfer rate for communication functions. Range of values: 0 to 2, 125 to 500 kbps (→ section “Data transfer rates – automatic baud rate detection”, page 22).	1
3	Read/Write	BOI (Bus-Off interrupt)	This attribute can be used to define the reaction to a Bus-Off event (CAN-specific).	1
4	Read/Write	Bus-Off counter	This values shows how often a Bus-Off event has occurred. Value range: 0 to 255.	1

easy Object

Object address		Function	Access
Class ID	Instance ID	Attribute ID	Service code
64_{hex}	01_{hex}	→ table 4	→ table 5

The easy object can be used to access easy/MFD functions via the DeviceNet communication bus. The table below shows the attributes supported by this object. The 2 bytes of the attributes 1 and 2 give the diagnostics data of the device. Attribute 3 can be use to access the outputs (S1 to S8) and attribute 4 to access the inputs (R1 to R16) of the basic unit.

By using a DeviceNet configuration software (e.g. RS NetWorx), you can map these data directly to the corresponding memory areas of a PLC.

Table 4: Attribute IDs of the Easy Object instance

Attribute ID	Access	Name	Description	Size [byte]
1	Read	easy Status	This attribute can be used to read the status of easy (RUN or STOP).→ table 6	1
2	Read	Coupling Module Status	This attribute can be used to read the status of easyLink.→ table 6	1
3	Read	Inputs – Send Data	easy transfers the input data to the DeviceNet bus. The easy outputs S1 to S8 must be used for this function. The structure of these 3 bytes is described in detail under Section "Input data: Mode, S1 – S8", page 49.	3

Attribute ID	Access	Name	Description	Size [byte]
4	Read/ Write	Outputs – Receive Data	The DeviceNet bus transfers the data to easy. The easy inputs R1 to R16 must be used for this function. The structure of these 3 bytes is described in detail under Section "Output data: mode, R1 – R16", page 51.	3
5	Read/ Write	Predefined Outputs	This attribute is used to predefine the output data (R data) that the EASY222-DN device indicates on power up. The structure of these 3 bytes is described in detail under Section "Output data: mode, R1 – R16", page 51.	3

Service code

The easy object instance supports the following services.

Table 5: Service code

Service code value	Service name	Description
$0E_{hex}$	Get_Attribute_Single	This service can be used to fetch the value of a selected attribute from the communication module.
10_{hex}	Set_Attribute_Single	This service can be used to set a selected attribute in the device.
32_{hex}	Extended access1)	This service can be used to address the supplementary parameters1) of the control relay:

1) Additional parameters are "Time", "Image data" and "Function block". Addressing of the parameters is easy specific and is described in chapters 5 – 7 in detail.
Extended access is implemented via explicit message transfer. This transfer protocol allows the exchange of control data. Further information about the transfer protocol can be found in Section "DeviceNet Communication profile" on page 41.

Change of State I/O connection

Table 6: Diagnostics data: 2 Byte

Byte	Meaning	Value	Meaning
0	easy status (attribute ID 1)	00_{hex}	Static value.
1	Coupling module status (attribute ID 2)	00_{hex}	The basic unit is connected with the EASY222-DN gateway via the easyLink.
		04_{hex}	The basic device is not switched on or not connected with the EASY222-DN gateway via the easyLink.

→ When communication between the easy/MFD basic device and the EASY222-DN expansion device, is interrupted, the corresponding error code is generated in the third data byte. Furthermore, the R/S data of the gateway is transferred with the value 00_{hex}.

DeviceNet Communication profile

DeviceNet is based on a connection-oriented communication model. This means that the data can always only be exchanged via the specific connections assigned to the devices.

DeviceNet stations communicate either by means of I/O messages or explicit messages.

I/O Messages

I/O messages are used to exchange high-priority process and application data via the network. The communication between the stations on the DeviceNet is implemented here with the client/server model. This means: a "producing" application transfers data to one or several "consuming" applications. It is entirely possible that several application objects are addressed in a single device.

Prerequisite for communication between the units via I/O messages is the implementation of an I/O Messaging Connection Object. You can activate this function in two ways:

- Either by means of a static and in the unit already existing "I/O connection object" or via the "Predefined Master/ Slave Connection Set", or
- by means of a dynamically set I/O connection object which you can configure using an Explicit Messaging Connection Object that already exists in the device.

Explicit Messages

Explicit messages are used to transfer low-priority configuration data, general management data or diagnostics data between two specific devices. This is always a point-to-point connection in a client/server system, which means that a response must always be issued by the server after a request from a client.

Same as for I/O messaging, the prerequisite for explicit messaging between the is the implementation of a "Connection Object", namely the Explicit Messaging Connection Object". This can be achieved either by activating an existing static connection object in the unit, or via the Predefined Master/Slave Connection Set", or dynamically across the so-called UCMM port (Unconnected Message Manager Port) of a device.

All data of the function relay (easy basic unit) are processed by means of explicit messages. The master PLC can thus read/write access the parameters of the following functions.

- Time
- Image data
- Function blocks (counters, timers, analog value comparators,...).

→ The DeviceNet connection of the easy control relay to an SLC 500 requires specific control and handshake routines in the PLC program for the execution of the control commands (Explicit Messages).

The application note AN2700K17G supports the control commands of EASY222-DN. It provides subroutines in the program for controlling the required “Explicit Messages”, i.e. the programming is replaced by the call and the parameter assignment of the subroutine. Parameters are assigned by means of an integer file.

The self-unpacking application note AN2700K17G.exe (for easy600) is available at https://es-assets.eaton.com/AUTOMATION/APPLICATION_Notes/an27k17g.exe for download.

General method of operation

The following is a description of the general operation with the EASY222-DN. The acyclic data transfer is implemented with the aid of explicit messages. The function blocks of the easy basic unit can be addressed via the service code = 32_{hex}. The assigned attribute ID is used here to distinguish between different parameters and functions.

Service code	Object address	
	Class ID	Instance ID
32_{hex}	64_{hex}	01_{hex}

Note:
DeviceNet is based on the standard CAN protocol and therefore uses an 11-bit message identifier. As a result 2^{11} = 2048 messages (000_{hex} - $7FF_{hex}$) can be defined. As the maximum number of stations on a DeviceNet network is 64 stations, 6 bits are used for identifying a device. These are referred to as the MAC-ID (device or node address).

Four message groups of differing sizes are available to suit the utilization model.

In DeviceNet language terms the CAN identifier is referred to as the Connection ID. This is comprised of the identifier for the message group (Message ID) and the MAC ID of the device:

- The source and target addresses are possible as the MAC ID; the definition is dependant on the message group and message ID.
- The significance of the message is defined in the system with the message ID.

The world of the DeviceNet provides four message groups. The EASY222-DN uses message group 2. This group uses 512 CAN identifiers (400_{hex} – $5FF_{hex}$). Most messages Ids of this group are optionally defined for using the Predefined Master/Slave Connection Sets. A message ID is used for network management. The priority is defined here primarily by the device address and only then by the message ID. A closer look at the bit position shows that a CAN controller with an 8-bit mask is able to selectively filter out its group 2 messages.

Connection ID = CAN identifier											Meaning
10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
1	0	MAC ID						Message ID			Message group 2
1	0	Source MAC ID						0	0	0	Master's I/O Bit–Strobe Command Message
1	0	Source MAC ID						0	0	1	Reserved for Master's Use – Use is TBD
1	0	Destination MAC ID						0	1	0	Master's Change of State or Cyclic Acknowledge Message
1	0	Source MAC ID						0	1	1	Slave's Explicit/ Unconnected Response Messages
1	0	Destination MAC ID						1	0	0	Master's Explicit Request Messages
1	0	Destination MAC ID						1	0	1	Master's I/O Poll Command/Change of State/Cyclic Message
1	0	Destination MAC ID						1	1	0	Group 2 Only Unconnected Explicit Request Messages
1	0	Destination MAC ID						1	1	1	Duplicate MAC ID Check Messages

Source: ODVA- DeviceNet Specification Release 2.0, Chapter 7-2

The data transfer on the DeviceNet communication bus is indicated in the following table. The data flow indicates the telegram for reading the date and time in the easy700 (→ section "Read/write date and time" on page 101).

The EASY222-DN communication module has the MAC ID = 3. It must be taken into account with the data sequence that the access is implemented in fragmented form. Further information on this is provided in the ODVA specification.

Description		ID (hex)	Length	DeviceNet – Byte (hex) 0	1	2	3	4	5	6	7
Master sends a request (hex) with:		41C	8	80	00	32	64	01	**93**	**05**	**00**
Byte 2 - service code = 32 Byte 3 - CLASS ID = 64 Byte 4 - Instance ID = 01	DeviceNet specific										
Byte 5 - Attribute ID = 93 Byte 6 - Len = 05 Byte 7 - Index = 0	easyLink specific										
Confirmation of the slave (Fragmentation protocol)		41B	3	80	C0	00					
Master sends remaining easyLink byte		41C	6	80	01	**00**	**00**	**00**	**00**		
Byte 2 - Data 1 = 00 Byte 3 - Data 2 = 00 Byte 4 - Data 3 = 00 Byte 5 - Data 4 = 00											
Acknowledgement of the slave (Fragmentation protocol)		41B	3	80	C1	00					
Slave sends a response to the request		41B	8	80	00	B2	**C2**	**05**	**00**	**05**	**09**
Byte 3 – response = C2 (read successful) Byte 4 – Len = 05 Byte 5 – Index = 00 Byte 6 – Data 1 = 05											
Acknowledgement from master (Fragmentation Protocol)		41C	3	80	C0	00					
Slave sends remaining easyLink data:		41B	5	80	81	**0D**	**05**	**04**			
Data 2 = 0D Data 3 = 05 Data 4 = 04											
Acknowledgement from master (Fragmentation protocol)		41C	3	80	C1	00					

5 Direct data exchange with easy/MFD (Polled I/O Connection)

The DeviceNet master can exchange the following data with the easy/MFD via the direct cyclic data exchange:

- Write operation
  - Setting or /resetting of the easy/MFD inputs
  - Determination of the RUN/STOP mode.
- Read operation
  - Scanning the output states of the easy/MFD
  - Scanning the mode of the easy/MFD.

In order to transfer data between the slave EASY222-DN and a DeviceNet master control, you must map the respective cyclic data to the respective slave configuration.

→ The interconnection to the DeviceNet controls from Allen Bradley is implemented using an assignment table in the RS-NetWorx software tool.

→ The terms “input data” and “output data” are used relative to the point of view of the DeviceNet master.

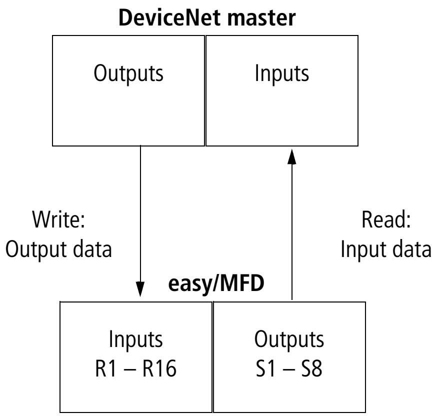


Figure 12: Input and output data relative to the DeviceNet master

Input data: Mode, S1 – S8

Attribute ID: 3

The cyclic data transfer between DeviceNet master and the EASY222-DN slave is provided by the input data byte 0, 1 and 2.

Table 7: Byte 0 to 2: input data, mode

Byte	Meaning	Value
0	Operating mode scan	→ table 8
1	Scan status of the easy outputs S1 to S8	→ table 9
2	Not used	00_{hex}

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

Table 8: Byte 0: Operating mode

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

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

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

Table 9: Byte 1: Status of the easy/MFD outputs S1 to S8

easy/MFD	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 terminated.

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.

Output data: mode, R1 – R16

Attribute ID: 4

The cyclic data transfer between DeviceNet master and the EASY222-DN slave is provided by the output data byte 0, 1 and 2.

Table 10: Byte 0 to 2: output data, mode

Byte	Meaning	Value
0	Specifying the control mode	→ table 11
1	Setting/resetting of the easy/MFD inputs R9 to R16	→ table 12
2	Setting/resetting of the easy/MFD inputs R1 to R8	→ table 13

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

Table 11: Byte 0: Operating mode

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

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

Explanation:

Value $14_{hex} = 0001\,0100_{bin}$:
Byte 0 must always contain this value if data are to be written to the easy/MFD basic unit via the EASY222-DN gateway.

