

EASY204-DP

PROFIBUS-DP

Slave Interface

Operating manual
03/05 AWB2528-1401GB

MOELLER 

Think future. Switch to green.

All brand and product names are trademarks or registered trademarks of the owner concerned.

1st published 2001, edition date 04/01

2nd edition 05/2003

3rd edition 07/2004

See revision protocol in the "About this manual" chapter

© Moeller GmbH, 53105 Bonn

Authors: Ronny Happ, Dieter Bauerfeind

Editor: Thomas Kracht

Translator: Terence Osborn

All rights reserved, including those of the translation.

No part of this manual may be reproduced in any form (printed, photocopy, microfilm or any other process) or processed, duplicated or distributed by means of electronic systems without written permission of Moeller GmbH, Bonn.



Warning!

Dangerous electrical voltage!

Before commencing the installation

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

- Measures should be taken to ensure the proper restart of programs interrupted after a voltage dip or failure. This should not cause dangerous operating states even for a short time. If necessary, emergency-stop devices should be implemented.
- Wherever faults in the automation system may cause damage to persons or property, external measures must be implemented to ensure a safe operating state in the event of a fault or malfunction (for example, by means of separate limit switches, mechanical interlocks etc.).

Contents

About This Manual	11
List of revisions	11
Target readership	11
Other manuals on the device	11
Device designation	12
Abbreviations	14
Writing conventions	14
1 EASY204-DP	15
System overview	16
Device setup	17
Device function description	18
– easy600/700/800, MFD-CP8..	18
– easy800/MFD-CP8..	19
Hardware and operating system requirements	19
List of modifications	20
Improper use	20
2 Installation	21
Connecting EASY204-DP to the basic unit	21
Connecting the power supply	22
Connecting PROFIBUS-DP	23
PROFIBUS-DP connection assignment	23
Bus terminating resistors	24
EMC wiring	24
Electrical isolation	25
Transfer rates – automatic baud rate detection	25
Maximum distances/bus cable lengths	26
3 Device Operation	29
Initial power up	29
Setting the PROFIBUS-DP station address	30
– Setting the address on the basic unit with a display	30
– Setting the address using EASY-SOFT	32

Status LEDs	33
– POW LED, Function	33
– BUS LED, Function	33
Cycle time of EASY basic unit	34
4 PROFIBUS-DP Functions	35
Configuration of the DP master, class 1	35
Slave modules	36
Diagnostics data	39
– Format of diagnostics information	39
– Meaning of diagnostics information	42
GSD file	44
Application modules	45
PROFIBUS certification	46
5 Inputs/Outputs, easy600/700/800/MFD	
Operating Mode	47
“Inputs 3 bytes” module: operating mode, S1 – S8	48
“Inputs 1 byte” module: S1 – S8	50
“Outputs 3 bytes” module: operating mode, R9 - R16, R1 – R8	50
“Outputs 1 byte” module: R1 – R8	54
6 Additional Input/Output Data easy800/MFD	57
7 Control Commands for easy600 (DPV0)	59
Data exchange procedure	59
Date and time, Summer/winter time	61
Read/write	
image	64
– General notes on working with image data	64
– Overview	64
– Read status of markers, digital outputs and text display markers	65
– Read status of P buttons and operating buttons	68

– Read status of timing relays, counter relays, time switches and analog value comparators	69
Read/write function blocks	72
– Overview	72
– Write analog value comparators (function, comparator values)	73
– Read counter relay actual value	76
– Write counter relay setpoint	78
– Read analog and digital inputs (I7, I8, I1 to I16)	81
– Read timing relay actual value (time base, actual value, switch function)	84
– Write timing relay actual value (time base, setpoint, switch function)	87
– Read time switch (channel, ON time, OFF time)	94
– Write time switch (channel, ON time, OFF time)	98
8 Control Commands for easy700 (DPV0)	101
Data exchange procedure	101
Read/write date and time	104
Read/write image data	108
– Overview	109
– Analog value comparators/threshold comparators: A1 – A16	110
– Counters: C1 – C16	111
– Text function blocks: D1 – D16	112
– Local inputs: I1 – I16	113
– Local analog inputs: IA1 – IA4	115
– Markers: M1 – M16/N1 – N16	117
– Markers: M1 – M16/N1 – N16	119
– Operating hours counters: O1 – O4	121
– Local P buttons: P1 – P4	122
– Local outputs: Q1 – Q8	124
– Inputs/outputs of EASY-LINK: R1 – R16/S1 – S8	125
– Timing relays: T1 – T16	127
– Year time switch: Y1 – Y8	128
– Master reset: Z1 – Z3	129

– 7-day time switch: Θ 1 – Θ 8	130
Read/write function block data	131
– General notes	131
– Overview	131
– Analog value comparator/threshold comparator: A1 – A16	132
– Counter relays: C1 – C16	135
– Operating hours counters: O1 – O4	138
– Timing relays: T1 – T16	140
– Year time switch: Y1 – Y8	144
– 7-day time switch: Θ 1 – Θ 8	147
Analysis – error codes via EASY-LINK	150

9 Control Commands for easy800/MFD

(DPV0)	153
Data exchange procedure	153
Version history	155
Read/write date and time	156
– Winter/summer time, DST	157
Read/write image data	160
– Overview	160
– Read local inputs IW0	161
– Read inputs of the stations IW1 to IW8	163
– Read local analog inputs IA1 to IA4	164
– Read local diagnostics ID1 to ID16	166
– Read and write local QW0 outputs/outputs of the stations QW1 to QW8	168
– Read and write local analog output QA1	170
– Read local P buttons	171
– Read RW.. inputs/SW.. outputs from EasyLink	173
– Read receive data network RN1 .. RN32/send data network SN1 .. SN32	175
– Read and write markers	177
Read/write function block data	181
– General notes	181
– Overview	182
– Analog value comparators A01 .. A32	183
– Arithmetic function blocks AR01 .. AR32	185
– Block compare function blocks BC01 .. BC32	187
– Block transfer function blocks BT01 .. BT32	189

– Boolean operation function blocks BV01 ..	191
BV32	
– Counters C01 .. C32	193
– Frequency counters CF01 .. CF04	195
– High-speed counters CH01 .. CH04	197
– Incremental encoder counters CI01 .. CI02	199
– Comparators CP01 .. CP32	201
– Text output function blocks D01 ..D32	203
– Data function blocks DB01 .. DB32	206
– PID controllers DC01 .. DC32	208
– Signal smoothing filters FT01 .. FT32	211
– Receive network data function blocks GT01 ..	
GT32	213
– 7-day time switches HW01 .. HW32	215
– Year time switches HY01 .. HY32	218
– Value scaling function blocks LS01 .. LS32	221
– Master reset function blocks MR01 .. MR32	223
– Numerical converters NC01 .. NC32	225
– Operating hours counters OT01 .. OT04	227
– Send network data function blocks PT01 ..	
PT32	229
– Pulse width modulation function blocks	
PW01 .. PW02	231
– Synchronize clock function block SC01	233
– Set cycle time function block ST01	234
– Timing relays T01 .. T32	236
– Value limitation function blocks VC01 .. VC32	239

10 Process Data for easy600 (DPV1)	241
Object overview	242
Accessing objects	244
Read identification	245
– “Easy Identification” object	245
Read/write mode	247
– “Easy operating mode” object	247
Read/write image	249
– “Easy 600 inputs” object	249
– “Easy 600 outputs and M markers” object	251
– “Easy 600 function relay” object	254
– “Easy 600 pushbuttons” object	257

– “Easy Link input data” object	259
– “Easy Link output data” object	261
Read/write function block data	263
– “Easy 600 timing relay parameters T1 to T8” objects	263
– “Easy 600 counters C1 to C8 parameters” objects	268
– “Easy 600 analog value comparator A1 to A8 parameter” objects	271
– “Easy 600 ④1 channel A to 7-day time switch ④4 channel D parameter” objects	273
Read/write date and time	276
– “Easy clock” object	276
Read/write DST	279
– Easy 600 DST setting objects	279
11 Process Data for easy700 (DPV1)	281
Object overview	282
Accessing objects	284
Read identification	285
– “Easy identification” object	285
Read/write mode	285
– “Easy mode” object	285
Read/write image	286
– “Easy 700/800 inputs” object	286
– “Easy 700/800 analog inputs IA1 to IA4” objects	288
– “Easy 700/800 outputs” object	290
– “Easy 700/800 pushbuttons” object	292
– “Easy 700 analog value comparators” object	294
– “Easy 700 7-day time switches” object	296
– “Easy 700 year time switches” objects	298
– “Easy 700 master reset” object	300
– “Easy 700 text function blocks” object	302
– “Easy 700 timing relays” objects	304
– “Easy 700 counters” object	306
– “Easy 700 operating hours counters” object	308
– “Easy 700 actual value M markers” object	310
– “Easy 700 actual value N markers” object	312
– “Easy 700 new value markers M1 to M16”	

objects	314
– “Easy 700 new value markers N1 to N16” objects	316
– “Easy Link input data” object	317
– “Easy Link output data” object	317
Read/write function block data	318
– Procedure	318
– “Easy 700/800 function block data selection” object	319
– “Easy 700/800 read function block data” object	321
– “Easy 700/800 write function block data” object	323
– Analog value comparator/threshold comparator: A1 – A16	325
– Counters C1 – C16	327
– Operating hours counters O1 – O4	329
– Timing relays T1 – T16	331
– Year time switches Y1 – Y8	334
– 7-day time switches Θ1 to Θ8	336
Read/write date and time	338
– “Easy clock” object	338
Read/write DST	338
– “Easy 700 DST setting” objects	338
12 Process Data for easy800/MFD (DPV1)	345
Object overview	346
Accessing objects	348
Read identification	349
– “Easy identification” object	349
Read/write mode	349
– “Easy mode” object	349
Read/write image	349
– “Easy 700/800 inputs” object	349
– “Easy 700/800 analog inputs IA1 to IA4” objects	349
– “Easy 700/800 outputs” objects	349
– “Easy 700/800 pushbuttons” object	349
– “Easy 800 analog output” object	350
– “Easy 800 local diagnostics” object	352