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

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

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

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

Table 12: Byte 1: Setting/resetting of the easy/MFD inputs R9 to R16

easy/MFD	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} = $0001\,1001_{bin}$:
Enable R13, R12 and R9.

Table 13: Byte 2: Setting/resetting of 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 executed.

6 Control commands for easy600

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

- „Read and write date and time, summer and winter time“ (page 57)
- „Read image data“ (page 61)
- „Read/write function blocks“ (page 72).

For this the message transfer protocol of the explicit messages is accessed in the master controller. All the parameters are addressed via the service code 32_{hex}. The assigned attribute ID is here used to distinguish between different parameters and functions.

Service code	Object address	
	Class ID	Instance ID
32_{hex}	64_{hex}	01_{hex}

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

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

A data exchange procedure is required in order to ensure the safe exchange of data via DeviceNet from master to slave and vice versa.

→ The operating mode of the basic unit must correspond with the status indicated at the LEDs when the various parameters are being set.

In the communication between the stations, the master initiates the data exchange with a control command. The slave always gives a response to the request. The response will provide information whether the data exchange was executed or not. An error code is returned if the data exchange could not be executed. This is defined exactly by the ODVA, → section "References" on page 8.

Read and write date and time, summer and winter time

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID										
		Read	**5D**	–	0	1	0	1	1	1	0	1
		Write	**2A**	–	0	0	1	0	1	0	1	0
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Write successful	–	C1	1	1	0	0	0	0	0	1
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Day of week										
		Read operation	00	→ table 14								
		month Write operation	→ table 14	00								
1	2	Hour										
		Read operation	00	→ table 15								
		month Write operation	→ table 15	00								
2	3	Minute										
		Read operation	00	→ table 16								
		month Write operation	→ table 16	00								
3	4	Summer-/winter switchover										
		Read operation	00	→ table 17								
		month Write operation	→ table 17	00								

M = master

S = Slave

Table 14: Byte 0 (master) or byte 1 (slave): weekday (value range 00 to 06)

Day of week	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 15: Byte 1 (master) or byte 2 (slave): 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 16: Byte 2 (master) or byte 3 (slave): 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 17: Byte 3 (master) or byte 4 (slave): 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

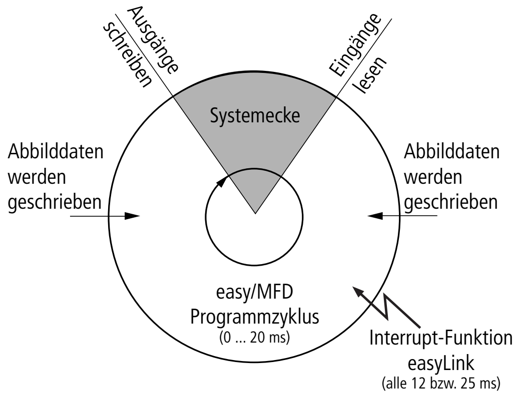
Example:
It is Friday, the current time-of-day is set to CET summer time, 14:36 p.m. .

Byte		Meaning	Value (hex), sent by		Bit							
M	**S**		**Master**	**Slave**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
		Attribute ID										
		Write	**2A**	–	0	0	1	0	1	0	1	0
	0	Response										
		Write successful	–	C1	1	1	0	0	0	0	0	1
0	1	Day of week	04	00								
1	2	Hour (14_{dec})	0E	00								
2	3	Minute (36_{dec})	24	00								
3	4	Summer-/winter switchover	01	00								

M	= master
S	= Slave

Read image data

General information on working with image data



When writing the image data, it must be taken into account that an image used in the easy/MFD program (e.g. inputs, outputs,…) is also written cyclically by the actual program. Only the image data that is not used in the program and is thus not written in the program cycle is unchanged. This operation also means that an image written via the easyLink, e.g. output data is only output to the physical outputs of the easy/MFD if the control relay is operating in RUN mode.

Overview

Operands	Meaning	Read/Write	Attribute ID	Page
I1 – I16, P1 – P4, ESC/OK/DEL/ALT	„Digital inputs, P buttons and operating buttons“	Reading	**5C**	62
I7 – I8	„Analog inputs: I7 – I8“	Reading	**5B**	65
T1 – T8, C1 – C8, Ö1 – Ö4, A1 – A8	„Timing relays, counter relays, timer switch, analog value comparator“	Reading	**5E**	66
M1 – M16, Q1 – Q8, D1 – D8	„Auxiliary relay (marker), digital outputs, text display“	Reading	**5F**	69

Digital inputs, P buttons and operating buttons

Using the following command the logical states of the digital button inputs P1 to P4 as well as the logical states of the digital inputs I1 to I16 can be read.

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 M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID										
		Read	**5C**	–	0	1	0	1	1	1	0	0
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Status of inputs I1 to I8	00	→ table 18								
1	2	State of the inputs I9 to I16	00	→ table 19								
2	3	State of the buttons	00	→ table 20								

M = master

S = Slave

Table 18: Byte 1: status 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 19: Byte 2: status 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 20: Byte 3: Status of pushbuttons

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

Analog inputs: I7 – I8

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

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

M = master

S = Slave

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

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

Example:

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

Timing relays, counter relays, timer switch, analog value comparator

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

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID										
		Read	**5E**	–	0	1	0	1	1	1	1	0
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Status of timing relay	00	→ table 21								
1	2	Counter relay status	00	→ table 22								
2	3	Time switch status	00	→ table 23								
3	4	Analog value comparator status	00	→ table 24								

M = master

S = Slave

Table 21: 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 22: 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 23: 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 24: 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.

Auxiliary relay (marker), digital outputs, text display

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

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID										
		Read	**5F**	–	0	1	0	1	1	1	1	1
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Status of markers M1 to M8	00	→ table 25								
1	2	Status of markers M9 to M16	00	→ table 26								
2	3	Status of digital outputs Q1 to Q8	00	→ table 27								
3	4	Status of text display markers D1 to D8	00	→ table 28								

M = master

S = Slave

Table 25: Byte 1: Status of the marker relays 1 to 8

	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 26: Byte 2: Status of the marker relays 9 to 16

	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 27: 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 28: 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 on **command** represents a command for easy600 and defines the meaning of the remaining 6 data bytes. The following table shows the possible commands.

Operands	Meaning	Command	Page
A1 – A8	„Analog value comparator A1 – A8: write actual values (function, comparison values)“	22_{hex} – 29_{hex}	73
C1 – C8	„Counter relays C1 – C8: read actual value“	$3B_{hex}$ – 42_{hex}	76
	„Counter relay C1 – C8: write reference value“	09_{hex} – 10_{hex}	78
	„Counter relay C1 – C8: read reference value“	43_{hex} – $4A_{hex}$	80
T1 – T8	„Timing relays T1 – T8: read actual value (timing range, actual value, switching function)“	$2B_{hex}$ – 32_{hex}	82
	„Timing relays T1 – T8: write parameters (timing range, reference value, switching function)“	01_{hex} – 08_{hex}	86
⌚1 – ⌚4	“Time switch ⌚1 – ⌚4: read actual value (channel, ON time, OFF time)”	$4B_{hex}$ – $5A_{hex}$	90
	“Time switch ⌚1 – ⌚4: read setpoint value (channel, ON time, OFF time)”	12_{hex} – 21_{hex}	94

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

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID: Write										
		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
	0	Response										
		Write successful	–	C1	1	1	0	0	0	0	0	1
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte:	→ table 29	00								
1	2	Comparison value for comparison with constant	→ page 74	00								

M = master

S = Slave

→ Keep to the value range: The comparison values as well as the function are part of an "*.eas file". If these values are changed, the original "*.eas file" no longer matches the file in the EASY6….

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

Table 29: Byte 0: control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
Compare: "≧"								0
Compare: "≦"								1
I7 to 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						
Edit	1							

Example:
82_{hex} = $1000\,0010_{bin}$ means that the selected analogue value comparator will be enabled in the circuit diagram of the basic unit as soon as the analogue value input I7 ≧ the defined constant (→ byte 1).

Comparison value (byte1)
This byte contains the comparison value in the form of a constant. It is between 0 to 99 and corresponds to a comparison voltage from 0.0 to 9.9 V. You must also state this value in hexadecimal format.

Example:
The reference 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
	Attribute ID: A8	29	0	0	1	0	1	0	0	1
0	Control byte:	→	1	0	0	0	0	0	1	1
1	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	C1	1	1	0	0	0	0	0	1
1	Comparator	00								
2	Comparison value for comparison with constant	00								

Counter relays C1 – C8: read actual value

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID: 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
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte:	00	→ table 30	x	x	x	x	x	x	x	x
1	2	Counter relay actual value (low byte)	00	→ page 77								
2	3	Counter relay actual value (high byte)	00									

M = master

S = Slave

Table 30: Byte 1: control byte

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

Process variable (byte 2 and byte 3)
These two bytes define the process variable of the counter relay. The value of the process variable can lie within the range 0 to $9\,999_{dec}$. In order to determine the corresponding process variable, you need to convert the 16-bit hexadecimal low and high values into the decimal format.

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

Counter relay C1 – C8: write reference value

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID: Write										
		C1	**09**	–	1	0	0	0	1	0	0	1
		C2	**0A**	–	1	0	0	0	1	0	1	0
		C3	**0B**	–	1	0	0	0	1	0	1	1
		C4	**0C**	–	1	0	0	0	1	1	0	0
		C5	**0D**	–	1	0	0	0	1	1	0	1
		C6	**0E**	–	1	0	0	0	1	1	1	0
		C7	**0F**	–	1	0	0	0	1	1	1	1
		C8	**10**	–	1	0	0	1	0	0	0	0
	0	Response										
		Write successful	–	C1	1	1	0	0	0	0	0	1
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte:	→ table 31	00								
1	2	Setpoint value (low byte)	→ page 79	00								
2	3	Setpoint value (high byte)		00								

M = master
S = Slave

Value range of the counter values: 0000 to 9999

→ Keep within the value range.

The value is part of an easySoft file (*.eas). If these values are changed, the original "*.eas file" no longer matches the file in the EASY6…..

Remember this feature when uploading, downloading or comparing "easy" circuit diagrams with easySoft.

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

The comparison shows that the circuit diagrams are not identical.

Table 31: Byte 0: control byte

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

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

Setting the reference value (byte 1 and byte 2)
These two bytes determine the reference value of the counter relay. The reference value can be set within the range from 0 to 9999_{dec}. To do so, you must convert the required decimal into the equivalent hexadecimal value and then split it up into the low-byte and high-byte.

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

Counter relay C1 – C8: read reference value

Telegram structure

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID: Read										
		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
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte:	00	→ table 32								
1	2	Counter relay reference value (low byte)	00	→ page 81								
2	3	Counter relay reference value (high byte)	00									

M = master

S = Slave

Table 32: Byte 1: control byte

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

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

Reference value (byte 2 and byte 3)

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

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

Timing relays T1 – T8: read actual value (timing range, actual value, switching function)

Telegram structure

Byte		Meaning	Value (hex), sent by		Bit
M	**S**		**Master**	**Slave**	**7 6 5 4 3 2 1 0**
		Attribute ID: 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
	0	Response			
		Read successful	–	C2	1 1 0 0 0 0 1 0
		Command rejected	–	C0	1 1 0 0 0 0 0 0
0	1	Control byte:	00	→ table 33	
1	2	Time actual value (low byte)	00	→ page 84	
2	3	Time actual value (high byte)	00		
3	4	Random value	00	→ page 84	
4 – 5	5 – 6		00	00	

M = master

S = Slave

Table 33: 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 time with random switching						0	1	0
Off-delayed with random switching,						0	1	1
Single pulse						1	0	0
Flashing						1	0	1
s time base				0	0			
M:S time base				0	1			
Time base “H:M”				1	0			
Not used			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							

Process variable (byte 2 and byte 3)

These two bytes determine the process variable of the timing relay. The process variable 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} ($3B_{hex}$). The table below is the results:

Table 34: Bytes 2 to 3: time actual value

Time base	Low value	High-value
millisecond	0 to 59 (10 ms)	0 to 59 s
Second	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 set reference time for relays operating with random switching characteristics. This reference 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	Attribute ID: 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 (timing range, reference value, switching function)

Byte M	Byte S	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave	Bit 7	6	5	4	3	2	1	0
		Attribute ID: 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
	0	Response										
		Write successful	–	C1	1	1	0	0	0	0	0	1
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte:	→ table 35	invalid								
1	2	Low reference value	→ page 89	00								
2	3	High reference value										
3 – 5	4 – 6		00	00								

M = master

S = Slave

→ 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 time values and the setpoint of the timing relay are part of an “*.eas file”. If these values are changed, the original “*.eas file” no longer matches the file in the EASY6…..

Remember this characteristic when uploading, downloading or comparing “easy” circuit diagrams with easySoft.

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 35: Byte 0: control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
On-delayed						0	0	0
Off-delayed						0	0	1
On time 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			
M:S time base				0	1			
Time base “H:M”				1	0			
Not used			0					
Does not appear in the parameter menu		1						
Appears in the parameter menu		0						
Edit	1							

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

Timing relay, setting the reference value (byte 1 and byte 2)

Bytes 1 and 2 determine the reference value for the timing relay. The reference value is based on the selected time base. When 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} ($3B_{hex}$). The table below is the results:

Time base	Low value	High-value
Milliseconds	0 to 59 (10 ms)	0 to 59 s
Second	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]

Time switch ⏲1 – ⏲4: read actual value (channel, ON time, OFF time)

Telegram structure

Byte		Meaning	Value (hex), sent by		Bit							
M	S		Master	Slave	7	6	5	4	3	2	1	0
		Attribute ID: Read										
		⏲1 channel A	**4B**	–	0	1	0	0	1	0	1	1
		⏲1 channel B	**4C**	–	0	1	0	0	1	1	0	0
		⏲1 channel C	**4D**	–	0	1	0	0	1	1	0	1
		⏲1 channel D	**4E**	–	0	1	0	0	1	1	1	0
		⏲2 channel A	**4F**	–	0	1	0	0	1	1	1	1
		⏲2 channel B	**50**	–	0	1	0	1	0	0	0	0
		⏲2 channel C	**51**	–	0	1	0	1	0	0	0	1
		⏲2 channel D	**52**	–	0	1	0	1	0	0	1	0
		⏲3 channel A	**53**	–	0	1	0	1	0	0	1	1
		⏲3 channel B	**54**	–	0	1	0	1	0	1	0	0
		⏲3 channel C	**55**	–	0	1	0	1	0	1	0	1
		⏲3 channel D	**56**	–	0	1	0	1	0	1	1	0
		⏲4 channel A	**57**	–	0	1	0	1	0	1	1	1
		⏲4 channel B	**58**	–	0	1	0	1	1	0	0	0
		⏲4 channel C	**59**	–	0	1	0	1	1	0	0	1
		⏲4 channel D	**5 A**	–	0	1	0	1	1	0	1	0
	0	Response										
		Read successful	–	C2	1	1	0	0	0	0	1	0
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte switching timer	00	→ table 36								
1	2	Control byte channel	00	→ table 37								
2	3	Minute (switch point ON)	00	→ page 93								
3	4	Hour (switch point ON)	00									
4	5	Minute (switch point OFF)	00									
5	6	Hour (switch point OFF)	00									

M = master

S = Slave

Table 36: Byte 1: "switching timer" control byte

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

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

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

Table 37: Byte 2: "channel" control byte

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

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

Example:
Value $31_{hex} = 0011\,0001_{bin}$:
The previously selected channel X of weekly timer Y is active Monday through Saturday.

Switching times (byte 3 to byte 6)

The table below shows which bytes precisely determine the ON and OFF times of a channel. The resolution is in seconds.

Switch on time		Switch Off Time	
bytes3	bytes4	bytes5	bytes6
Minute ON	Hour ON	Minute OFF	Hour OFF
00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})	00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})

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

Example:

Byte	Value	Description
0	42_{hex}	The read request has been executed. Data follow.
1	80_{hex}	The addressed switching timer is used in the circuit 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 ⌚1 – ⌚4: read setpoint value (channel, ON time, OFF time)

Telegram structure

Byte		Meaning	Value (hex), sent by		Bit							
M	S		Master	Slave	7	6	5	4	3	2	1	0
		Command										
		⌚1 channel A	12	–	0	0	0	1	0	0	1	0
		⌚1 channel B	13	–	0	0	0	1	0	0	1	1
		⌚1 channel C	14	–	0	0	0	1	0	1	0	0
		⌚1 channel D	15	–	0	0	0	1	0	1	0	1
		⌚2 channel A	16	–	0	0	0	1	0	1	1	0
		⌚2 channel B	17	–	0	0	0	1	0	1	1	1
		⌚2 channel C	18	–	0	0	0	1	1	0	0	0
		⌚2 channel D	19	–	0	0	0	1	1	0	0	1
		⌚3 channel A	1A	–	0	0	0	1	1	0	1	0
		⌚3 channel B	1B	–	0	0	0	1	1	0	1	1
		⌚3 channel C	1C	–	0	0	0	1	1	1	0	0
		⌚3 channel D	1D	–	0	0	0	1	1	1	0	1
		⌚4 channel A	1E	–	0	0	0	1	1	1	1	0
		⌚4 channel B	1F	–	0	0	0	1	1	1	1	1
		⌚4 channel C	20	–	0	0	1	0	0	0	0	0
		⌚4 channel D	21	–	0	0	1	0	0	0	0	1
	0	Response										
		Write successful	–	C1	1	1	0	0	0	0	0	1
		Command rejected	–	C0	1	1	0	0	0	0	0	0
0	1	Control byte (day begin/end)	→ page 95	00								
1	2	Minute (switch point ON)	→ page 97	00								
2	3	Hour (switch point ON)		00								
3	4	Minute (switch point OFF)		00								
4	5	Hour (switch point OFF)		00								
5	6	not used										

M = master

S = Slave

→ Keep to the value range: The values of minute and hour of the switch points are part of an easySoft file (*.eas). If these values are changed, the original "*.eas file" no longer matches the file in the EASY6….

Remember this feature when uploading, downloading or comparing "easy" circuit diagrams with easySoft. 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 weekly timer is assigned a control byte that defines the start/stop conditions. The table below shows the precise structure of this control byte.

Table 38: Byte 0: control byte

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

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

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

Setting the ON and OFF time (byte 2 to byte 5)

The table below shows which bytes precisely determine the ON and OFF times of a channel. The resolution is in seconds.

Switch on time		Switch Off Time	
bytes1	**Byte 2**	**bytes3**	**bytes4**
Minute ON	Hour ON	Minute OFF	Hour OFF
00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})	00 to $3B_{hex}$ (00 to 59_{dec})	00 to 17_{hex} (00 to 23_{dec})

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

Example:

Description	Instruction/byte	Value
Data of channel A of switching timer 4:	Attribute ID	$1E_{hex}$
Day: Monday through Saturday The channel appears in the parameter menu	Byte 0	31_{hex} (see above)
ON 19:00	bytes1	00_{hex}
	Byte 2	13_{hex}
OFF: 06:30	bytes3	$1E_{hex}$
	bytes4	06_{hex}

Example

The master initiates the command to write the following data to channel “C” ⌚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	Attribute ID: ⌚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								

7 Control commands for easy700

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

- „Read/write date and time“ (page 101)
- „Read/write image data“ (page 105)
- „Read/write function block data“ (page 126).

For this the message transfer protocol of the explicit messages is accessed in the master controller. The parameters are addressed via the service code 32_{hex}. The assigned attribute ID is here used to distinguish between different parameters and functions.

Service code	Object address	
	Class ID	Instance ID
32_{hex}	64_{hex}	01_{hex}

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

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

A data exchange procedure is required in order to ensure the safe exchange of data via DeviceNet from master to slave and vice versa.

→ The operating mode of the basic unit must correspond with the status indicated at the LEDs when the various parameters are being set.

In the communication between the stations the master initiates the data exchange with a control command. The slave always gives a response to the request. The response provides information whether the data exchange was executed or not. An error code is returned if the data exchange could not be executed. This is defined exactly by the ODVA, → section "References" on page 8.

Read/write date and time

→ Please also note the relevant description of the real-time clock provided in the easy700 manual (MN05013003Z-EN; previous description manual AWB2528-1508GB).

Telegram structure

Byte		Meaning	Value (hex), sent by	
M	**S**		**Master**	**Slave**
		Attribute ID		
		Read	**93**	–
		Write	**B3**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	05	05
1	2	Index	0 – 2 1)	0 – 2 1)
2 – 6	3 – 7	Data 1 – 5	depending on index, → table 39	depending on index, → table 39

1) 0 = Time/date, → table 39
1 = Summer time, → table 40
2 = Winter time, → table 41

M = Master

S = Slave

Table 39: Index 0 – date and time of real-time clock

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

Table 40: Index 1 – Summer time

Byte		Contents			Value (hex)
Master	Slave				
2	3	Data 1	Area		
				None	00
				Rule	01
				Automatic EU	02
				Automatic GB	03
				Automatic US	04
for “Area” = “Rule”:					
3	4	Data 2	Summer time switching rule		→ table 42
4	5	Data 3			
5	6	Data 4			
6	7	Data 5			

Table 41: Index 2 – Winter time (only valid if Area = "Rule" selected)

Byte		Contents		Value (hex)
Master	Slave			
2	3	Data 1	Area = Rule	01
3 – 6	4 – 7	Data 2 – 5	Winter time switching rule	→ table 42

Switching rule bit array

→ Please also read the detailed description in the easy700 manual (MN05013003Z-EN; previous description AWB2528-1508GB).

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

Table 42: Switching rule bit array

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

Read/write image data

→ Please also observe the relevant description of possible image data provided in the easy700 manual (MN05013003Z-EN, previous description AWB2528-1508GB) or in the easySoft Help.

The information provided in Section “General information on working with image data” on page 61 also applies to easy700.

Overview

Operands	Meaning	Read/Write	Type (hex)	Page
A1 – A16	„Analog value comparators/threshold comparators: A1 – A16“	Reading	8B	106
C1 – C16	„Counters: C1 – C16“	Reading	EE	107
D1 – D16	„Text function blocks: D1 – D16“	Reading	94	108
I1 – I16	„Local inputs: I1 – I16“	Reading	84	109
IA1 – IA4	„Local analog inputs: IA1 – IA4“	Reading	8C	110
M1 – M16, N1 – N16	„Write marker: M1 – M16/N1 – N16“	Writing	86/87	112
M1 – M16, N1 – N16	„Read marker: M1 – M16/N1 – N16“	Reading	86/87	114
O1 – O4	„Operating hours counters: O1 – O4“	Reading	EF	116
P1 – P4	„Local P buttons: P1 – P4“	Reading	8A	117
Q1 – Q8	„Local outputs: Q1 – Q8“	Reading	85	119
R1 – R16/ S1 – S8	„Inputs/outputs of easyLink: R1 – R16/S1 – S8“	Reading	88/89	120
T1 – T16	„Timers: T1 – T16“	Reading	ED	122
Y1 – Y4	„Year time switch: Y1 – Y8“	Reading	91	123
Z1 – Z3	„Master reset: Z1 – Z3“	Reading	93	124
H1 – H4	"Weekly timer: ⌚1 – ⌚8“	Reading	90	125

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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	8B	8B
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 43
4	5	Data 2 (Low Byte)	00	→ table 43
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 43: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	EE	EE
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 44
4	5	Data 2 (Low Byte)	00	→ table 44
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 44: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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**		**Master**	**Slave**
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	94	94
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 45
4	5	Data 2 (High Byte)	00	→ table 45
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 45: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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 Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	02	02
1	2	Part no.	84	84
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 46
4	5	Data 2 (High Byte)	00	→ table 46
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 46: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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 DeviceNet. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0 1)
0	1	Len	02	02
1	2	Part no.	8C	8C
2	3	Index	00 – 03 2)	00 – 03 2)
3	4	Data 1 (Low Byte)	00	→ table 47
4	5	Data 2 (High Byte)	00	→ table 47
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

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 47: Example telegram for reading the value at the analog input "1"

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	88	–
	0	Response: read successful	–	C2
0	1	Len	02	02
1	2	Part no.	8C	8C
2	3	Index	02 1)	02 1)
3	4	Data 1	00	4B
4	5	Data 2	00	03
5	6	Data 3	00	00
6	7	Data 4	00	00

1) 02 = Analog input I11

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

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

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

Write marker: M1 – M16/N1 – N16

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Write	**8C**	–
	0	Response:		
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Type[2]		
		With M marker	86	86
		With N marker	87	87
2	3	Index[2)]	00 – 0F	00 – 0F
3	4	Data 1 (Low Byte)[3]	00/01	00/01
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144
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 other than zero is written to the data byte.
If the value 0 is written to data byte Data 1, the marker is reset accordingly.

Example:
Marker M13 is set.