– “Easy 800 inputs network stations IW1 to IW8” objects	354
– “Easy 800 Link inputs network stations RW1 to RW8” objects	356
– “Easy 800 outputs network station SW1 to SW8” objects	358
– “Easy 800 Link outputs network station SW1 to SW8” objects	360
– “Easy 800 receive data network stations RNW1 to RNW8” objects	362
– “Easy 800 send data network SNW1 to SNW8” objects	364
– “Easy 800 bit markers M1 to M96” objects	366
– “Easy 800 byte markers MB1 to MB96” objects	368
– “Easy 800 word markers MW1 to MW96” objects	370
– “Easy 800 double word markers MD1 to MD96” objects	372
– “Easy 800 8 byte data” object	374
– “Easy 800 16 byte data” object	376
– “Easy 800 32 byte data” object	378
– “Easy 800 64 byte data” object	381
– “Easy Link input data” object	384
– “Easy Link output data” object	384
Read/write function block data	385
– Procedure	385
– Analog value comparators A1 to A32	386
– Arithmetic function blocks AR1 to AR32	388
– Block Compare function blocks BC1 to BC32	390
– Block Transfer function blocks BT1 to BT32	392
– Boolean operation function blocks BV1 to BV32	394
– Counters C1 – C32	396
– Frequency counters CF1 to CF4	398
– High-speed counters CH1 to CH4	399
– Incremental counters CI1 to CI4	401
– Comparators CP1 to CP32	403
– Text output function blocks D1 to D32	404
– Data function blocks DB1 to DB32	406

– PID controllers DC1 to DC32	407
– Signal smoothing filters FT1 to FT32	409
– Get network data function blocks GT1 to GT32	410
– 7-day time switches HW1 to HW32	412
– Year time switches HY1 – HY32	414
– Value scaling function blocks LS1 to LS32	416
– Master reset MR1 to MR32	418
– Numerical converters NC1 to NC32	420
– Operating hours counters OT1 – OT4	422
– Put network data function blocks PT1 to PT32	423
– Pulse width modulation PW1 to PW2	424
– Synchronize clock function block SC1	426
– Set cycle time function block ST1	427
– Timing relays T1 – T32	428
– Value limitation VC1 to VC32	430
Read/write date and time	431
– “Easy clock” object	431
Read/write DST	431
– “Easy 800 DST setting” objects	431
DPV1 error messages	437

Appendix	439
What Happens If ...?	439
Overview of commands	440
– easy600	440
– easy800/MFD	443
Technical Data	445
– General	445
– Ambient temperatures	445
– Ambient mechanical conditions	446
– Electromagnetic compatibility (EMC)	446
– Dielectric strength	447
– Tools and cable cross-sections	447
– Power supply	447
– LEDs	447
– PROFIBUS-DP	448
Dimensions	449

Glossary	451
-----------------	-----

Index	459
--------------	-----

About This Manual

List of revisions

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

Edition date	page/chapter	Description	New	Modification	Omitted
08/02	46	section "PROFIBUS certification"	✓		
05/03	gen.	easy800/MFD	✓		
07/04	gen.	easy700	✓		
03/05	chapter 10, 11, 12	easy600/700/800, MFD with DPV1 protocol	✓		

Target readership

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

Other manuals on the device

The following operating manuals should be followed:

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

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

Device designation

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

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

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

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

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

Abbreviations

This manual uses abbreviations with the following meaning:

hex: Hexadecimal (number system with base 16)

dec: Decimal (number system with base 10)

bcd: binary coded decimal code

VR: **Value range**

PC: **Personal Computer**

Writing conventions

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

► Indicates actions to be taken.

**Attention!**

Warns of the possibility of slight damage.

**Caution!**

Warns of the possibility of serious damage and slight injury.

**Warning**

Warns of the possibility of substantial damage, serious injury or death.



Indicates interesting tips and additional information

1 EASY204-DP

The EASY204-DP communication module was developed for automation tasks using the PROFIBUS-DP fieldbus. The EASY204-DP is a gateway and can only be used in conjunction with the easy600, easy700, easy800 or MFD basic units. The easy control relay and MFD device with a PROFIBUS-DP gateway always work as a slave station on the network.

System overview

The "easy" PROFIBUS-DP slaves are integrated in a PROFIBUS-DP system.

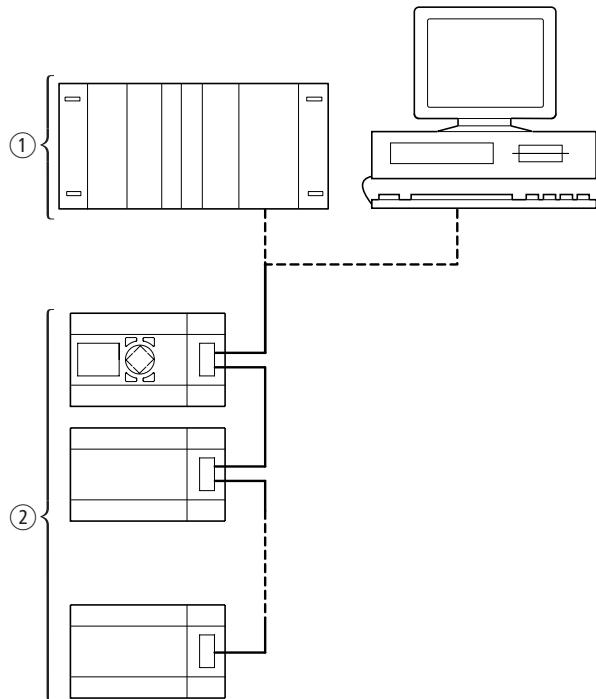


Figure 1: Integration of EASY204-DP in the DP network

- ① Master area, PLC or PC
- ② Slave area, e.g. easy /MFD with DP interface

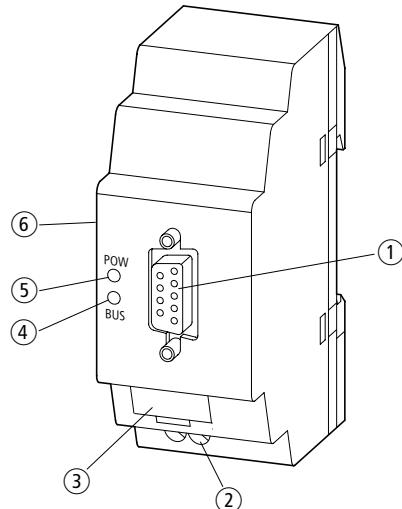
Device setup

Figure 2: View of the device

- (1) PROFIBUS-DP connection, 9-pole SUB-D socket
- (2) 24 V DC supply voltage
- (3) Device designation plate
- (4) BUS communication LED
- (5) POW operation LED
- (6) EASY-LINK socket

**Device function
description**

The EASY204-DP module allows the easy and MFD series devices to be connected to a PROFIBUS-DP communication network. The following data can be transferred by selecting the appropriate I/O modules and by using the DPV1 "Read" and "Write" services (from device version 07):

easy600/700/800, MFD-CP8..

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

easy800/MFD-CP8..

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

Hardware and operating system requirements

The EASY204-DP expansion device operates with the easy600, easy700, easy800 and MFD basic units from the following operating system versions:

Basic unit		EASY204-DP expansion device	
Device version	OS version	Device version \leq 04	Device version \leq 05
easy600			
\geq 04	From 2.4	×	×
easy700			
\geq 01	From 1.10.xxx	–	×
easy800			
\geq 04	From 1.10.xxx	–	×
MFD-CP8..			
\geq 01	From 1.10.xxx	–	×

The device version of the appropriate basic unit or expansion device is specified on the right of the housing.

Example: EASY204-DP: 03-228xxxxxx (03 = device version)

The operating system version (OS) of the corresponding basic unit can be read using EASY-SOFT. With easy700, easy800 and MFD-CP8.. devices it is also possible to read out the information directly from the device. Read the appropriate manual for further information on this.

List of modifications

Modifications from version 04 to version 05:

- Extension of the EASY-LINK protocol for connection to easy800.
- 1 byte module added to cyclical data transfer.

Modifications from version 05 to version 06:

- Extension of the EASY-LINK protocol for connection to easy700.
- Cyclical data transfer (3-byte module) adapted in Byte 0
(see also chapter "What Happens If ...?" on page 439).

Modifications from version 06 to version 07:

- Addition of DPV1 services "Read" and "Write".
- Addition of modules for additional I/O data for easy800.

Improper use

easy must not be used as a replacement for safety PLCs such as

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

2 Installation

The same installation procedures are used on easy600, easy700, easy800 and MFD basic units with expansion devices.

Connecting EASY204-DP to the basic unit

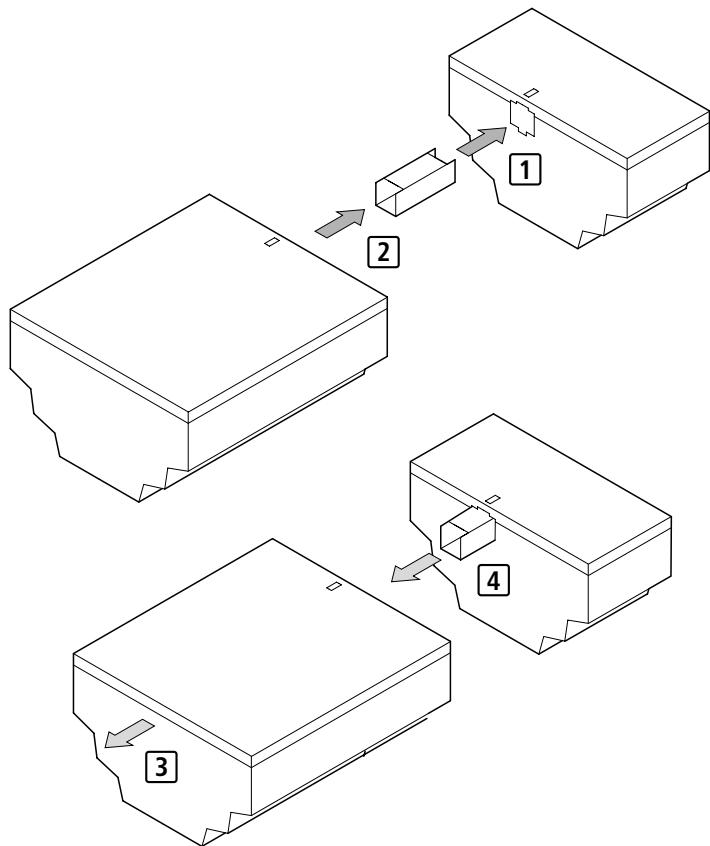


Figure 3: Fitting or removing the EASY204-DP to the basic unit [1] + [2] [3] + [4]

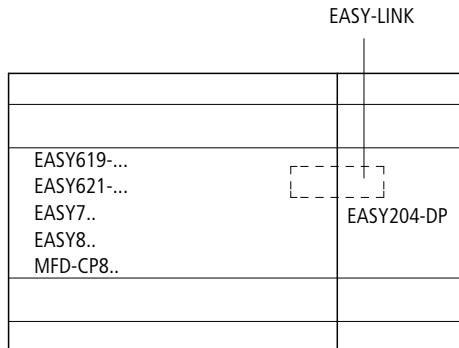


Figure 4: Connection between basic unit and EASY204-DP

Connecting the power supply

The EASY204-DP unit is run on a 24 V DC power supply
(\rightarrow section "Technical Data" from page 445).