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Write	8C	–
	0	Response:		
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.		
		M marker	86	86
2	3	Index	0C	0C
3	4	Data 1	01	00
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Read marker: 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.		
		M marker	86	86
		N Marker	87	87
2	3	Index[2)]	00	00
3	4	Data 1 (Low Byte)	00	→ table 48
4	5	Data 2 (Low Byte)	00	→ table 48
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

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 48: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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**		**Master**	**Slave**
		Attribute ID: Read	88	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.		
		N Marker	87	87
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	04
4	5	Data 2 (Low Byte)	00	84
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	EF	EF
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 49
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 49: Byte 3 (master) or byte 4 (slave): 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 device).

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

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	8A	8A
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 50
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 50: Byte 3 (master) or byte 4 (slave): 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 DeviceNet fieldbus.

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	85	85
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 51
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 51: 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 easyLink: 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.		
		for R data	88	88
		for S data	89	89
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 52
4	5	Data 2 (Low Byte)	00	→ table 52
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 52: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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							

Timers: 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	ED	ED
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 53
4	5	Data 2 (Low Byte)	00	→ table 53
5 – 6	6 – 7	Data 3 – 4	00	00

1) Possible causes → page 144

Table 53: Byte 3 to 4 (master) or Byte 4 to 5 (slave): 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	91	91
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 54
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 54: Byte 3 (master) or byte 4 (slave): 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		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	93	93
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 55
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 55: Byte 3 (master) or byte 4 (slave): 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			

Weekly timer: ⏲1 – ⏲8

The following commands are used to read the logic state of the individual weekly timers.

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**88**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0[1)]
0	1	Len	01	01
1	2	Part no.	90	90
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 56
4 – 6	5 – 7	Data 2 – 4	00	00

1) Possible causes → page 144

Table 56: Byte 3 (master) or byte 4 (slave): 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} → ⏲3 is active.

Read/write function block data

→ Please also observe the relevant description of the function blocks provided in the easy700 manual (MN05013003Z-EN, previous description AWB2528-1508GB) or in the easySoft 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	127
C1 – C16	„Counter relays: C1 – C16“	Read/Write	8F	130
O1 – O4	„Operating hours counters: O1 – O4“	Read/Write	92	133
T1 – T16	„Timing relays: T1 – T16“	Read/Write	8E	135
Y1 – Y8	„Year time switch: Y1 – Y8“	Read/Write	A2	138
⌚1 – ⌚8	"Weekly timer: ⌚1 – ⌚8“	Read/Write	A1	141

Analog value comparator/threshold comparator: A1 – A16

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Part no.	8D	8D
1	2	Instance[2]	00 – 0F	00 – 0F
2	3	Index	→ table 57	→ table 57
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 58	depending on index, → table 58

1) Possible causes → page 144

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

Table 57: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 58		×	
01	Control byte → table 59		×	
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 to a constant in the program.

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

Table 58: 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 59: 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**		**Master**	**Slave**
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Part no.	8F	8F
1	2	Instance[2]	00 – 0F	00 – 0F
2	3	Index	→ table 60	→ table 60
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 61	depending on index, → table 61

1) Possible causes → page 144

2) easy provides 16 counters C1 to C16 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	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 the data bytes Data 1 – Data 2. Be aware that the Low byte is kept 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
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 62: Index 01 – Control byte

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

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

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

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Command: Read	89	–
	0	Response: read successful	–	C2
0	1	Part no.	8F	8F
1	2	Instance	02	02
2	3	Index	02	02
3	4	Data1	00	12
4	5	Data 2	00	03
5	6	Data 3	00	00
6	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 Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1]
0	1	Part no.	92	92
1	2	Instance[2]	00 – 03	00 – 03
2	3	Index	→ table 63	→ table 63
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 64	depending on index, → table 64

1) Possible causes → page 144
2) easy provides 4 operating hours counters O1 to O4. These can be addressed using the instance (0 – 3).

Table 63: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 64		×	
01	Control byte → table 65		×	
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 the data bytes Data 1 – Data 4. Be aware that the Low byte is kept in Data 1 and the High byte in Data 4.

Table 64: 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 65: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	–	RE3)	EN2)	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		Master	Slave
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Part no.	8E	8E
1	2	Instance[2]	00 – 0F	00 – 0F
2	3	Index	→ table 66	→ table 66
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 67	depending on index, → table 67

1) Possible causes → page 144

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

Table 66: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 67		×	
01	Control byte → table 68		×	
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 the data bytes Data 1 – Data 2. Be aware that the Low byte is kept in Data 1 and the High byte in Data 2.

Table 67: 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	
Pulse 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 68: 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	Part no.	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	16.12 s
Second	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		Master	Slave
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Part no.	A2	A2
1	2	Instance[2]	00 – 07	00 – 07
2	3	Index	→ table 69	→ table 69
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 70	depending on index, → table 70

1) Possible causes → page 144

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

Table 69: Operand overview

Index (hex)	Operand	Read	Write
00	Parameters → table 70	×	
01	Control byte → table 71	×	
	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) In the data bytes Data 1 – Data 3 the switching points are transferred.

Table 70: 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 for the year time switch of channels A and B appear in the parameter menu.

Table 71: Index 01 – Control byte

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

1) Status 1 if count condition is fulfilled.

Channel A, index 11/12

Index 0x11 channel A timepoint of switch on
Index 0x12 channel A timepoint of switch off
Data 1 (Byte 4) – day
Data 2 (Byte 5) – month
Data 3 (Byte 6) – year

Example:
The year time switch channel A is required to activate on 21.04.2004.

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

The year time switch channel B is required to activate on 05.11.2012.

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

Weekly timer: Ö1 – Ö8

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**89**	–
		Write	**8D**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0[1)]
0	1	Part no.	A1	A1
1	2	Instance[2)]	00 – 07	00 – 07
2	3	Index	→ table 72	→ table 72
3 – 6	4 – 7	Data 1 – 4	depending on index, → table 73	depending on index, → table 73

1) Possible causes → page 144

2) easy provides 8 seven-day time switches Ö1 to Ö8 use as required. These can be addressed using the instance (0 – 7).

Table 72: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 73		×	
01	Control byte → table 74		×	
11	Channel A	Day on/off	×	c[1]
12		Time on	×	c[1]
13		Time off	×	c[1]
21	Channel B	Day on/off	×	c[1]
22		Time on	×	c[1]
23		Time off	×	c[1]
31	Channel C	Day on/off	×	c[1]
32		Time on	×	c[1]
33		Time off	×	c[1]
41	Channel D	Day on/off	×	c[1]
42		Time on	×	c[1]
43		Time off	×	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. Be aware that the Low byte is kept in Data 1 and the High byte in Data 2.

Table 73: 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

Significance:
The values for the weekly timer WH... of channels A and B appear in the parameter menu.

Table 74: Index 01 – Control byte

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

1) Status 1 if count condition is fulfilled.

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

If the channel is not used the 16 bit value is equal to 0x00.

Index 0x12 – time on (2 bytes)
Index 0x13 – time off (2 bytes)
Data 1 (Byte 4) – hour
Data 2 (Byte 5) – minute

Example: time on at 13:43
Data 1 = 0x0D
Data 2 = 0x2B

Analysis – error codes via easyLink

The easy700 basic device 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	Part no.	00
2	Instance	00
3	Index	00
4	Failure code	→ table 75

Table 75: Error codes

Failure code	Description
0x01	An unknown telegram has been sent.
0x02	An unknown object has been sent.
0x03	An unknown command has been sent.
0x04	An invalid instance has been sent.
0x05	An invalid parameter set has been used.
0x06	An attempt has been made to write a variable which is not a constant.
0x0C	The device is in an invalid device mode. STOP → RUN or RUN → STOP
0x0D	An invalid display access occurs. Please exit the menu level to allow the status display to be shown on the display. Writing to the clock is not possible.
0xF0	An attempt has been made to control an unknown parameter.
0xF1	Invalid value

8 easy800/MFD control commands

Data exchange procedure

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

- Read/write date and time (page 148)
- Read/write image data (page 154)
- Read/write function block data (page 174)

For this the message transfer protocol of the explicit messages is accessed in the master controller. The parameters are addressed via the service code 32_{hex}. The assigned attribute ID is here used to distinguish between different parameters and functions.

Service code	Object address	
	Class ID	Instance ID
32_{hex}	64_{hex}	01_{hex}

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

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

A data exchange procedure is required in order to ensure the safe exchange of data via DeviceNet from master to slave and vice versa.

→ The operating mode of the basic unit must correspond with the status indicated at the LEDs when the various parameters are being set.

In the communication between the stations the master initiates the data exchange with a control command. The slave always gives a response to the request. The response provides information whether the data exchange was executed or not. An error code is returned if the data exchange could not be executed. This is defined exactly by the ODVA.

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 01	From 04	From 05	From 07
Support for complete PDO access				
R data writable	✓	✓	✓	✓
S data readable	✓	✓	✓	✓
Function blocks				
Function Blocks	–	Read		
	–	–	–	DG, JC, MX, PO, SP, SR, TB
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 (MN04902001Z-EN; previous description AWB2528-1423GB).

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**93**	–
		Write	**B3**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	05	05
1	2	Index	00	00
2 – 6	3 – 7	Data 1 – 5		
		Read operation	00	→ table 76
		month Write operation	→ table 76	00

Table 76: Byte 2 to 6 (master) or Byte 3 to 7 (slave): Data 1 to 5

Byte		Contents	Operand		Value (hex)
Master	Slave				
2	3	Data 1	Hour	0 up to 23	00 – 17
3	4	Data 2	Minute	0 up to 59	00 – 3B
4	5	Data 3	Day	Day (1 to 28; 29, 30, 31 ; depending on month and year)	01 – 1F
5	6	Data 4	Month	1 up to 12	01 – 0C
6	7	Data 5	Year	0 to 99 (corresponds to 2000-2099)	00 – 63

Winter/summer time, DST

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**93**	–
		Write	**B3**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	05	05
1	2	Index		
		01: Summer/Winter time	→ table 77	→ table 77
		02: Winter time (to the “Area” = rule”)[1]	→ table 78	→ table 78
2 – 6	3 – 7	Data 1 – 5		
		Read operation	00	depending on index, → table 77, 78
		month Write operation	depending on index, → table 77, 78	00

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

Table 77: Index 01 – Summer/Winter time switchover

Byte		Contents		Value (hex)
Master	Slave			
2	3	Data 1	Area	
			None	00
			Manual	01
			Automatic EU	02
			Automatic GB	03
			Automatic US	04
			Rule[1)]	05
for “Area” = “manual”:				
3	4	Data 2	Set summer time day (1 to 28, 29, 30, 31 depending on month and year)	00 – 3B
4	5	Data 3	Set Summer time month (1 to 12)	01 – 1F
5	6	Data 4	Set winter time day (1 to 28, 29, 30, 31 depending on month and year)	01 – 0C
6	7	Data 5	Set Winter time month (1 to 12)	00 – 63
for “Area” = “Rule”[1]:				
3 – 6	4 – 7	Data 2 – 5	Summer time switching rule	→ table 79

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

Table 78: Index 02 – Winter time (only valid if Area = “Rule” selected)

Byte		Contents		Value (hex)
Master	Slave			
2	3	Data 1	Area = Rule	01
3 – 6	4 – 7	Data 2 – 5	Winter time switching rule	→ table 79

Switching rule bit array

→ Please also read the detailed description in the easy800 manual (MN04902001Z-EN; previous description AWB2528-1508GB). The following table shows the composition of the corresponding data bytes.

Table 79: Switching rule bit array

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

Example

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

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Write	**B3**	–
	0	Response: Write successful	–	C1
0	1	Len	05	05
1	2	Index	00	00
2	3	Data 1 (hex)	0E	00
3	4	Data 2 (minute)	24	00
4	5	Data 3 (day)	17	00
5	6	Data 4 (month)	05	00
6	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 (MN04902001Z-EN, previous description AWB2528-1508GB) or in the easySoft Help.

The information provided in Section "General information on working with image data" on page 61 also applies to easy700.

Overview

Operands	Meaning	Read/Write	Command (hex)	Page
IA1 – IA4	„Local analog inputs: IA1 – IA4"	Reading	02	155
ID1 – ID16	„Local diagnostics: ID1 – ID16"	Reading	03	157
IW0	„Read local inputs: IW0"	Reading	01	159
IW1 – IW8	„Inputs of the network station: IW1 – IW8"	Reading	01	161
M...	„Marker: M.."	Read/Write	0B – 0E	162
P1 – P4	„Local P buttons: P1 – P4"	Reading	06	165
QA1	„Local analog output: QA1"	Read/Write	05	167
QW0, QW1 – QW8	„Local outputs: QW0/ outputs of the network station: QW1 – QW8"	Read/Write	04	168
R1 – R16 S1 – S8	„Inputs/outputs of easyLink: RW/SW"	Reading	07/09	170
RN1 – RN32 SN1 – SN32	„Receive data network: RN1 – RN32/ Send data network: SN1 – SN32"	Reading	08/0A	172

Local analog inputs: IA1 – IA4

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

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	02	02
2	3	Index	01 – 04[1)]	01 – 04[1)]
3	4	Data 1 (Low Byte)	00	→ example on page 156
4	5	Data 2 (High Byte)	00	
5 – 6	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 appropriate telegrams for reading the analog value are as follows:

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**91**	–
	0	Response: Read successful	–	C2
0	1	Len	02	02
1	2	Part no.	02	02
2	3	Index	01[1)]	01[1)]
3	4	Data 1	00	D9
4	5	Data 2	00	02
5	6	Data 3	00	00
6	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) bytes 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		Master	Slave
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	03	03
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 80
4	5	Data 2 (High Byte)	00	→ table 80
5 – 6	6 – 7	Data 3 – 4	00	00

Table 80: 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

Read 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		Master	Slave
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	01	01
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 81
4	5	Data 2 (High Byte)	00	→ table 81
5 – 6	6 – 7	Data 3 – 4	00	00

Table 81: 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		Master	Slave
		Attribute ID: Read	**91**	–
	0	Response: Read successful	–	C2
0	1	Len	02	02
1	2	Part no.	01	01
2	3	Index	00	00
3	4	Data 1	00	C4
4	5	Data 2	00	02
5	6	Data 3	00	00
6	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 network station: IW1 – 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**		**Master**	**Slave**
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	01	01
2	3	Index	01 – 08[1)]	01 – 08[1)]
3	4	Data 1 (Low Byte)	00	→ table 81 on page 159.
4	5	Data 2 (High Byte)	00	
5 – 6	6 – 7	Data 3 – 4	00	00

1) Corresponds to address of network station

Marker: M..

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**91**	–
		Write	**B1**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	→ table 82	→ table 82
1	2	Part no.		
2	3	Index		
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	→ „Example 1: Set/reset market bit“ on page 164
		month Write operation	→ „Example 2: Write marker word“ on page 164	00

Table 82: Byte 0 to 2 (master) or: Byte 1 to 3 slave: Len, Type, Index

Operand			Len	Part no.	Index
Bit Marker	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 bits	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 bits	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: Set/reset market bit

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

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID: Write	B1	–
	0	Response: Write successful	–	C1
0	1	Len	01	01
1	2	Part no.	0B	0B
2	3	Index	3E	3E
3	4	Data 1	01 1)	00
4 – 6	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 marker word MW32: $823_{dec} = 337_{hex} \rightarrow$ Data 1 = 37_{hex}, Data 2 = 03_{hex}

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID: Write	B1	–
	0	Response: Write successful	–	C1
0	1	Len	02	01
1	2	Part no.	0D	0D
2	3	Index	20	20
3	4	Data 1	37	00
4	5	Data 2	03	00
5	6	Data 3	00	00
6	7	Data 4	00	00

Local P buttons: P1 – P4

The local P buttons are the display cursor buttons of the easy800/MFD basic device. 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 device).

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

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	06	06
2	3	Index	00	00
3	4	Data 1 (Low Byte)	00	→ table 83
4 – 6	5 – 7	Data 2 – 4	00	00

Table 83: 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 from easy800, device version 04) the value will only be output if the respective device is in RUN mode and if the respective image is not written by the actual program, → section "Read/write image data"on page 154.

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**91**	–
		Write[1)]	**B1**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	05	05
2	3	Index	00	00
3 – 4	4 – 5	Data 1 – 2		
		Read operation	00	→ example
		month Write operation	→ example	00
5 – 6	6 – 7	Data 3 – 4	00	00

1) Writing is only possible from easy800, version 04 → section "Read/write date and time" on page 148.

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

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

Local outputs: QW0/ outputs of the network station: QW1 – QW8

You can read the local outputs directly via the DeviceNet and also write them from easy800, Version 04. 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 154.

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Command		
		Read	**91**	–
		Write1)	**B1**	–
	0	Response		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	04	04
2	3	Index2)	00/01 – 08	00/01 – 08
3	4	Data 1		
		Read operation	00	→ table 80
		month Write operation	→ table 84	00
4 – 6	5 – 7	Data 2 – 4	00	00

1) Writing is only possible from easy800, device version 04 → section "Read/write date and time" on page 148.

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

Table 84: 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 easyLink: 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 easyLink, again from the relevant easy800/MFD image.

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**91**	–
		Response:		
	0	Read successful	–	C2
		Command rejected	–	C0
0	1	Len	02	02
1	2	Part no.	For RW: 07	For RW: 07
2			For SW: 09	For SW: 09
	3	Index	00/01 – 08[1)]	00/01 – 08[1)]
3	4	Data 1 (Low Byte)	00	→ table 85
4	5	Data 2 (High Byte)	00	→ table 85
5 – 6	6 – 7	Data 3 – 4	00	00

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

Table 85: 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/ Send 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 EASY204-DP 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		Master	Slave
		Attribute ID: Read	**91**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Len	04	04
1	2	Part no.	For RN1 – RN32: 08	For RN1 – RN32: 08
			For SN1 – SN32: 0A	For SN1 – SN32: 0A
2	3	Index	01 – 08[1)]	01 – 08[1)]
3 – 6	4 – 7	Data 1 – 4	00	→ table 86

1) Corresponds to NET-ID

Table 86: 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. If you transfer 32-bit values ensure that the appropriate value range corresponds to the long integer, i.e. is signed.
  32-bit value: –2147483648…0…+2147483647

Overview

Operands	Meaning	Read/Write	Type (hex)	Page
A01 – A32	„Analog value comparator: A01 – A32"	Read/Write	11	177
AR01 – AR32	„Arithmetic function block: AR01 – AR32"	Read/Write	12	179
BC01 – BC32	„Block Compare: BC01 – BC32"	Read/Write	25	181
BT01 – BT32	„Block Transfer: BT01 – BT32"	Read/Write	26	183
BV01 – BV32	„Boolean operation: BV01 – BV32"	Read/Write	13	185
C01 – C32	„Counter: C01 – C32"	Read/Write	14	187
CF01 – CF04	„Frequency counters: CF01 – CF04"	Read/Write	15	189
CH01 – CH04	„High-speed counter: CH01 – CH04"	Read/Write	16	191
CI01 – CI02	„Incremental encoder counters: CI01 – CI02"	Read/Write	17	193
CP01 – CP32	„Comparator: CP01 – CP32"	Read/Write	18	195
D01 – D32	„Text output function block: D01 – D32"	Read/Write	19	197
DB01 – DB32	„Data function block: DB01 – DB32"	Read/Write	1A	200
DC01 – DC32	„PID controller: DC01 – DC32"	Read/Write	27	202
DG01 – DG16	„DG01…DG16 diagnostics"	Reading	39	205
FT01 – FT32	„Signal smoothing filter: FT01 – FT32"	Read/Write	28	207
GT01 – GT32	„Receipt of network data: GT01 – GT32"	Reading	1B	209
HW01 – HW32	„Comparator: CP01 – CP32"	Reading	1C	211
HY01 – HY32	„Year time switch: HY01 – HY32"	Reading	1D	214
JC01 – JC32	„Conditional jump JC01…JC32"	Reading	2F	217
LS01 – LS32	„Value scaling: LS01 – LS32"	Read/Write	29	219
MR01 – MR32	„Master Reset: MR01 – MR32"	Reading	0F	221
MX01 – MX32	„Data Multiplexer MX01…MX32"	Read/Write	31	223
NC01 – NC32	„Numerical Converter: NC01 – NC32"	Read/Write	2A	225
OT01 – OT04	„Hours-run Counter: OT01 – OT04"	Read/Write	1E	227

Operands	Meaning	Read/Write	Type (hex)	Page
PO01 – PO02	„Pulse width modulation: PW01 – PW02"	Read/Write	32	229
PT01 – PT32	„Value scaling function blocks LS01 .. LS32"	Reading	1F	232
PW01 – PW02	„Pulse width modulation: PW01 – PW02"	Read/Write	2B	234
SC01	„Synchronize Clock: SC01"	Reading	20	236
SP01 - SP32	„Serial output SP01…SP32"	Reading	35	237
SR01 - SR32	„Sending of network data: PT01 – PT32"	Reading	33	239
ST01	„Set cycle time: ST01"	Read/Write	2C	242
T01 – T32	„Timing relays: T01 – T32"	Read/Write	21	244
TB01 – TB32	„Value limitation: VC01 – VC32"	Read/Write	34	247
VC01 – VC32	„Value limitation: VC01 – VC32"	Read/Write	2D	249

Analog value comparator: A01 – A32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	11	11
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 87	→ table 87
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 88, 89

Table 87: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 88		×	
01	Mode, → table 89		×	
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 the 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 88: 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 89: Index 1 - Mode

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

Arithmetic function block: AR01 – AR32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	12	12
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 90	→ table 90
3 – 6	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		Reading	Writing
00	Bit IO, → table 91		×	
01	Mode, → table 92		×	
02	First operand	I1	×	c1)
03	Second operand	I2	×	c1)
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 91: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1
FB output Data 3		–	–	–	–	–	ZE1	CY2)

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 92: 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 Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	25	25
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 93	→ table 93
3 – 6	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 ing	Writing
00	Bit IO, → table 94		×	
01	Mode, → table 95		×	
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 94: 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 95: 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 Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	26	26
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 96	→ table 96
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 97, 98
		Write operation	depending on index, → table 97, 98	00

Table 96: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 97		×	
01	Mode, → table 98		×	
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 (Data 1 – Low Byte to Data 2 – High Byte).

Table 97: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T1
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 98: Index 1 - Mode

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

Boolean operation: BV01 – BV32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	13	13
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 99	→ table 99
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 100, 101
		Write operation	depending on index, → table 100, 101	00

Table 99: Operand overview

Index (hex)	Operand		Read ing	Writing
00	Bit IO, → table 100		×	
01	Mode, → table 101		×	
02	First operand	I1	×	c[1)]
03	Second operand	I2	×	c[1)]
04	Result of the operation	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 100: 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 101: 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.

Counter: C01 – C32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	14	14
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 102	→ table 102
3 – 6	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	Reading	Writing
00	Bit IO		→ table 103	×	
01	Mode/Parameter		–	–	–
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		–	–	–	–	SE[1]	D[2]	C[3]	RE[4]
FB output Data 3		–	–	–	–	ZE[5]	CY[6]	FB[7]	OF[8]

1) With a rising edge transfer the preset actual value
2) Count direction: 0 = up counting, 1 = down counting
3) Count coil, counts on every rising edge
4) Reset the 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 Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	15	15
1	2	Instance	01 – 04	01 – 04
2	3	Index	→ table 104	→ table 104
3 – 6	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		Reading	Writing
00	Bit IO, → table 105		×	
01	Mode/Parameter		–	–
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 105: 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) Enable for counter function block
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 counter: CH01 – CH04

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	16	16
1	2	Instance	01 – 04	01 – 04
2	3	Index	→ table 106	→ table 106
3 – 6	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		Value	Reading	Writing
00	Bit IO		→ table 107	×	
01	Mode/Parameter		–	–	–
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 107: 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) Enable for counter function block
2) With a rising edge transfer the preset actual value
3) Count direction: 0 = up counting, 1 = down counting
4) Reset the 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 encoder counters: CI01 – CI02

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	17	17
1	2	Instance	01 – 02	01 – 02
2	3	Index	→ table 108	→ table 108
3 – 6	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		Value	Reading	Writing
00	Bit IO		→ table 109	×	
01	Mode/Parameter		–	–	–
02	Upper setpoint	SH	In integer range from –2 147 483 648 to +2 147 483 647	×	c1)
03	Lower setpoint	SL		×	c1)
04	Preset actual value	SV		×	c1)
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 109: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	EN1)	SE2)	RE3)
FB output Data 3		–	–	–	–	ZE4)	CY5)	FB6)	OF7)

1) Enable for counter function block
2) With a rising edge transfer the preset actual value
3) Reset the 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

Comparator: CP01 – CP32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	18	18
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 110	→ table 110
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 111
		Write operation	depending on index, → table 111	00

Table 110: Operand overview

Index (hex)	Operand	Reading	Writing
00	Bit IO, → table 111	×	
01	Mode/Parameter	–	–
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 111: 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 block: D01 – D32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	19	19
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 112	→ table 112
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index,→ table 113
		Write operation	depending on index,→ table 113	00

Table 112: Operand overview

Index (hex)	Operand	Reading	Writing
00	Bit IO, → table 113	×	
01	Mode/Parameter	–	–
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	Reading	Writing
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 113: 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 block: DB01 – DB32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	1A	1A
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 114	→ table 114
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 115
		Write operation	depending on index, → table 115	00

Table 114: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 115		×	
01	Mode/Parameter		–	–
02	Input value: value that is transferred to the QV output when the FB is triggered.	I1	×	c1)
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 115: Index 0 – Bit IO

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

1) Transfer of the value present at I1 when there is a rising edge.