Warning

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

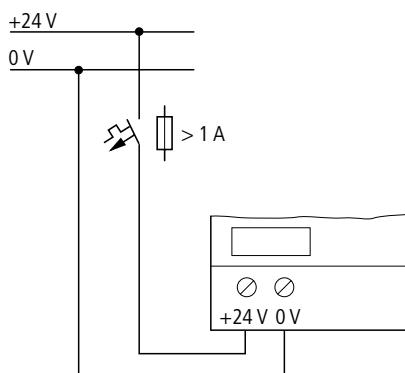
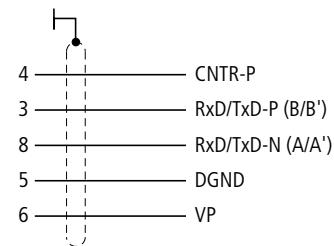
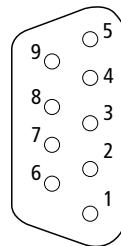


Figure 5: Standard connection

Connecting PROFIBUS-DP

Use a 9-pole SUB-D plug to connect the PROFIBUS-DP interface to the PROFIBUS-DP fieldbus. For this use the special PROFIBUS-DP plug and the special PROFIBUS-DP cable available from the Moeller range of accessories. The type of cable used determines the permissible maximum bus length and the transfer rate.

**PROFIBUS-DP connection
assignment**


Pin	Signal name	Designation
1	Not assigned	-
2	Not assigned	-
3	RxD/TxD-P (B Line)	Receive/Send data P
4	CNTR-P / RTS	Request to Send
5	DGND	Data reference potential
6	VP	+5V DC for external bus connection
7	Not assigned	-
8	RxD/TxD-N (A-Line)	Receive/Send data N
9	Not assigned	-

Connections 3, 8 and the shield are sufficient for data transfer.

Bus terminating resistors

The first and last station in a bus segment must be connected to the bus with the bus terminating resistor switched on. The bus terminating resistor is switched externally. This external switch function can either be implemented as a separate bus terminating resistor or with a special Sub-D plug with an integrated bus termination.

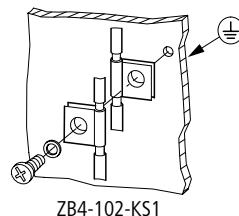
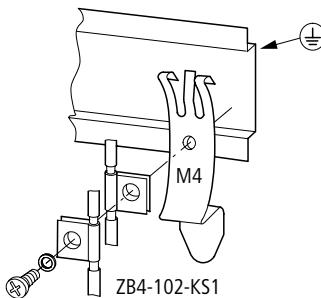
Moeller's PROFIBUS-DP data plug enables both bus terminating resistors to be switched on and off.

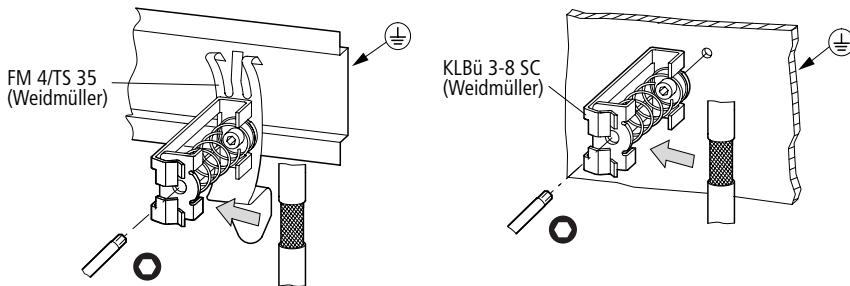
EMC wiring

Electromagnetic interference may have adverse effects on the communication fieldbus. This can be minimised in advance by taking suitable EMC measures. These include:

- System design in accordance with EMC requirements,
- EMC cabling and
- Measures that prevent the occurrence of large potential differences,
- Correct installation of the PROFIBUS system (cable, connection of bus connector,...).

The effects of electromagnetic interference can be significantly reduced by fitting the shield. The following two figures illustrate how to fit the shield.





Electrical isolation

The following electrical isolation should be provided for the interfaces of the EASY204-DP:

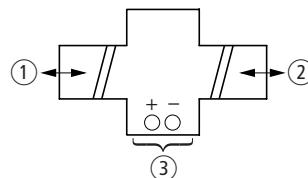


Figure 6: Potential isolation between the power supply and outputs

- ① Safe isolation of EASY-LINK 240 V AC
- ② Simple isolation of PROFIBUS-DP
- ③ 24 V DC supply voltage

Transfer rates – automatic baud rate detection

The EASY204-DP module automatically detects the baud rate used in the communication network after it is switched on. However, this requires that at least one station sends valid telegrams in the network.

The following transfer rates are supported:

- 9.6 Kbit/s to 12000 Kbit/s

Maximum distances/bus cable lengths

Two types of bus cable are specified in IEC 61158. Cable type B should no longer be used with new applications because it has been discontinued. Cable type A allows all transfer rates up to 12 000 Kbit/s to be used. Cables for burial in the ground, festoon suspension and drum cables are also available.

The cable parameters are as follows:

Parameter	Cable type A
Surge impedance in Ω	135 ... 165 at 3 ... 20 MHz
Effective capacitance (pF/m)	< 30
Loop resistance (Ω/km)	< 110
Core diameter (mm)	> 0.64
Core cross-section (mm^2)	> 0.34

The cable parameters specified allow the following bus segment lengths.

Distance between stations when using Type A cable to IEC 61158:

Baud rate [Kbit/s]	Max. cable length Type A cable [m]
9.6	1200
19.2	1200
93.75	1200
187.5	1000
500	400
1500	200
3000	100
6000	100
12000	100

Distance between two stations when using Type B cable to IEC 61158:

Baud rate [Kbit/s]	Max. cable length Type B cable [m]
9.6	1200
19.2	1200
93.75	1200
187.5	1000
500	400
1500	—

3 Device Operation

Initial power up

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

The Power LED of the EASY204-DP is lit. The BUS LED is off (no communication via PROFIBUS-DP).

The GW message (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

As soon as the device is integrated in the PROFIBUS-DP network, the BUS LED is continuously lit ("static") and the GW message is statically displayed, also on devices with a flashing GW message.



Valid data is only transferred via PROFIBUS-DP to the basic unit if the GW is displayed statically.

If the PROFIBUS-DP unit is factory set, the station address of the PROFIBUS-DP station must be set.

Setting the PROFIBUS-DP station address

Every PROFIBUS-DP station requires a unique address in the PROFIBUS-DP structure. There are two ways of setting the PROFIBUS-DP addresses on the EASY204-DP:

- Using the integrated display and keypad on the easy or MFD-Titan basic unit
- Using EASY-SOFT Version 3.01 or higher on the PC.

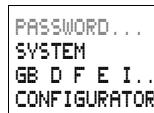
Address range: 001 to 126

Setting the address on the basic unit with a display

Requirements:

- The basic unit (easy600, easy700, easy800 or MFD-Titan) and the EASY204-DP expansion unit must be fed with power.
- The basic unit has been unlocked (no password activated).
- The basic unit has a valid operating system version (→ page 19).
- The basic unit must be in STOP mode.
- The EASY204-DP is not communicating with the PROFIBUS-DP master (Bus LED is off).

- Enter the System menu by pressing DEL + ALT simultaneously.



- Use cursor buttons \wedge or \vee to select CONFIGURATOR



- Confirm your entry with OK



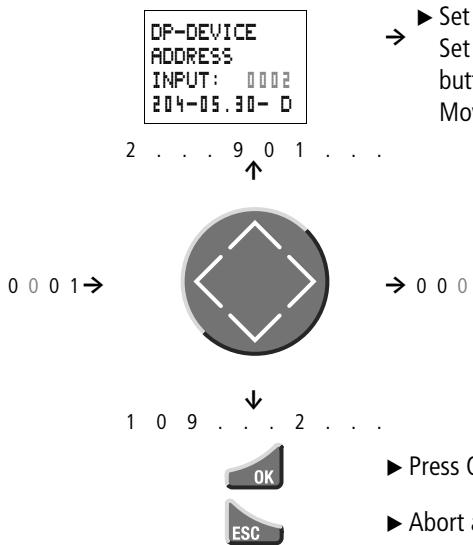
- With easy800/MFD devices select LINK...



- Confirm with OK.



easy600 and easy 700 devices show the following dialog immediately:



- ▶ Set the address:
- Set the value of the current digit with \wedge or \vee buttons.
- Move to the next digit with $<$ or $>$.

Information on the 4th display line:

xxx - xx . xx - xx
204 - 05 30 - d

Hardware version, Index: d
Software version, OS version 5.3
Device ID: EASY204-DP

Setting the address using EASY-SOFT

With EASY-SOFT, version 3.1

↳ Menu → Online → Configure Expansion Devices

From EASY-SOFT version 4.01 or higher:

Choose → Communication → Configuration → Expansion Devices → EASY204-DP.



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

Status LEDs

The EASY204-DP expansion unit has two LEDs.

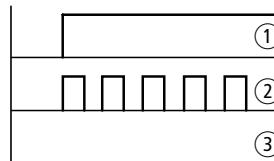
POW LED, Function

Figure 7: Function of the POW LED

- ① LED continuously lit:
 - Power supply present
 - Communication with the basic unit aborted
- ② LED flashing:
 - Power supply present
 - Communication with the basic unit correct
- ③ LED not lit:
 - No power supply present
 - Communication with the basic unit aborted

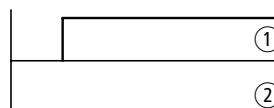
BUS LED, Function

Figure 8: Function of the BUS LED

- ① LED continuously lit:
 - PROFIBUS-DP communication correct
- ② LED not lit:
 - No PROFIBUS-DP communication present

Cycle time of EASY basic unit Communication between the basic unit and EASY204-DP via EASY-LINK increases the cycle time of the basic unit.

In extreme cases the cycle time may increase by 40 ms.

This should be taken into account for the reaction times of the basic unit.