2) Status 1 if the trigger signal is 1.

PID controller: DC01 – DC32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	27	27
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 116	→ table 116
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 117, 118
		Write operation	depending on index, → table 117, 118	

Table 116: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 117		×	
01	Mode, → table 118		×	
02	Setpoint: –32768 to +32767	I1	×	c[1)]
03	Actual value: –32768 to +32767	I2	×	c[1)]
04	Proportional Gain [%], Value range: 0 to 65535	KP	×	c[1)]
05	Reset time [0.1 s], Value range: 0 to 65535	TN	×	c[1)]
06	Rate time [0.1 s], Value range: 0 to 65535	TV	×	c[1)]
07	Scan time = Time between function block calls Value range: 0.1s to 6553.5s If 0 is entered as the value, the scan time will be determined by the program cycle time.	TC	×	c[1)]
08	Manual manipulated variable, value range: –4096 to +4095	MV	×	c[1)]
09	Manipulated variable • Mode: UNI, value range: 0 to +4095 (12 bit) • Mode: BIP, value range: –4096 to +4095 (13 bit)	QV	×	

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

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

Table 117: 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 118: 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

DG01…DG16 diagnostics

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	39	39
1	2	Instance	01 - 10	01 - 10
2	3	Index	00 - 03	00 - 03
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 119, 120

Table 119: Operand overview

Index (hex)	Data	Data 1, Data 3, Data 4	Data 2	Read/Write
0	Bit IO	→ table 120	–	R
2	Diagnostics register QV	ST 1)		R
3	Output states ON	ST 1)		R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

Table 120: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN1)
FB output Data 3		Q82)	Q72)	Q62)	Q52)	Q42)	Q32)	Q22)	Q12)
FB output Data 4		–	–	–	–	–	–	–	QC3)

1) Reset coil: Status 1 resets the counter actual value to zero.
2) 1 is set if the selected safety function block has the selected state.
3) 1 is set if one of the outputs Q1 to Q8 is 1.

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Signal smoothing filter: FT01 – FT32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	28	28
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 121	→ table 121
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 122
		Write operation	depending on index, → table 122	00

Table 121: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 122		×	
01	Mode/Parameter		–	–
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 up 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 122: 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.

Receipt of network data: GT01 – GT32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**92**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Part no.	1B	1B
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 123	→ table 123
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 124, 125

Table 123: Operand overview

Index (hex)	Operand	Read ing	Writing
00	Bit IO, → table 124	×	
01	Mode/Parameters, → table 125	×	–
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 124: 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 125: 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 sending the value. Possible station numbers: 01 to 08
2) Send FB (e.g. PT 20) of the sending NET station. Possible station numbers: 01 – 32

Comparator: CP01 – CP32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID: Read	**92**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Part no.	1C	1C
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 126	→ table 126
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 127

Table 126: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO	→ table 127	×	
01	Mode/Parameter		–	–
02	Parameters	→ table 128	×	
	Channel A			
03	Channel B			
04	Channel C			
05	Channel D			

Table 127: 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 128: 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	d4	d3	d2	d1	d0	h4	h3	h2	h1	h0	m5	m4	m3	m2	m1	m0
	Day of week					**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
	Day of week					**Hour**					**Minute**					

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

Example

The channel A parameters of HW19 weekly timer are to be read.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	**Attribute ID: Read**	**92**	–
	Response: Read successful	–	C2
1	Part no.	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
	Day of week					**Hour**					**Minute**					

Switch-on time:
Weekday = 01_{hex} ... Monday
Hour = $0D_{hex}$…13 pm
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
	Day of week					**Hour**					**Minute**					

Switch-off time:
Weekday = 04_{hex} .. Thursday
Hour = 15_{hex}…21 pm
Minute = 59_{hex} .. 34 minutes

Year time switch: HY01 – HY32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID: Read	**92**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Part no.	1D	1D
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 129	→ table 129
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 130

Table 129: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO	→ table 130	×	
01	Mode/Parameter		–	–
02	Parameters	→ table 131	×	
	Channel A			
03	Channel B			
04	Channel C			
05	Channel D			

Table 130: 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 131: 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	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 = $0E_{hex}$ = 000**0 1110**$_{bin}$
Month = 6 (June) = 06_{hex} = 0000 **0110**$_{bin}$
Year = 20**03** = 03_{hex} = 0**000 001**1$_{bin}$

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} = 000**0 0011**$_{bin}$
Month = 10 (October) = $0A_{hex}$ = 0000 **1010**$_{bin}$
Year = 20**12** = $0C_{hex}$ = 0**000 1100**$_{bin}$

Resulting telegram:

Byte	Meaning	Value (hex), sent by	
		Master	**Slave**
0	Attribute ID: Write	B2	–
	Response: Write successful	–	C1
1	Part no.	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

Conditional jump JC01…JC32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	2F	2F
1	2	Instance	01 - 20	01 - 20
2	3	Index	00	00
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 132, 133
		Date 1 - 4 Write operation	depending on index, → table 132, 133	00

Table 132: Operand overview

Index (hex)	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
0	Bit IO	→ table 133	–	→ table 133	–	R

Table 133: 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) When 1, the program branches to the associated jump label.
2) 1 is set if the associated jump label was not found.

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Value scaling: LS01 – LS32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	29	29
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 134	→ table 134
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 135
		Write operation	depending on index, → table 135	

Table 134: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 135		×	
01	Mode/Parameter		–	–
02	Input value, value range: 32 bit	I1	×	c[1)]
03	Interpolation point 1, X co-ordinate, value range: 32 bit	X1	×	c[1)]
04	Interpolation point 1, Y co-ordinate, value range: 32 bit	Y1	×	c[1)]
05	Interpolation point 2, X co-ordinate, value range: 32 bit	X2	×	c[1)]
06	Interpolation point 2, Y co-ordinate, 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 135: 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		Master	Slave
		Attribute ID: Read	**92**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Part no.	0F	0F
1	2	Instance	01 – 20	01 – 20
2	3	Index		
		Bit IO	00	00
		mode	01	01
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 136, 137

Table 136: 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 137: Index 1 - Mode

Data 1 (hex)		
00	Q	The outputs Q…, *Q…, S…, *S…, *SN…, QA01 are reset to 0. * according to the NET-ID
01	M	The marker range MD01 to MD48 is reset to 0.
02	ALL	Reset of Q and M.

Data Multiplexer MX01…MX32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	31	31
1	2	Instance	01 - 20	01 - 20
2	3	Index	00 – 0B	00 – 0B
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 138, 139
		Date 1 - 4 Write operation	depending on index, → table 138, 139	00

Table 138: Operand overview

Index (hex)	Data	Data 1 Data 3	Data 2 Data 4	Read/Write
0	Bit IO	→ table 139	–	R
2	Channel selection: 0 up to 7	ST[1]		R/W[2]
3	Input value channel 1	ST[1]		R/W[2]
4	Input value channel 2	ST[1]		R/W[2]
5	Input value channel 3	ST[1]		R/W[2]
6	Input value channel 4	ST[1]		R/W[2]
7	Input value channel 5	ST[1]		R/W[2]
8	Input value channel 6	ST[1]		R/W[2]
9	Input value channel 7	ST[1]		R/W[2]
CSA	Input value channel 8	ST[1]		R/W[2]
B	Output value QV	ST[1]		R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

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

Table 139: 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) When 1 is set, the selected input value is entered in the output value.

2) 1 is set if the channel selection is invalid.

Numerical Converter: NC01 – NC32

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	2A	2A
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 140	→ table 140
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 141, 142
		Write operation	depending on index, → table 141, 142	00

Table 140: Operand overview

Index (hex)	Operand	Read ing	Writing
00	Bit IO, → table 141	×	
01	Mode, → table 142	×	
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 to Data 2 – High Byte).

Table 141: 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 142: 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

Hours-run Counter: OT01 – OT04

→ Further information is provided in the S40 Application Note AN27K21d.exe EASY800/MFD-DP Data Handling Function Blocks for PS416 and PS4-341.

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	1E	1E
1	2	Instance	01 – 04	01 – 04
2	3	Index	→ table 143	→ table 143
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 144
		Write operation	depending on index, → table 144	00

Table 143: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 144		×	
01	Mode/Parameter		–	–
02	Command rejected	I1	×	c[1)]
03	Actual value of the operating hours counter	QV	×	

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

Table 144: 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 is reached (greater/equal).

→ 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).

Pulse width modulation: PW01 – PW02

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	32	32
1	2	Instance	01 - 02	01 - 02
2	3	Index	00 - 0A	00 - 0A
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 145, 146
		Date 1 - 4 Write operation	depending on index, → table 145, 146	00

Table 145: Operand overview

Index (hex)	Data	Data 1 Data 3	Data 2 Data 4	Read/ Write
0	Bit IO	→ table 146	–	R
2	Pulse count in positioning mode I1: 0 to 2147483647	ST1)		R/W2)
3	Start frequency FS: 0 bis 5000 Hz	ST1)		R/W2)
4	Operating frequency FO: 0 bis 5000 Hz	ST1)		R/W2)
5	Frequency change in acceleration ramp RF: 0 to 65535 mHz	ST1)		R/W2)
6	Frequency change in brake ramp BF: 0 to 65535 mHz	ST1)		R/W2)
7	Number of steps in jog mode P1: 0 up to 65535	ST1)		R/W2)
8	Frequency in jog mode PF: 0 up to 5000 Hz	ST1)		R/W2)
9	Actual step number QV	ST1)		R
A	Actual frequency QF	ST1)		R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

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

Table 146: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	TP[1)]	BR[2)]	ST[3)]	EN[4)]
FB output Data 3		–	–	–	–	–	–	E1[5)]	AC[6)]

1) Jog mode is started with a rising edge.
2) The positioning job is aborted with a rising edge.
3) The positioning job is started with a rising edge.
4) Reset coil: Status 1 resets the counter actual value to zero.
5) 1 is set if the parameter entry is invalid.
6) 1 is set if a positioning job is active.

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Value scaling function blocks LS01 .. LS32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	1F	1F
1	2	Instance	01 - 20	01 - 20
2	3	Index	00 - 02	00 - 02
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 147, 148
		Date 1 - 4 Write operation	depending on index, → table 147, 148	00

Table 147: Operand overview

Index (hex)	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
0	Bit IO	→ table 148	–	→ table 148	–	R
2	Setpoint QV for the network	ST1)				R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

Table 148: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	T[1)]
FB output Data 3		–	–	–	–	–	E1[2)]	AC[3)]	Q1[4)]

1) Trigger coil. If the coil is triggered (receives a rising edge), the corresponding value is put on the NET.
2) 1 is set if the send job was aborted due to an error.
3) 1 is set if the trigger coil is triggered. 0 is set if the send job was successfully completed or aborted due to an error.
4) Status 1 if the status of the trigger coil is also 1.

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN; previous description AWB2528-1423GB) or in the easySoft Help.

Pulse width modulation: PW01 – PW02

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	2B	2B
1	2	Instance	01 – 02	01 – 02
2	3	Index	→ table 149	→ table 149
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 150
		Write operation	depending on index, → table 150	00

Table 149: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 150		×	
01	Mode/Parameter		–	–
02	Manipulated variable, value range: 0 to 4095 (12 Bit)	SV	×	c[1]
03	Period duration [ms], Value range: 0 up to 65535	PD	×	c[1]
04	Minimum on duration [ms], Value range: 0 up to 65535	ME	×	c[1]

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

Table 150: 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) 1 is set if the value is below the minimum on time or minimum off time

Synchronize Clock: SC01

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID: Read	**92**	–
	0	Response:		
		Read successful	–	C2
		Command rejected	–	C0
0	1	Part no.	20	20
1	2	Instance	01	01
2	3	Index	→ table 151	→ table 151
3 – 6	4 – 7	Data 1 – 4	00	depending on index, → table 152

Table 151: Operand overview

Index (hex)	Operand	Reading	Writing
00	Bit IO, → table 152	×	
01	Mode/Parameter	–	–

Table 152: 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 with a rising edge, the current date, weekday and time of the transmitting station is automatically sent to the NET network.