4 PROFIBUS-DP Functions

Configuration of the DP master, class 1

The following steps are required to configure the central DP master:

- ▶ Addition of the GSD file "Moe4d10.gsd" (from device version 07) and "Moel4d10.gsd" (up to device version 06) in the GSD database of the configuration tool of your DP master.
- ▶ Addition of an EASY204-DP station to the topology of the bus segment to be configured.
- ▶ Assignment of the specified address to this station.
- ▶ Selection for this station of up to 5 of the proposed modules for cyclical data transfer, → section "Slave modules".
- ▶ Repetition of steps 2 to 4 for each of the EASY204-DP stations to be added to the topology.
- ▶ Saving of the configuration.
- ▶ Transfer of the configuration to the DP master.



Refer to the documentation of the DP master when carrying out this configuration.

Slave modules

The EASY204-DP expansion module is a PROFIBUS-DP slave in compliance with IEC 61158.

You can select the following EASY204-DP slave modules via the PROFIBUS-DP Configurator in the master PLC by using the appropriate GSD file. The modules are described in detail in the following chapters.

Module designation	Inputs (Byte)	Outputs (Byte)	Inputs/outputs (Byte)	Service	Supported devices	Code in GSD file	→ Page
Control level							
1: Easy 600 control commands	–	–	7	<ul style="list-style-type: none"> • Time • Image • Function blocks 	easy600	0xB6	59
2: Easy700/800 control commands	–	–	9	<ul style="list-style-type: none"> • Real-time clock • Image • Function blocks 	easy700, easy800, MFD-CP8..	0xB8	101
Input/output level							
3: Inputs, 3 bytes	3	–	–	<ul style="list-style-type: none"> • Read data: S1 – S8 • Mode 	easy600, easy700, easy800, MFD-CP8..	0x92	48
4: Outputs, 3 bytes	–	3	–	<ul style="list-style-type: none"> • Write data: R1 – R8, R9 – R16 • Mode 		0xA2	50
5: Inputs, 1 byte	1	–	–	<ul style="list-style-type: none"> • Read data: S1 – S8 		0x90	50
6: Outputs, 1 byte	–	1	–	<ul style="list-style-type: none"> • Write data: R1 – R8 		0xA0	54

Module designation	Inputs (Byte)	Outputs (Byte)	Inputs/outputs (Byte)	Service	Supported devices	Code in GSD file	→ Page
Additional I/O data (from device version 07)							
7: Easy 800 additional inputs, 4 bytes	4	–	–	• Read data: MD63	easy800	0x13	57
8: Easy 800 additional inputs, 8 bytes	8	–	–	• Read data: MD63 – MD64		0x17	57
9: Easy 800 additional inputs, 12 bytes	12	–	–	• Read data: MD63 – MD65		0x1B	57
10: Easy 800 additional inputs, 16 bytes	16	–	–	• Read data: MD63 – MD66		0x1F	57
11: Easy 800 additional outputs, 4 bytes	–	4	–	• Write data: MD59		0x23	58
12: Easy 800 additional outputs, 8 bytes	–	8	–	• Write data: MD59 – MD60		0x27	58
13: Easy 800 additional outputs, 12 bytes	–	12	–	• Write data: MD59 – MD61		0x2B	58
14: Easy 800 additional outputs, 16 bytes	–	16	–	• Write data: MD59 – MD62		0x2F	58
Without function							
15: Empty space ¹⁾	–	–	–	–	easy600, easy700, easy800, MFD-CP8..	0x00	–

1) The EASY204-DP provides slots in the PROFIBUS-DP configurator that you can assign with the required modules for your application. Unused slots can be assigned an "Empty Space" module.

Observe the requirements of the operating system specified on page 19!

The following rules must be observed for configuring the modules:

- Use no more than one module from the "Control level" section.
- Use no more than one input and one output module from the "I/O level" section.
- Use no more than one input and one output module (from device version 07) from the "Additional I/O data" section.

Diagnostics data

In addition to the standard DP diagnostics, the EASY204-DP provides additional diagnostics information that appears in a class 1 DPV0 master as "Device specific diagnostics" and, from device version 07, as "Status diagnostics" in a class 1 DPV1 master.

Format of diagnostics information

The diagnostics are read directly by means of the DP diagnostics commands or by means of the diagnostics bytes of the DP master defined in the PROFIBUS-DP configuration. Refer to the documentation of the master device.

On a DPV0 master, eight octets are read as diagnostics information and eleven on a DPV1 master. These contain the following data:

Table 1: Address location of the diagnostics information in a DPV0 and DPV1 master

Diagnostics position	Designation		DPV1 master
	DPV0 master		
Octet 1			
Bit 0	Non-existent station		
Bit 1	Station not ready		
Bit 2	Configuration error		
Bit 3	Additional diagnostics information		
Bit 4	Function not supported		
Bit 5	Invalid DP slave response		
Bit 6	Parameter error		
Bit 7	Master already present		
Octet 2			
Bit 0	Parameter request		
Bit 1	Static diagnostics		
Bit 2	Not used		
Bit 3	Response monitoring activated		
Bit 4	Freeze mode active		
Bit 5	Sync mode active		
Bit 6	Not used		
Bit 7	Slave deactivated		
Octet 3			
Bit 0 – 6	Not used		
Bit 7	Overflow additional diagnostics information		
Octet 4	Station address DP master		
Octet 5 and 6	Ident number of DP slave		
Octet 7	Length of additional diagnostics information		

Diagnostics position	Designation	
	DPV0 master	DPV1 master
Octet 8		Status type
Bit 0	Easy Link communication error	–
Bit 1 – 7	Not used	–
Octet 9	–	Slot Number
Octet 10	–	Status specifier
Octet 11	–	
Bit 0	–	Easy Link communication error
Bit 1	–	Invalid command for easy operating mode
Bit 2	–	Additional I/O data not yet available
Bit 3 – 7	–	Not used



When accessing the “Ident number” diagnostics data, observe the Motorola coding format used in PROFIBUS-DP (octet N: High byte, octet N+1: Low byte) for data in WORD format. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

Meaning of diagnostics information

The read diagnostics data has the following meaning:

Table 2: Data contents of the diagnostics information

Designation	Meaning	Meaning/Remedy
Response monitoring activated	The response monitoring in the EASY204-DP was activated properly.	Normal status
Freeze mode active	The DP master has activated the synchronous reading of the data inputs of several stations.	Desired user action
Function not supported	The DP master has requested a function not supported by the EASY204-DP.	Check the configuration of the DP master.
Ident number DP slave	Contains the Ident number of the EASY204-DP: 4D10 _{hex}	
Easy Link communication error	Communication between EASY204-DP and easy/MFD interrupted	Check the connection between EASY204-DP and easy/MFD.
Configuration error	The DP master has sent an invalid configuration telegram to the EASY204-DP (e.g. incorrect length of the data inputs and/or data outputs)	Check the configuration of the DP master.
Length of additional diagnostics information	Contains the length of the additional diagnostics information: 02 _{hex} with DPV0 05 _{hex} with DPV1	
Master already present	The EASY204-DP is assigned by another DP master.	
Not used	Does not contain any information to be evaluated.	
Parameter request	The EASY204-DP is waiting for the parameter telegram of the DP master.	Temporary status
Parameter error	The DP master has sent an invalid parameter telegram to the EASY204-DP.	Check the configuration of the DP master.

Designation	Meaning	Meaning/Remedy
Slave deactivated	The DP master has removed (deactivated) the EASY204-DP from its cyclical processing.	Desired user action
Slot number	Contains the slot number from which the additional diagnostics information (Status) is obtained: 00 _{hex}	
Station not ready	The EASY204-DP is not yet ready for the communication (Initialization phase).	Temporary status
Non-existent station	No station reporting under the used station address.	Check the configuration of the DP master. Check the address setting on the EASY204-DP.
Station address DP master	Contains the station address of the DP master.	
Static diagnostics	Communication between EASY204-DP and EASY/MFD interrupted.	Check the connection between EASY204-DP and EASY/MFD.
Status specifier	The EASY204-DP is not transferring an "Incoming/outgoing" message for the sent diagnostics information (Status): 00 _{hex}	
Status type	The EASY204-DP uses the status type "Status message": 01 _{hex}	
Sync mode active	The DP master has activated the synchronous output of the data outputs to several stations.	Desired user action
Overflow of additional diagnostics information	The additional diagnostics data (status) is larger than the memory reserved for it in the DP master.	Check the configuration of the DP master.
Invalid DP slave response	The EASY204-DP has sent an invalid response.	Check the cabling and EMC measures.

Designation	Meaning	Meaning/Remedy
Invalid command for easy operating mode	An impermissible bit combination was selected for easy operating mode in the cyclical output data.	Observe the information in the documentation and adapt your application program accordingly.
Additional diagnostics information	The EASY204-DP has sent additional diagnostics information (Status).	Normal status
Additional I/O data not yet available	The EASY204-DP is still in the initialization phase and has not yet received any valid additional input data from the easy800.	Temporary status during startup

GSD file

A PROFIBUS-DP GSD file is required for selecting the device and for running it on the PROFIBUS-DP communication bus. The GSD file contains standard PROFIBUS station descriptions and is contained in the Appendix of this manual. You can download the files "Moe4d10.gsd" (from device version 07) and "Moel4d10.gsd" (up to device version 06) at the following Internet addresses:

- <http://www.moeller.net> → Support →...
- <http://easy.moeller.net> → Download →...
- <ftp://ftp.moeller.net/EASY/GSD-FILES/>

Follow the links on these pages.

Application modules

All easy600 functions that are available via the EASY204-DP are supported by Moeller PLCs PS4-341, PS416, XControl and the Siemens S7 PLC. The following PLC application modules offer a convenient option for implementing the data exchange between the EASY control relay and the master PLC.

The following application modules are available:

PLC	Application module Application note	File	
		German	English
Moeller PLCs			
PS4-341 and PS416			
easy600	S40-AM-K6-D/GB	s40amk6d.exe	s40amk6g.exe
easy800/MFD	AN2700K21D/GB	an27k21d.exe	an27k21g.exe
XC PLCs			
easy600	S40-AM-K6-D/GB	xs-easydp_d.exe	xs-easydp_g.exe
Siemens PLCs			
SIMATIC S7-300			
easy600	S7-AM-K6-D/GB	s7amk6d.exe	s7amk6g.exe

Both the application modules and application notes listed in the table and other modules for the user-friendly configuration of easy800 and MFD-Titan functions can be downloaded from the following Internet address:

- [http://www.moeller.net/automation/...](http://www.moeller.net/automation/)

PROFIBUS certification

EASY204-DP was certified as a PROFIBUS-DP slave by the PROFIBUS Users' Organisation. EASY204-DP contains the PROFIBUS SPC3 interface (from device version 07) and VPC3+ (up to device version 06)



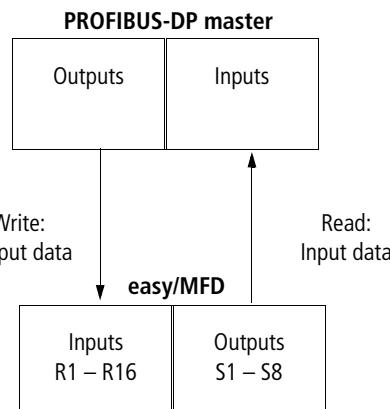
When operating EASY204-DP devices up to device version 04 uncontrolled behaviour may occur under the following circumstances:

- When Class I and Class II DP masters in a multimaster system with parameter or configuration data access the slave at the same time (highly unlikely).
- Or if other masters based on PROFIBUS layer 2 are connected.