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

Serial output SP01…SP32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	Slave		Master	Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	35	35
1	2	Instance	01 - 20	01 - 20
2	3	Index	00	00
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 153, 154

Table 153: Operand overview

Index (hex)	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
0	Bit IO	→ table 154	–	→ table 154	–	R

Table 154: Index 0 – Bit IO

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

1) The send job is triggered on a rising edge.
2) Reset coil: Status 1 resets the counter actual value to zero.
3) 1 is set if an error occurred during the send job.
4) 1 is set if the send job is active.

Sending of network data: PT01 – PT32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	33	33
1	2	Instance	01 - 20	01 - 20
2	3	Index	00 – 0B	00 – 0B
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 155, 156
		Date 1 - 4 Write operation	depending on index, → table 155, 156	00

Table 155: Operand overview

Index (hex)	Data	Data 1	Data 2 Data 4	Data 3	Read/ Write
0	Bit IO	→ table 156	–	→ table 156	R
1	mode	→ table 157	–	–	R
2	Data input forwards I1	ST[1]			R/W[2]
3	Data input backwards 2	ST[1]			R/W[2]
4	Data output 1 (D1)	ST[1]			R
5	Data output 2 (D2)	ST[1]			R
6	Data output 3 (D3)	ST[1]			R
7	Data output 4 (D4)	ST[1]			R
8	Data output 5 (D5)	ST[1]			R
9	Data output 6 (D6)	ST[1]			R
CSA	Data output 7 (D7)	ST[1]			R
B	Data output 8 (D8)	ST[1]			R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

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

Table 156: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	BD 1)	FD 2)	RE 3)	BP 4)	FP 5)	EN 6)
FB output Data 3		Q8 7)	Q8 7)	Q6 7)	Q5 7)	Q4 7)	Q3 7)	Q2 7)	Q1 7)

1) Input bit value for the backward shift operation in BIT mode.
2) Input bit value for the forward shift operation in BIT mode.
3) If 1 is set, the function block is reset.
4) On receipt of a rising edge in BIT mode, the value of BD is entered in the last register field Q8 and the original contents of the register fields are moved one field in the direction of the lower field numbers. On receipt of a rising edge in DW mode, the value of I2 is entered in the last register field D8 and the original contents of the register fields are moved by one field in the direction of the lower field numbers.
5) On receipt of a rising edge in BIT mode, the value of FD is entered in the first register field Q1 and the original contents of the register fields are moved one field in the direction of the higher field numbers. On receipt of a rising edge in DW mode, the value of I1 is entered in the first register field D1 and the original contents of the register fields are moved by one field in the direction of the higher field numbers.
6) Reset coil: Status 1 resets the counter actual value to zero.
7) Status of the eight fields of the bit shift register.

Table 157: : Index 1 – Mode

Data 1 (hex)		
00	BIT	Mode: shift bit
01	DW	Mode: shift double word

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Set cycle time: ST01

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID:		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	2C	2C
1	2	Instance	01	01
2	3	Index	→ table 158	→ table 158
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 159
		Write operation	depending on index, → table 159	00

Table 158: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 159		×	
01	Mode/Parameter		–	–
02	Cycle time in ms, value range: 0 – 1000	I1	×	c1)

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

Table 159: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB output Data 3		–	–	–	–	–	–	–	EN1)

1) Activates the function block on status 1.

Timing relays: T01 – T32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	21	21
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 160	→ table 160
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 161, 162
		Write operation	depending on index, → table 161, 162	

Table 160: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 161		×	
01	Mode/Parameters, → table 162		×	
02	Setpoint value 1: Time setpoint 1	I1	×	c 1)
03	Setpoint value 2: Time setpoint 2 (with timing relay with 2 setpoint values):	I2	×	c 1)
04	Actual value: Timed-out actual time 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 161: 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) Switching contact

Table 162: Index 1 – Mode/Parameter

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	Pulse transmitter
	7	Flashing relay (two time setpoints)
	8	Off-delayed, retriggerable (easy600 mode)
	9	Off-delayed, with random set value, retriggerable (easy600 mode)
Parameters	**Data 3**	**Operating Mode**
	0	S (Milliseconds)
	1	M:S (Seconds)
	2	H:M (Minutes)

Value limitation: VC01 – VC32

Telegram structure

Byte Master	Byte Slave	Meaning	Value (hex), sent by Master	Value (hex), sent by Slave
		Attribute ID Read	92	–
		Response:	B2	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	34	34
1	2	Instance	01 - 20	01 - 20
2	3	Index	00 - 04	00 - 04
3 - 6	4 - 7	Data 1 - 4 Read operation	00	depending on index, → table 163, 164
		Date 1 - 4 Write operation	depending on index, → table 163, 164	00

Table 163: Operand overview

Index (hex)	Data	Data 1 Data 3	Data 2 Data 4	Read/ Write
0	Bit IO	→ table 164	–	R
2	Input value I1 for table of TB…	ST[1]		R/W[2)]
3	Output value QV from table of TB…	ST[1]		R
4	Number of entries QN in table of TB…	ST[1]		R

1) Value transferred in Intel format: Data 1 contains Low byte, Data 4 contains High byte

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

Table 164: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	RE[1)]	RL[2)]	RF[3)]	WP[4)]	EN[5)]
FB output Data 3		–	–	–	–	–	–	TF[6)]	TE[7)]

1) On receipt of a rising edge, all entries are removed from the table. The number of table entries QN is set to 0.
2) On receipt of a rising edge the newest entry in the table is output at output QV and removed from the table.
The number of table entries QN is decremented by one.
3) On receipt of a rising edge the oldest entry in the table is output at output QV and removed from the table. The number of table entries QN is decremented by one.
4) On receipt of a rising edge, the value of I1 is transferred to the table and the number of table entries is incremented by one, as long as the maximum number of entries is not exceeded.
In this case, the value of I1 is output at the output QV.
5) Reset coil: Status 1 resets the counter actual value to zero.
6) 1 is set if the table is full.
7) 1 is set if the table is empty.

→ Further information on this module is provided in the easy800 manual (MN04902001Z-EN, previous description AWB2528-1423GB) or in the easySoft Help.

Value limitation: VC01 – VC32

Telegram structure

Byte		Meaning	Value (hex), sent by	
Master	**Slave**		**Master**	**Slave**
		Attribute ID		
		Read	**92**	–
		Write	**B2**	–
	0	Response:		
		Read successful	–	C2
		Write successful	–	C1
		Command rejected	–	C0
0	1	Part no.	2D	2D
1	2	Instance	01 – 20	01 – 20
2	3	Index	→ table 165	→ table 165
3 – 6	4 – 7	Data 1 – 4		
		Read operation	00	depending on index, → table 166
		Write operation	depending on index, → table 166	00

Table 165: Operand overview

Index (hex)	Operand		Reading	Writing
00	Bit IO, → table 166		×	
01	Mode/Parameter		–	–
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 166: 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 easyLink

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	Part no.	
2	Instance	
3	Index	
4	Failure code	→ table 167
5 – 7	Data 2 – 4	

Table 167: Error codes

Error code	Description
0x00	No error
0x03	formal fault in the response relating to type, instance or index
0x04	no communication possible (timeout)
0x05	DP module has only sent 0xC0 (Easy800 Basic II, MFD version I).
0x45	the value selected by the type and index may not be written (bit IO, mode/parameter or output value).
0x46	the value selected by the type and index is not assigned with a constant.
0x9E	access to the FB data not possible (program download active).
0x9F	type is invalid (no defined FB, also dependant on the version of the addressed device).
0xA0	FB selected by type and instance does not exist in program.
0xA1	index relative to the defined FB type is invalid

9 What happens if...?

Module status LED MS	**Possible cause**	**Remedy**
Off	No power at EASY222-DN.	Switch on the power supply.
Green	EASY222-DN is in standby mode.	None
Green flashing	EASY222-DN not configured.	Verify the correct setting of the MAC ID.
Red flashing	Invalid configuration	Check configuration data.
Red	Module error which can not be resolved.	Replace the EASY222-DN.

Network Status LED NS	**Possible cause**	**Remedy**
Off	• EASY222-DN without power or • communication is blocked at this channel because – of bus-off state or – power loss or – the channel was blocked explicitly.	• Switch on the EASY222-DN, • supply the mains voltage to the channel and • ensure that the channel is active.
Green	Although the channel is enabled, communication is not possible.	Check the communication function at the master PLC.
Green flashing	Normal operation	None
Red flashing	Communication error or the EASY222-DN may be defective.	Reset the module. If further errors occur, replace the EASY222-DN.
Red	Communication error.	Check the master PLC.

Appendix

Technical data

General		
Standards		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 ambient air temperatures (Cold to IEC 60068-2-1, Heat to IEC 60068-2-2)		
Operating ambient temperature Installed horizontally/vertically	°C	–25 to +55
Condensation		Prevent condensation by means of suit-able measures
Storage/transport temperature	°C	–40 to +70
Relative humidity (IEC 60068-2-30), no moisture condensation	%	5 to 95
Air pressure (in operation)	hPa	795 up 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 up to 57
Constant acceleration, 2 g	Hz	57 up 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, 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 (IEC/EN 61000-4-3)	V/m	10
Radio interference suppression (EN 55011, EN 55022), class		B
Fast transient burst (IEC/EN 61000-4-4, severity level 3)		
Supply cables	kV	2
Signal cables	kV	2
High energy pulses (Surge) easy-AC (IEC/EN 61000-4-5), power cable symmetrical	kV	1
High-energy pulses (surge) of "easy" DC current (IEC/EN 61 000-4-5, severity level 2), power cable symmetrical	kV	0.5
Immunity to line-conducted interference to (IEC/EN 61000-4-6)	V	10
Insulation resistance		
Clearance in air and creepage distances		EN 50178, UL508, CSA C22.2 No. 142
Insulation resistance		EN 50178
Tools and cable cross-sections		
Conductor cross-sections		
Solid, minimum to maximum	mm^2	0.2 up to 4
	AWG	22 up to 12
Flexible with ferrule, minimum to maximum	mm^2	0.2 up to 2.5
	AWG	22 up to 12
Slot-head screwdriver, width	mm	3.5 × 0.8
Tightening torque	Nm	0.5

Current supply		
Rated voltage		
Nominal value	V DC	24 (–15, +20)
Permissible range	V DC	20.4 to 28.8
Ripple	%	< 5
Input current at 24 V DC, typical	mA	200
Voltage dips (IEC/EN 61131-2)	ms	10
Power loss at 24 V DC, normally	W	4.8
LED indicators		
Module Status LED MS	Color	green/red
Network Status LED NS	Color	green/red
DeviceNet		
Device connection		5-pole socket
Potential isolation		Bus to power supply (simple) Bus and power supply to easy basic unit (safety isolation)
Function		DeviceNetSlave
Interface		DeviceNet (CAN)
Bus protocol		DeviceNet
Baud rate, automatic detection up to	kBd	500
Bus Terminating Resistors		Separate installation at the bus possible
Bus addresses, accessible via easy basic unit with display or easySoft		0 up to 63
Services		
Module inputs		all data S1 to S8 (easy600)
Module outputs		all data R1 to R16 (easy600)
Module control commands		Read/Write Weekday, time-of-day, summer/winter time All parameters of the easy functions

Dimensions

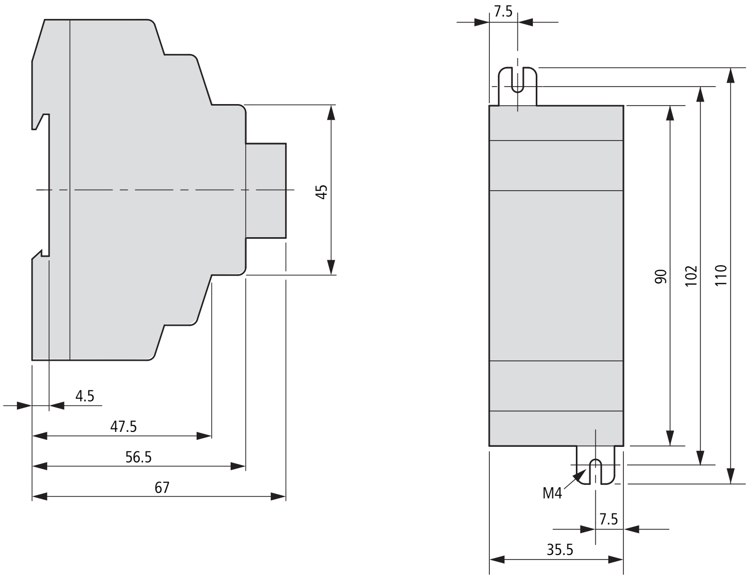


Figure 13: EASY222-DN dimensions in [mm]