5 Inputs/Outputs, easy600/700/800/ MFD Operating Mode

The appropriate module must be selected in the slave configuration in order for I/O data to be transferred between the EASY204-DP slave and a PROFIBUS-DP master.

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



"Inputs 3 bytes" module: operating mode, S1 – S8 The normal PROFIBUS-DP master data exchange with the EASY204-DP slave is via input data bytes 0, 1, 2.

Byte	Meaning	Value
0	Scan the operating mode	→ table 3
1	Scan status of the easy outputs S1 to S8	→ table 4
2	Not assigned	00 _{hex}

Requirement:

The "Inputs, 3 bytes" module must have been selected.



The output data and control commands can now only be used if you have selected the appropriate modules as well.

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

Table 3: Byte 0: Operating mode

"easy" operating mode	Bit	7	6	5	4	3	2	1	0	Stop/Run
with debounce		0	0	0	1	0	0	0	0	0/1
without debounce		0	0	1	0	0	0	0	0	0/1

Example:

Value 21_{hex} "easy"/MFD is in Run mode and is working with input debounce.

Table 4: Byte 1: Status of S1 to S8 on the basic unit

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

Example:

Value 19_{hex} S5, S4 and S1 are active.**Attention!**

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

- Whilst the control command is being executed, the inputs will remain in the state before the control command was called.
- After the "Control commands" data exchange has been completed, the input bytes are refreshed.

**"Inputs 1 byte" module:
S1 – S8** When this module is selected, the master only receives 1 byte (coil output data S1 to S8) via PROFIBUS.

Byte	Meaning	Value
0	Scan status of the easy outputs S1 to S8	→ table 4 on page 49

Requirement:

The "Inputs, 1 byte" module must have been selected.



The output data and control commands can now only be used if you have selected the appropriate modules as well.

**"Outputs 3 bytes"
module: operating mode,
R9 - R16, R1 – R8**

The normal PROFIBUS-DP master data exchange with the EASY204-DP slave is provided with output data bytes 0, 1, 2.

Byte	Meaning	Value
0	Set operating mode	→ table 5
1	Set/reset the easy/MFD inputs R9 to R16	→ table 6
2	Set/reset the easy/MFD inputs R1 to R8	→ table 7

Requirement:

The "Outputs; 3 bytes" module must have been selected.



The output data and control commands can now only be used if you have selected the appropriate modules as well.

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

Table 5: Byte 0: Operating mode

easy operating mode	Bit								
	7	6	5	4	3	2	1	0	
Index for setting the basic unit to the safety state	0	0	0	0	0	0	0	0	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 34_{hex} = 00110100_{bin}:

This value sets the easy/MFD status from STOP to RUN. It is only interpreted as a command and therefore does not permit an additional transfer of data. This requires the index value 14_{hex}.

Value 44_{hex} = 01000100_{bin}:

This value sets the easy/MFD status from RUN to STOP. It is also used only as a command and therefore works in the same way as the RUN command.

This should be observed without fail up to device version 05!:

Value 14_{hex} = 00010100_{bin}:

Byte 0 must always contain this value if data is to be written to the easy/MFD basic unit via the gateway.



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

From version 06 this is no longer necessary.

Table 6: Byte 1: Write status of R9 to R16

EASY6.. Input	Bit 7	6	5	4	3	2	1	0
R9								0/1
R10							0/1	
R11							0/1	
R12						0/1		
R13					0/1			
R14				0/1				
R15		0/1						
R16	0/1							

Example:

Value 19_{hex} R13, R12 and R9 should be active.

Table 7: Byte 2: Write status of R1 to R8

EASY6.. 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								

Example:

Value 2B_{hex} R6, R4, R2 and R1 should be active.



Attention!

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

- Whilst the control command is being executed, the inputs will remain in the state before the control command was called.
- After the "Control commands" data exchange has been completed, the output bytes are refreshed.

"Outputs 1 byte" module: When this module is selected, the master only sends 1 byte (coil output data S1 to S8) via PROFIBUS.
R1 – R8

Byte	Meaning	Value
0	Status of R1 to R8	→ table 7 on page 53

Requirement:

The "Outputs, 1 byte" module must have been selected.



The input data and control commands can now only be used if you have selected the appropriate modules as well.

Note on using the 1 byte modules

The 1 byte modules are not available in all device combinations. If any problems occur in handling, first check the state of the GW message in the status display of the basic unit:

GW static: The 1-byte mode can be used

GW flashing: Check the device version of the EASY204-DP and the basic unit. If these are valid, check the configuration in the PROFIBUS network and the Configurator.

Table 8: Possible device combinations for using the 1-byte mode

Basic unit	Easy204-DP
Easy600:	Version 05 or 06
Easy700:	Version 06
easy800	
Version \leq 04:	Version 05 or 06
Version \geq 05:	Version 06
MFD	
Version \leq 02:	Version 05 or 06
Version \geq 03:	Version 06

If the 1-byte mode cannot be used, use the 3-byte mode and ensure that the value = 0x14 is entered in Byte 0. Without this additional entry no valid values will be detected by the basic unit. As soon as the device is ready for operation and data communication is running, this is indicated by the static display of the GW message on the LCD of the basic unit.

6 Additional Input/Output Data easy800/MFD

From device version 07, EASY204-DP provides input/output modules for 4, 8, 12 and 16 bytes of data for cyclically transferring large data volumes with easy800/MFD connected. This data is assigned to the marker range of the easy800/MFD.



This data is transferred in contiguous bytes.

The following tables show the address location of the input and output data of these modules.

Table 9: Address location of the module input data in a DP master

	Data position	Designation
Module "EASY800 additional inputs, 16 bytes"	Octet 1	MD63 first byte (low)
	Octet 2	MD63 second byte
	Octet 3	MD63 third byte
	Octet 4	MD63 fourth byte (high)
	Octet 5	MD64 first byte (low)
	Octet 6	MD64 second byte
	Octet 7	MD64 third byte
	Octet 8	MD64 fourth byte (high)
	Octet 9	MD65 first byte (low)
	Octet 10	MD65 second byte
	Octet 11	MD65 third byte
	Octet 12	MD65 fourth byte (high)
	Octet 13	MD66 first byte (low)
	Octet 14	MD66 second byte
	Octet 15	MD66 third byte
	Octet 16	MD66 fourth byte (high)

Table 10: Address location of the module output data in a DP master

Data position	Designation
Octet 1	MD59 first byte (low)
Octet 2	MD59 second byte
Octet 3	MD59 third byte
Octet 4	MD59 fourth byte (high)
Octet 5	MD60 first byte (low)
Octet 6	MD60 second byte
Octet 7	MD60 third byte
Octet 8	MD60 fourth byte (high)
Octet 9	MD61 first byte (low)
Octet 10	MD61 second byte
Octet 11	MD61 third byte
Octet 12	MD61 fourth byte (high)
Octet 13	MD62 first byte (low)
Octet 14	MD62 second byte
Octet 15	MD62 third byte
Octet 16	MD62 fourth byte (high)

7 Control Commands for easy600 (DPV0)

Data exchange procedure

The “Control commands, 7 bytes” module allows extended data exchange of the easy600 on the PROFIBUS-DP communication bus. This allows you to transfer services from the following ranges:

- “Date and time, Summer/winter time”,
- “Read/write function blocks” and
- “Read/write image”.

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

A special command code in byte 0 is used to activate the services required. Data bytes 1 to 6 are used to write or read the values concerned.



Attention!

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



Caution!

Only those values specified for the command code should be used.

Check the values that you write in order to avoid malfunctions.

Requirement:

The “Control commands, 7 bytes” module must have been selected.



Data can only be written if the “easy” basic unit with the LCD display is showing the Status display.

The master initiates the data exchange of the control commands and the addressed slave responds.

During the communication 7 bytes of data are transferred on PROFIBUS.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
--------	--------	--------	--------	--------	--------	--------

Byte 0, Toggle byte

Byte 0 contains the Toggle bit and the command. It is used to activate the sending of a control command.

Byte 0, Toggle byte								
Bit	7	6	5	4	3	2	1	0
	0/1	x	x	x	x	x	x	x
Toggle bit	Command							

Procedure

- ▶ To send a command, bit 7 must be toggled, i.e. set either from 1 to 0 or from 0 to 1.
- ▶ Then poll the toggle bit for the coupling module's response until it has the same status as the toggle bit sent. This status indicates to the master that the response to the sent command is valid.
- ▶ Do not send a new command until you have received a response (changing of the toggle bit), otherwise the response of the previous command will be overwritten before it can be read.



In order to use input/output data and control commands simultaneously:

Only after the "Control commands" data exchange has been completed will the I/O data be refreshed.

**Date and time, Summer/
winter time Telegram structure**

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	Read	3C	—	0/1	0	1	1	1	1	0	0
	Write	2A	—	1/0	0	1	0	1	0	1	0
	Response ¹										
	Read successful	—	C2/42	1/0	1	0	0	0	0	1	0
	Write successful	—	C1/41	1/0	1	0	0	0	0	0	1
	Command rejected	—	C0/40	1/0	1	0	0	0	0	0	0
1	Weekday										
	Read operation	00	→ table 11								
	For write operation	→ table 11	00								
2	Hour										
	Read operation	00	→ table 12								
	For write operation	→ table 12	00								
3	Minute										
	Read operation	00	→ table 13								
	For write operation	→ table 13	00								
4	Summer/winter time										
	Read operation	00	→ table 14								
	For write operation	→ table 14	00								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 11: Byte 1: Weekday (value range 00 to 06)

Weekday	Bit							
	7	6	5	4	3	2	1	0
Monday = 0	0	0	0	0	0	0	0	0
Tuesday = 1	0	0	0	0	0	0	0	1
Wednesday = 2	0	0	0	0	0	0	1	0
Thursday = 3	0	0	0	0	0	0	1	1
Friday = 4	0	0	0	0	0	1	0	0
Saturday = 5	0	0	0	0	0	1	0	1
Sunday = 6	0	0	0	0	0	1	1	0

Table 12: Byte 2: Hour (value range 00 to 23)

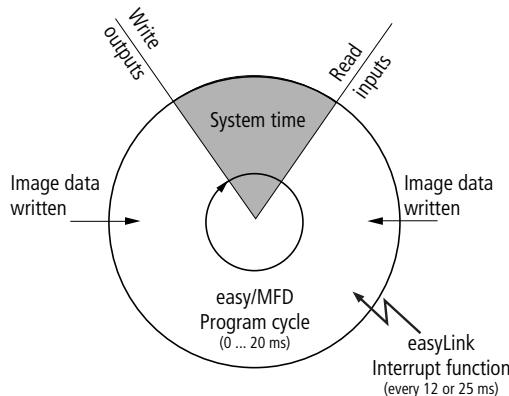
Value (bcd)	Value 10					Value 1			
	Bit	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	1
....									
9	0	0	0	0	1	0	0	0	1
....									
14	0	0	0	1	0	1	0	0	0
...									
23	0	0	1	0	0	0	1	1	1

Table 13: Byte 3: Minute (value range 00 to 59)

Value (bcd)	Value 10 Bit				Value 1 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 14: Byte 4: Winter/summer time (value range 00 to 01)

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

**Read/write
image****General notes on working with image data**

When writing to image data, it must be remembered that an image (e.g. inputs, outputs,...) used in the easy800/MFD program is also written cyclically by the actual program. The only image data that is unchanged is the data that is not used in the program and is therefore not overwritten in the program cycle. This operating principle also means that an image written via EASYLINK, such as output data is only then output at the physical outputs of the easy800/MFD when the control relay is in Run mode.

Overview

Operands	Meaning	Read/ write	Comm and	Page
M1 – M16, Q1 – Q8, D1 – D8	"Read status of markers, digital outputs and text display markers"	Read	40	65
P1 – P4, ESC/OK/DEL/ALT	"Read status of P buttons and operating buttons"	Read	3E	68
T1 – T8, C1 – C8, @1 – @4, A1 – A8,	"Read status of timing relays, counter relays, time switches and analog value comparators"	Read	3F	69

Read status of markers, digital outputs and text display markers

The following command will read the logical state of all markers M1 to M16, digital outputs Q1 to Q7, text display markers D1 to D8.

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	Read	40	—	1	0	0	0	0	0	0	0
	Response ¹	—	C2/42	1	0	0	0	0	0	1	0
	Read successful	—	C0/40	1	0	0	0	0	0	0	0
1	Command rejected	—	—	1	0	0	0	0	0	0	0
	Status of markers M1 to M8	00	→ table 15								
	Status of markers M9 to M16	00	→ table 16								
	Status of digital outputs Q1 to Q8	00	→ table 17								
4	Status of text markers D1 to D8	00	→ table 18								

- 1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 15: Byte 1: Status of markers M1 to M8

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

Table 16: Byte 2: Status of markers M9 to M16

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

Table 17: 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								

Table 18: 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								

Read status of P buttons and operating buttons

The following command is used to read the logical state of the digital pushbutton inputs P1 to P4.

The status of the pushbuttons is only displayed if

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

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	Read	3E	—	1	0	1	1	1	1	1	0
	Response ¹										
	Read successful	—	C2/42	1	0	0	0	0	0	1	0
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Status of timing relay	00	→ table 19								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 19: Byte 1: 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 pressed/pressed		0/1								
ALT not actuated/actuated	0/1									

Read status of timing relays, counter relays, time switches and analog value comparators

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

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	Read	3F	—	1	0	1	1	1	1	1	1
	Response ¹										
	Read successful	—	C2/42	1	0	0	0	0	0	1	0
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Status of timing relay	00	→ table 20								
2	Counter relay status	00	→ table 21								
3	Time switch status	00	→ table 22								
4	Analog value comparator status	00	→ table 23								

- 1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 20: 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							

Table 21: 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							

Table 22: Byte 3: Status of time switches

Bit	7	6	5	4	3	2	1	0
Θ1								0/1
Θ2								0/1
Θ3							0/1	
Θ4						0/1		
				0				
			0					
		0						
0								

Table 23: 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							

Overview

Read/write function blocks

Operands	Meaning	Command	Page
A1 – A8	"Write analog value comparators (function, comparator values)"	22 _{hex} – 29 _{hex}	73
C1 – C8	"Read counter relay actual value"	49 _{hex} – 50 _{hex}	76
	"Write counter relay setpoint"	09 _{hex} – 10 _{hex}	78
I1 – I16	"Read analog and digital inputs (I7, I8, I1 to I16)"	3D _{hex}	81
T1 – T8	"Read timing relay actual value (time base, actual value, switch function)"	41 _{hex} – 48 _{hex}	84
	"Write timing relay actual value (time base, setpoint, switch function)"	01 _{hex} – 08 _{hex}	87
T1 – T4	"Read time switch (channel, ON time, OFF time)"	2B _{hex} – 3A _{hex}	94
	"Write time switch (channel, ON time, OFF time)"	12 _{hex} – 21 _{hex}	98

**Write analog value comparators
(function, comparator values)**

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	A1	22	–	0	0	1	0	0	0	1	0
	A2	23	–	0	0	1	0	0	0	1	1
	A3	24	–	0	0	1	0	0	1	0	0
	A4	25	–	0	0	1	0	0	1	0	1
	A5	26	–	0	0	1	0	0	1	1	0
	A6	27	–	0	0	1	0	0	1	1	1
	A7	28	–	0	0	1	0	1	0	0	0
	A8	29	–	0	0	1	0	1	0	0	1
	Response ¹										
	Write successful	–	C1/41	1	/0	1	0	0	0	0	0
	Command rejected	–	C0/40	1	/0	1	0	0	0	0	0
1	Comparators	→ table 39	Invalid								
2	Comparison value for comparison with constant	→ table 40									

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

The comparison values and the function are part of an “*.eas file”. If these values are changed, the original “*.eas file” will no longer match the file in EASY6...

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

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

The comparison shows that the circuit diagrams are not identical.

Table 24: Byte 1: Control byte analog value comparator:
Comparator

Meaning	Bit	7	6	5	4	3	2	1	0
Compare: " \geq "									0
Compare: " \leq "									1
I7 with I8							0	0	
I7 with constant							0	1	
I8 with constant							1	0	
Fixed			0	0	0				
Does not appear in the Parameters menu		1							
Appears in the Parameters menu		0							
Processing	1								

Table 25: Byte 2: Comparison value for comparison with constant

Value (hex)	Bit	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0	0
63	0	1	1	0	0	0	1	1	

Example

The analog value comparator A8 has the following settings:

- Compare I7 < 4.7 V

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

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

The slave responds with the following telegram:

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

Read counter relay actual value

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	C1	49	—	1	0	1	0	0	1	0	0
	C2	4A	—	1	0	1	0	0	1	0	1
	C3	4B	—	1	0	1	0	0	1	0	1
	C4	4C	—	1	0	1	0	0	1	1	0
	C5	4D	—	1	0	1	0	0	1	1	0
	C6	4E	—	1	0	1	0	0	1	1	1
	C7	4F	—	1	0	1	0	0	1	1	1
	C8	50	—	1	0	1	0	1	0	0	0
	Response ¹										
	Read successful	—	C2/42	1	0	1	0	0	0	0	1
	Command rejected	—	C0/40	1	0	1	0	0	0	0	0
1	Invalid	00	→	x	x	x	x	x	x	x	x
2	Counter relay actual value (low byte)	00	→ table 26								
3	Counter relay actual value (high byte)	00	→ table 27								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 26: Byte 2: Counter relay actual value (low byte)

Value (hex)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	
FF	1	1	1	1	1	1	1	1	

Table 27: Byte 3: Counter relay actual value (high byte)

Value (hex)	Bit							
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
FF	1	1	1	1	1	1	1	1

Example

The master initiates the command for reading counter relay C5:

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Command: C5	4D	1	1	0	0	1	1	0	1
1 – 6		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	42	1	1	0	0	0	0	1	0
1	Invalid		x	x	x	x	x	x	x	x
2	Counter relay actual value (low byte)	67	0	1	1	0	0	1	1	1
3	Counter relay actual value (high byte)	12	0	0	0	1	0	0	1	0

Actual value of counter relay C5: 4711 (value $1267_{\text{hex}} = 4711_{\text{dec}}$)

Write counter relay setpoint**Telegram structure**

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	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
	Response ¹										
	Write successful	—	C1/41	1	0	0	0	0	0	0	1
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Parameters menu	→ table 28	00								
2	Setpoint value (low byte)	→ table 29	00								
3	Setpoint value (high byte)	→ table 30	00								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Value range of the counter values: 0000 to 9999



Keep within the value range.

The value is part of an “*.eas file”. If these values are changed, the original “*.eas file” will no longer match the file in EASY6...

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

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

The comparison shows that the circuit diagrams are not identical.

Table 28: Byte 1: Counter relay control byte

Meaning	Bit								
	7	6	5	4	3	2	1	0	
Not assigned			0	0	0	0	0	0	
Does not appear in the Parameters menu		1							
Appears in the Parameters menu		0							
Processing	1								

Table 29: Byte 2: Counter value (low byte)

Value (hex)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	
FF	1	1	1	1	1	1	1	1	

Table 30: Byte 3: Counter value (high byte)

Value (hex)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	
FF	1	1	1	1	1	1	1	1	

Example: Change counter relay setpoint

The master initiates the command to change the setpoint value of counter relay C8 to 1542:

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Command: C8	10	0	0	0	1	0	0	0	0
1	Appears in the Parameters menu	→ Bit value	1	0	0	0	0	0	0	0
2	Setpoint value (low byte)	06	0	0	0	0	0	1	1	0
3	Setpoint value (high byte)	06	0	0	0	0	0	1	1	0

0606_{hex} = 1542_{dec}

The slave responds with the following telegram:

Byte	Meaning	Slave	Bit							
			7	6	5	4	3	2	1	0
0	Response: Write successful	41	0	1	0	0	0	0	0	1
1 – 3		00								

Read analog and digital inputs (I7, I8, I1 to I16)

The following command is used to read the values of both analog inputs I7, I8 (only EASY...-DC..) and the logical states of the digital inputs I1 to I16.