EDS file

```
$**********************************************************
$ Moeller GmbH
$ Device: EASY222-DN
$ Version: V1.0
$ Date: 27.05.02
$ Author: Ronny Happ
$ Description: EDS file for easy DeviceNet slave module
$ Modifications:
$
$ Copyright (c) 2002  by Moeller GmbH
$**********************************************************

[File]
$ File Description Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.1

    DescText     = "Moeller DeviceNet Coupler easy 222-DN";
    CreateDate   = 27-05-2002;
    CreateTime   = 17:00:00;
    ModDate      = 25-06-2002;
    ModTime      = 11:00:00;
    Revision     = 1.0;

[Device]
$ Device Description Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.2

    VendCode     = 248;                     $ Identity Object - Vendor ID
    ProdType     = 12;                      $ Identity Object - Device Type
    ProdCode     = 650;                     $ Identity Object - Product Code
    MajRev       = 1;                       $ Identity Object - Major Revision
    MinRev       = 1;                       $ Identity Object - Minor Revision
```

```
                                          $ Identity Object - Product Name
   ProdName     = "EASY 222-DN";
   VendName     = "Moeller ElectroniX";
   ProdTypeStr  = "Generic";
   Catalog      = "Moeller HPL order no. 233540";

[IO_Info]
$ I/O Characteristics Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.3

   Default      = 0x000D;                 $ Cyclic, Change of State and Poll

   PollInfo     =
                0x000D,                   $ Poll (OK to combine with Cyclic or COS)
                2,                        $ Default input = Input 2
                1                         $ Default output = Output 1
   COSInfo      =
                0x000D,                   $ COS (OK to combine with Poll)
                1                         $ Default input = Input 1
                2;                        $ Default output = Output 2
   CyclicInfo   =
                0x000D,                   $ Cyclic (OK to combine with Poll)
                1,                        $ Default input = Input 1
                2;                        $ Default output = Output 2
$ Input Connections
   Input1       =
                2,                        $ 2 bytes are transferred
                16,                       $ all bits are significant
                0x0004,                   $ COS only
                "Diagnostic Data from easy", $ Name
                6, "20 04 24 64 30 03",   $ Assembly Object Instance 100,
                                          $ Attribute 3
                "";                       $ Help
```

```
    Input2       =
                 3,                          $ 3 bytes are transferred
                 24,                         $ all bits are significant
                 0x0001,                     $ Poll only
                 "Input Data from easy",     $ Name
                 6, "20 04 24 65 30 03",     $ Assembly Object Instance 101,
                                             $ Attribute 3
                 "";                         $ Help
$ Output Connections
    Output1      =
                 3,                          $ 3 bytes are transferred
                 24,                         $ all bits are significant
                 0x0001,                     $ Poll and COS
                 "Output Data to easy",      $ Name
                 6, "20 04 24 66 30 03",     $ Assembly Object Instance 102,
                                             $ Attribute 3
                 "";                         $ Help
    Output2      =
                 0,                          $ 0 byte is transferred
                 0,                          $ all bits are significant
                 0x0004,                     $ Poll and COS
                 "Acknowledge Handler",      $ Name
                 6, "20 2B 24 01 30 00",     $ Acknowledge Handler
                 "Acknowledge Handler";      $ Help

[ParamClass]
$ Parameter Class Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.4 and Chapter 6-14.1

    MaxInst      = 0;                        $ no parameters are supported
    Descriptor   = 0;                        $
    CfgAssembly  = 0;                        $ not used here
```

```
[Params]
$ Parameter Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.5 and Chapter 6-14.2



[EnumPar]
$ Parameter Enumerated String Section:
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.6



[Groups]
$ Parameter Groups Section:
$   Not used here
$   For more information about the meaning of each entry, please check
$   DeviceNet Specification Volume II Chapter 4-3.5.7

$ End of File
```

→ Note on the EDS file:

The Identity Object entry - Major Revision defines the current operating system state of the EASY222-DN communication module. As the device with a newer operating system version can deviate from the EDS description in this point, this entry must be modified accordingly, → section "Identity object" on page 35.

Glossary

This glossary refers to topics related to DeviceNet.

Terminal resistor	Terminating resistor at the start and end of a bus cable. Prevents interference due to signal reflection and is used for the adaptation of bus cables. Bus terminating resistors must always be the last unit at the end of a bus segment.
Acknowledge	Acknowledgement returned by the receiving station after having received a signal.
Address	Number that identifies a memory area, systems or module within a network, for example.
Addressing	Assignment or setting of an address for a module in the network, for example.
Active metallic component	Conductor or conductive component that is live when in operation.
Analogue	Value, such as voltage, that is infinitely variable and proportional. Analogue signals can acquire any value within specific limits.
Automation product	I/O controlling device that is interconnected to a system process. PLCs represent a special group of automation products.
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.
Electrical equipment	Comprises all equipment used for the generation, conversion, transfer, distribution and application of electrical energy, e.g. power lines, cables, machines, controlgear.
Reference ground	Earth potential in the area of grounding devices. May have a potential other than the zero of "earth" potential.
Reference potential	Represents a reference point for measuring and/or visualising the voltage of any connected electrical circuits.
Bidirectional	Operation in both directions.

Bit	Abbreviation for the English “binary digit”. Represents the smallest information unit of a binary system. Its significance can be 1 or 0 (Yes/No decision).
Lightning protection	Represents all measures for preventing system damage due to overvoltage caused by lightning strike.
Bus	Bus system for data exchange, for example between the CPU, memory and I/O. A bus can consist of several parallel segments, e.g the data bus, address bus, control bus and power supply bus.
Bus line	Smallest unit connected to the bus. Consists of the PLC, a module and a bus interface for the module.
Bus system	All units as a whole which communicate across a bus.
Bus cycle time	Time interval in which a master provides services to all slaves or nodes of a bus system, i.e. writes data to their outputs and reads inputs.
Byte	A sequence of 8 bits
Code	Data transfer format
COS I/O connection	COS (Change Of State) I/O connections are used to establish event-controlled connections. This means that the DeviceNet devices generate messages from themselves as soon as a status change is present.
	2 byte diagnostics data of the easy control relay
	Coupling module status
CPU	Abbreviation for “Central Processing Unit”. Central unit for data processing, which represents the core element of a computer.
Cyclic I/O connection	Message triggering is timer-controlled when operating with a cyclic I/O connection.
Device Heartbeat Message	A DeviceNet unit can use the Device Heartbeat Message function to broadcast its native status at set time intervals. These messages are configured in the Identity Object.
Device Shut Down Message	A device shutting down due to internal errors or states can log off at the PLC by means of the Device Shut Down Message.

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."
Dual Code	Natural binary code. Frequently used code for absolute measurement systems.
EDS	This EDS file primarily defines the Polled I/O Connection, the COS I/O Connection and the Cyclic I/O Connection of the gateway. It does not contain data or parameters (easy object) for functions of the easy basic unit. These functions are accessed by means of explicit messages.
EEPROM	Abbreviation for "Electrically Erasable Programmable Read-only Memory".
EMC	Abbreviation for "Electromagnetic Compatibility". Defines the ability of electrical equipment to operate error-free and without causing a negative influence within a certain environment.
EN	Abbreviation for "European Norm".
Earth	In electrical engineering, the term given to conductive ground with the electrical potential of zero at any point. In the environment of grounding devices, the electrical potential may not equal zero, in which case it is called the "reference earth".
Earthing	Represents the connection of an electrically conductive component to the equipotential earth via a grounding device.
Earth electrode	One or several components with direct and good contact to earth.
ESD	Abbreviation for "Electrostatic Discharge".
Fieldbus	Data network on the sensor/actuator level. The fieldbus interconnects the devices at field level. Characteristic feature of the fieldbus is the highly reliable transfer of signals and real-time response.
Field power supply	Power supply for the field devices and signal voltage.

Galvanic coupling	Galvanic coupling generally develops between two circuits using a common cable. Typical interference sources are starting motors, static discharge, clocked devices and potential difference between the component enclosure and their common power supply.
GND	Abbreviation for "GROUND" (0 potential).
hexadecimal	Numerical system with the base 16. The count starts at 0 to 9 a continues with the letters A, B, C, D, E and F.
I/O	Abbreviation for "Input/Output".
Impedance	Alternating current-resistance of a component or of a circuit consisting of several components at a specific frequency.
Low-impedance connection	Connection with low alternating-current resistance.
Inactive metallic parts	Touch-protected conductive components, isolated electrically from active metallic parts by means of an insulation, but subject to fault-voltage.
Inductive coupling	Inductive (magnetic) coupling develops between two current-carrying conductors. The magnetic effect generated by the currents induces an interference voltage. Typical interference sources are, for example transformers, motors, mains cables installed parallel and RF signal cables.
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.
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).
Configure	Systematic arrangement of the I/O modules of a station.
short-circuit proof	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.

LSB	Abbreviation for “Least Significant Bit“. Bit with the least significant value.
Common	All interconnected inactive equipment parts which are not subject to hazardous fault voltage.
Ground strap	Flexible conductor, mostly braided. Interconnects inactive parts of equipment, e.g. the doors of a control panel and the switch cabinet body.
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 actuators are special types of two-wire actuators. They are highly resistant to interference and reliable due to their special construction, e.g. low internal resistance, few components and short design.
Offline Connection Set	The Offline Connection Set allows communication with a device that is in communication error state but not in bus-off state due to an ambiguous address. It is usually no longer possible to address this device on the network, and it must be initialized manually by switching it off and on. The Offline Connection Set can be used in this situation to address such a device on the network.
Overhead	System management time. Required once for each data transfer cycle.

Parameter Definition	Definition of parameters for individual bus stations or their modules in the configuration software of the DeviceNet master.
Polled I/O connection	A polled I/O connection establishes a conventional master/slave relationship between a controller and a DeviceNet device. A polled I/O connection is a point-to-point connection between two stations on the fieldbus. The master (client) sends a poll request to the slave (server) and this replies with a poll response. • 3 bytes of output data S1 to S8 easy/MFD output range, RUN/STOP (inputs at the DeviceNet master) • 3 bytes of input data R1 to R16 easy/MFD input range, RUN/STOP (outputs of the DeviceNet master)
Potential equalization	Adaptation of the electrical level of the body of electrical equipment and auxiliary conductive bodies by means of an electrical connection.
Potential-free	Galvanic isolation between the reference potentials of the control and load circuit of I/O modules.
Common potential	Electrical interconnection of the reference potentials of the control and load circuit of I/O modules.
Response time	In a bus system this represents the time interval between the transmission of a read request and receiving the answer. Within an input module, it represents the time interval between the signal change at an input and its output to the bus system.
Repeater	Amplifier for signals transferred across a bus.
Shield	Term that describes the conductive covering of cables, cubicles and cabinets.
Screen earth kit	Refers to all measures and equipment used to connect system parts to the screen.

Protective conductor	Conductor required for human body protection against hazardous currents. Abbreviation: PE (“Protective Earth”).
Serial	Describes an information transfer technique. Data are transferred in a bit-stream across the cables.
Slave	Station or node in a bus system that is subordinate to the master.
PLC	Abbreviation for Programmable Logic Controller.
Station	Function unit or module, consisting of several elements.
Noise emission (EMC)	Testing procedure to EN 61000-6-4
Noise immunity (EMC)	Testing procedure to EN 61000-6-2
Radiation 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.
Topology	Geometrical network structure, or circuit arrangement.
UART	Abbreviation for “Universal Asynchronous Receiver/Transmitter”. A "UART" represents a logical circuit used to convert an asynchronous serial data stream into a parallel bit stream and vice versa.
UCMM	The DeviceNet gateway provides an option of configuring dynamic connection objects via the UCMM port (Unconnected Message Manager Port).
Unidirectional	Operating in one direction.

Alphabetical index