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	Read	3D	—	0	0	1	1	1	1	0	1
	Response ¹										
	Read successful	—	C2/42	1	0	0	0	0	0	1	0
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Analog value of I7	00	→ table 31								
2	Analog value of I8	00	→ table 32								
3	Status of inputs I1 to I8	00	→ table 33								
4	Status of inputs I9 to I12, I15, I16	00	→ table 34								

- 1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 31: Byte 1: Analog value I7

Analog value I7(hex)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	
....	
64	0	1	1	0	0	1	0	0	

Examples: $0A_{hex} = 1\text{ V}$, $32_{hex} = 5\text{ V}$, $64_{hex} = 10\text{ V}$

Table 32: Byte 2: Analog value I8

Analog value I8 (hex)	Bit								
	7	6	5	4	3	2	1	0	
00 _{hex}	0	0	0	0	0	0	0	0	
....	
64 _{hex}	0	1	1	0	0	1	0	0	

Examples: $0A_{hex} = 1\text{ V}$, $3C_{hex} = 6\text{ V}$, $5A_{hex} = 9\text{ V}$

Table 33: Byte 3: Status of inputs I1 to I8

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

Value 0 = switched off, Value 1 = switched on

Table 34: Byte 4: Status of inputs I9 to I12, I15, 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				
I14				0					
I15			0/1						
I16		0/1							

Value 0 = switched off, Value 1 = switched on



I13 = 0, I14 = 0

If I14 = 1, EASY204-DP has disconnected from the basic unit.

I15, I16 are the short-circuit signals for EASY...-DC-.. transistor versions.

**Read timing relay actual value
(time base, actual value, switch function)**

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	T1	41	—	1	0	1	0	0	0	0	1
	T2	42	—	1	0	1	0	0	0	0	0
	T3	43	—	1	0	1	0	0	0	0	1
	T4	44	—	1	0	1	0	0	0	1	0
	T5	45	—	1	0	1	0	0	0	1	0
	T6	46	—	1	0	1	0	0	0	1	0
	T7	47	—	1	0	1	0	0	0	1	1
	T8	48	—	1	0	1	0	0	1	0	0
	Response ¹										
	Read successful	—	C2/42	1	0	1	0	0	0	0	1
	Command rejected	—	C0/40	1	0	1	0	0	0	0	0
1	Timing relay, time base, control status	00	→ table 35								
2	Time actual value (low byte)	00	→ table 36								
3	Time actual value (high byte)	00	→ table 37								

- 1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 35: Byte 1: Timing relay function, time base, control status

Meaning	Bit							
	7	6	5	4	3	2	1	0
On-delayed						0	0	0
Off-delayed						0	0	1
On-delayed with random switching						0	1	0
Off-delayed with random switching						0	1	1
Single pulse						1	0	0
Flashing						1	0	1
Time base "s"				0	0			
Time base "M:S"				0	1			
Time base "H:M"			1	0				
Not assigned		0						
Appears in the Parameters menu	0							
Does not appear in the Parameters menu	1							
Timing relay not processed by operating system	0							
Timing relay processed by operating system	1							

Table 36: Byte 2: Time actual value (low byte)

Value (hex)	Bit							
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
FF	1	1	1	1	1	1	1	1

Table 37: Byte 3: Time actual value (high byte)

Value (hex)	Bit							
	7	6	5	4	3	2	1	0
00 _{hex}	0	0	0	0	0	0	0	0
FF _{hex}	1	1	1	1	1	1	1	1

Example

The master initiates the command for reading timing relay T1:

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Command: T1	41	1	1	0	0	0	0	0	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 of set time = 0E10_{hex} = 3600

3600 s = 60:00 M:S

**Write timing relay actual value
(time base, setpoint, switch function)**

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	T1	01	—	1	0	0	0	0	0	0	1
	T2	02	—	1	0	0	0	0	0	1	0
	T3	03	—	1	0	0	0	0	0	1	1
	T4	04	—	1	0	0	0	0	1	0	0
	T5	05	—	1	0	0	0	0	1	0	1
	T6	06	—	1	0	0	0	0	1	1	0
	T7	07	—	1	0	0	0	0	1	1	1
	T8	08	—	1	0	0	0	1	0	0	0
	Response ¹										
	Write successful	—	C1/41	1	0	0	0	0	0	0	1
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Timing relay function, time base, Parameters menu	→ table 38	Invalid								
2	Time value --.xx with time base "S"	→ table 39									
3	Time value xx.-- with time base "S" or "--:xx" with time base M:S	→ table 40									
4	Time value "xx. --" with time base "M:S" or "--:xx" with time base "H:M"	→ table 41									
5	Time value "xx. --" with time base "H:M"	→ table 42									
6	Hour value in days	00	→ table 43								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.



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

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

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

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

The comparison shows that the circuit diagrams are not identical.

Value range of the time values

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



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

Table 38: Byte 1: Timing relay control byte

Meaning	Bit							
	7	6	5	4	3	2	1	0
On-delayed						0	0	0
Off-delayed						0	0	1
On-delayed with random switching						0	1	0
Off-delayed with random switching						0	1	1
Single pulse						1	0	0
Flashing						1	0	1
Time base "S"				0	0			
Time base "M:S"				0	1			
Time base "H:M"				1	0			
Not assigned			0					
Does not appear in the Parameters menu		1						
Appears in the Parameters menu		0						
Processing	1							

Table 39: Byte 2: Time value "--.xx" time base "S"

Value (bcd)	Value --.x-				Value --.x			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
05	0	0	0	0	0	1	0	1
17	0	0	0	1	0	1	1	1
42	0	1	0	0	0	0	1	0
99	1	0	0	1	1	0	0	1

Table 40: Byte 3: Time value "xx--" Time base "S"
Time value "xx--", Time base "M:S"

Value (bcd)	Value x--- "S"				Value -x--- "S"			
	Value --:x- "-S"				Value --:-x "-:S"			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
05	0	0	0	0	0	1	0	1
17	0	0	0	1	0	1	1	1
42	0	1	0	0	0	0	1	0
99	1	0	0	1	1	0	0	1

Table 41: Byte 4: Time value "xx--" Time base "M:S"
Time value "--:xx", Time base "H:M"

Value (bcd)	Value x--- "M:-"				Value -x--- "M:-"			
	Value --:x- "-:M"				Value --:-x "-:M"			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
05	0	0	0	0	0	1	0	1
17	0	0	0	1	0	1	1	1
42	0	1	0	0	0	0	1	0

Table 42: Byte 5: Time value "xx:--" Time base "H:M"

Value (bcd)	Value x:-- "H:-"				Value -x:-- "H:-"			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
05	0	0	0	0	0	1	0	1
17	0	0	0	1	0	1	1	1

Table 43: Byte 6: Time value in days

Value (bcd)	Bit				Days value			
	Bit				Bit			
	7	6	5	4	3	2	1	0
00	0	0	0	0	0	0	0	0
04	0	0	0	0	0	1	0	0

Example 1: timing relay

The master initiates the command for writing timing relay T8 with the following values:

Switch function	On-delayed
Time range	"s"
Setpoint time	50

T8 is to be assigned the time base "M:S" and the setpoint time 30 minutes, 25 seconds.

Byte	Meaning	Value	Bit							
			7	6	5	4	3	2	1	0
0	Command: T8	08 _{hex}	0	0	0	0	1	0	0	0
1	Timing relay function, time base, Parameters menu	→	1	0	0	0	1	0	0	0
2	Time value "--.xx" with time base "S"	→	0	0	0	0	0	0	0	0
3	Time value "xx--" with time base "S" or "--:xx" with time base M:S	25 _{bcd}	0	0	1	0	0	1	0	1
4	Time value "xx--" with time base "M:S" or "--:xx" with time base "H:M"	30 _{bcd}	0	0	1	1	0	0	0	0
5	Time value "xx--" with time base "H:M"	→	0	0	0	0	0	0	0	0
6	Hour value in days	→	0	0	0	0	0	0	0	0

The slave responds with the following values:

Byte	Meaning	Value	Bit							
			7	6	5	4	3	2	1	0
0	Response: Write successful	C1 _{hex}	0	1	0	0	0	0	0	1
1	Invalid	→	x	x	x	x	x	x	x	x
2 – 6	Same content as with master									

Example 2: timing relay

The master initiates the write command for timing relay T1 with the following values:

Switch function	Off-delayed
Time range	"M:S"
Setpoint time	10:30

T1 is to be assigned the time base "H:M" and the setpoint time 95 hours, 53 minutes. 95 hours = 3 days, 19 hours

Byte	Meaning	Value	Bit							
			7	6	5	4	3	2	1	0
0	Command: T1	01 _{hex}	1	0	0	0	0	0	0	1
1	Timing relay function, time base, Parameters menu	→ Bit value	1	0	0	1	0	0	0	1
2	Time value "--.xx" with time base "S"	→ Bit value	0	0	0	0	0	0	0	0
3	Time value "xx--" with time base "S" or "--:xx" with time base "M:S"	25 _{bcd}	0	0	0	0	0	0	0	0
4	Time value "xx--" with time base "M:S" or "--:xx" with time base "H:M"	53 _{bcd}	0	1	0	1	0	0	1	1
5	Time value "xx--" with time base "H:M"	23 _{bcd}	0	0	1	0	0	0	1	1
6	Hour value in days	03 _{bcd}	0	0	0	0	0	0	1	1

The slave responds with the following values:

Byte	Meaning	Value	Bit							
			7	6	5	4	3	2	1	0
0	Response: Write successful	C1 _{hex}	1	1	0	0	0	0	0	1
1	Invalid	→	x	x	x	x	x	x	x	x
2 – 6	Same content as with master									

Read time switch (channel, ON time, OFF time)

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	@1 Channel A	2B	—	1	0	1	0	1	0	1	1
	@1 Channel B	2C	—	1	0	1	0	1	1	0	0
	@1 Channel C	2D	—	1	0	1	0	1	1	0	1
	@1 Channel D	2E	—	1	0	1	0	1	1	1	0
	@2 Channel A	2F	—	1	0	1	0	1	1	1	1
	@2 Channel B	30	—	1	0	1	1	0	0	0	0
	@2 Channel C	31	—	1	0	1	1	0	0	0	1
	@2 Channel D	32	—	1	0	1	1	0	0	1	0
	@3 Channel A	33	—	1	0	1	1	0	0	1	1
	@3 Channel B	34	—	1	0	1	1	0	1	0	0
	@3 Channel C	35	—	1	0	1	1	0	1	0	1
	@3 Channel D	36	—	1	0	1	1	0	1	1	0
	@4 Channel A	37	—	1	0	1	1	0	1	1	1
	@4 Channel B	38	—	1	0	1	1	1	0	0	0
	@4 Channel C	39	—	1	0	1	1	1	0	0	1
	@4 Channel D	3A	—	1	0	1	1	1	0	1	0
	Response ¹										
	Read successful	—	C2/42	1	0	0	0	0	0	1	0
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Invalid	00	→	x	x	x	x	x	x	x	x
2	Weekday, Parameters menu display	00	→ table 44								
3	Minute (switch point ON)	00	→ table 45								
4	Hour (switch point ON)	00	→ table 46								
5	Minute (switch point OFF)	00	→ table 47								
6	Hour (switch point OFF)	00	→ table 48								

1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

Table 44: Byte 2: Weekday, starting, ending, Parameters menu

	Bit							
	7	6	5	4	3	2	1	0
ON day								
None set							0	0
Monday							0	0
Tuesday							0	1
Wednesday							0	1
Thursday							1	0
Friday							1	0
Saturday							1	1
Sunday							1	1
OFF day								
None set			0	0	0			
Monday			0	0	1			
Tuesday			0	1	0			
Wednesday			0	1	1			
Thursday			1	0	0			
Friday			1	0	1			
Saturday			1	1	0			
Sunday			1	1	1			
Switch time								
ON > OFF				1				
ON < OFF				0				
Appears in the Parameters menu								
No		1						
Yes		0						

Table 45: Byte 3: Minute (ON time)

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

Table 46: Byte 4: Hour (ON time)

Value (bcd)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	0
....
23	0	0	1	0	0	0	1	1	

Table 47: Byte 5: Minute (OFF time)

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

Table 48: Byte 6: Hour (OFF time)

Value (bcd)	Bit								
	7	6	5	4	3	2	1	0	
00	0	0	0	0	0	0	0	0	0
....
23	0	0	1	0	0	0	1	1	

Example

The master initiates the command to read the values of channel "A" of Obj4:

Byte	Meaning	Value (hex)	Bit							
			7	6	5	4	3	2	1	0
0	Command: Obj4 Channel A	37	0	0	1	1	0	1	1	1
1 – 6		00								

The slave responds with the following values:

- Day: Monday (001) to Friday (101)
- ON: 19:00
- OFF: 06:30
- Switch time ON > OFF (1)
- Channel appears in the Parameters menu

Byte	Meaning	Value	Bit							
			7	6	5	4	3	2	1	0
0	Response: : Read successful	42	0	1	0	0	0	0	1	0
1	Invalid		x	x	x	x	x	x	x	x
2	Weekday, Parameters menu display	69 _{bcd}	0	1	1	0	1	0	0	1
3	Minute (switch point ON)	00 _{bcd}	0	0	0	0	0	0	0	0
4	Hour (switch point ON)	19 _{bcd}	0	0	0	1	1	0	0	1
5	Minute (switch point OFF)	30 _{bcd}	0	0	1	1	0	0	0	0
6	Hour (switch point OFF)	06 _{bcd}	0	0	0	0	0	1	1	0

Write time switch (channel, ON time, OFF time)

Telegram structure

Byte	Meaning	Value (hex), sent by		Bit							
		Master	Slave	7	6	5	4	3	2	1	0
0	Command ¹										
	⊕1 Channel A	12	—	1	0	0	1	0	0	1	0
	⊕1 Channel B	13	—	1	0	0	1	0	0	1	1
	⊕1 Channel C	14	—	1	0	0	1	0	1	0	0
	⊕1 Channel D	15	—	1	0	0	1	0	1	0	1
	⊕2 Channel A	16	—	1	0	0	1	0	1	1	0
	⊕2 Channel B	17	—	1	0	0	1	0	1	1	1
	⊕2 Channel C	18	—	1	0	0	1	1	0	0	0
	⊕2 Channel D	19	—	1	0	0	1	1	0	0	1
	⊕3 Channel A	1A	—	1	0	0	1	1	0	1	0
	⊕3 Channel B	1B	—	1	0	0	1	1	0	1	1
	⊕3 Channel C	1C	—	1	0	0	1	1	1	0	0
	⊕3 Channel D	1D	—	1	0	0	1	1	1	0	1
	⊕4 Channel A	1E	—	1	0	0	1	1	1	1	0
	⊕4 Channel B	1F	—	1	0	0	1	1	1	1	1
	⊕4 Channel C	20	—	1	0	1	0	0	0	0	0
	⊕4 Channel D	21	—	1	0	1	0	0	0	0	1
	Response ¹										
	Write successful	—	C1/41	1	0	0	0	0	0	0	1
	Command rejected	—	C0/40	1	0	0	0	0	0	0	0
1	Weekday, Parameters menu display	00	→ table 44								
2	Minute (switch point ON)	00	→ table 45								
3	Hour (switch point ON)	00	→ table 46								
4	Minute (switch point OFF)	00	→ table 47								
5	Hour (switch point OFF)	00	→ table 48								
6	Not used										

- 1) Observe the Byte 0 Bit 7 data exchange procedure, → page 59.

The values for minute and hour of the switch points are part of an “*.eas file”. If these values are changed, the original “*.eas file” will no longer match the file in EASY6.... .

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

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

The comparison shows that the circuit diagrams are not identical.

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	Command: @2 Channel C	18 _{hex}	0	0	0	1	1	0	0	0
1	Weekday, Parameters 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								

8 Control Commands for easy700 (DPV0)

Data exchange procedure

The “Control commands 9 bytes” module allows extended data exchange of the easy700 on the PROFIBUS-DP communication bus. This allows you to transfer services from the following areas:

- “Read/write date and time” (page 104)
- “Read/write image data” (page 108) and
- “Read/write function block data” (page 131).

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



Attention!

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



Caution!

Only those values specified for the command code should be used.

Check the values that you write in order to avoid malfunctions.

Requirement:

The “Control commands 9 byte” module must have been selected.

The master initiates the data exchange of the control commands and the addressed slave responds.

During communication 9 data bytes (byte 0 = toggle byte, bytes 1 to 8 information bytes) are sent via PROFIBUS.

The basic telegram structure is shown in the following diagram.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
--------	--------	--------	--------	--------	--------	--------	--------	--------

Byte 0 – Toggle byte

Byte 0 is used to activate the sending of a control command with the toggle function.

Bit	7	6	5	4	3	2	1	0
01hex/ 861hex	0/1	0	0	0	0	0	0	1
Toggle bit	fixed							

Procedure

- ▶ To send a command, bit 7 must be toggled, i.e. set either from 1 to 0 or from 0 to 1.
- ▶ Then poll the toggle bit for the coupling modules response until it has the same status as the toggle bit sent. This status indicates to the master that the response to the sent command is valid.
- ▶ Do not send a new command until you have received a response (changing of the toggle bit), otherwise the response of the previous command will be overwritten before it can be read.



In order to use input/output data and control commands simultaneously:

Only after the “Control commands” data exchange has been completed will the I/O data be refreshed.

All specified commands and parameters must be transferred in hexadecimal format.

The following tables show the different control commands possible. These essential control commands fall into three essential categories – real-time clock, image and function blocks.

Read/write date and time

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Len	05	05
3	Index	0 – 2 ¹	0 – 2 ¹
4 – 8	Data 1 – 4	depending on index, → table 49	

- 1) 0 = Time/date, → table 49
- 1 = Summer time, → table 50
- 2 = Winter time, → table 51

Table 49: Index 0 – date and time of real-time clock

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

Table 50: Index 1 – Summer time

Byte	Contents		Value (hex)
4	Data 1	Area	
		None	00
		Rule	01
		Automatic EU	02
		Automatic GB	03
		Automatic US	04
for "Area" = "Rule":			
5	Data 2	Summer time switching rule	→ table 52
6	Data 3		
7	Data 4		
8	Data 5		

Table 51: Index 2 – Winter time
(only valid if Area = "Rule" selected)

Byte	Contents		Value (hex)
4	Data 1	Area = Rule	01
5 – 8	Data 2 – 5	Winter time switching rule	→ table 52

Switching rule bit array



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

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

Table 52: Switching rule bit array

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

Read/write image data



Please also observe the relevant description of possible image data provided in the easy700 manual (AWB2528-1508GB) or in the EASY-SOFT Help.

The latest edition of the manual is available as a PDF file from the Internet at: <http://www.moeller.net/support>:
Search Term: AWB2528-1508GB

The information provided in section "General notes on working with image data" on page 64 also applies to easy700.

Overview

Operands	Meaning	Read/ write	Type	Page
A1 – A16	"Analog value comparators/threshold comparators: A1 – A16"	Read	8B	110
C1 – C16	"Counters: C1 – C16"	Read	EE	111
D1 – D16	"Text function blocks: D1 – D16"	Read	94	112
I1 – I16	"Local inputs: I1 – I16"	Read	84	113
IA1 – IA4	"Local analog inputs: IA1 – IA4"	Read	8C	115
M1 – M16, N1 – N16	"Markers: M1 – M16/N1 – N16"	Write	86/87	117
M1 – M16, N1 – N16	"Markers: M1 – M16/N1 – N16"	Read	86/87	119
O1 – O4	"Operating hours counters: O1 – O4"	Read	EF	121
P1 – P4	"Local P buttons: P1 – P4"	Read	8A	122
Q1 – Q8	"Local outputs: Q1 – Q8"	Read	85	124
R1 – R16/ S1 – S8	"Inputs/outputs of EASY-LINK: R1 – R16/ S1 – S8"	Read	88/89	125
T1 – T16	"Timing relays: T1 – T16"	Read	ED	127
Y1 – Y4	"Year time switch: Y1 – Y8"	Read	91	128
Z1 – Z3	"Master reset: Z1 – Z3"	Read	93	129
H1 – H4	"7-day time switch: H1 – H8"	Read	90	130

**Analog value comparators/threshold comparators:
A1 – A16**

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	8B	8B
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 53
6	Data 2 (Low Byte)	00	→ table 53
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

Table 53: Byte 5 to 6: Data 1 to 2

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

Counters: C1 – C16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C01
2	Len	01	01
3	Type	EE	EE
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 63
6	Data 2 (Low Byte)	00	→ table 63
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

Table 54: Byte 5 to 6: Data 1 to 2

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

Text function blocks: D1 – D16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	94	94
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 55
6	Data 2 (High Byte)	00	→ table 55
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

Table 55: Byte 5 to 6: Data 1 to 2

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

Local inputs: I1 – I16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C01)
2	Len	02	02
3	Type	84	84
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 56
6	Data 2 (High Byte)	00	→ table 56
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

Table 56: Byte 5 to 6: 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 PROFIBUS-DP. The 16-bit value is transferred in Intel format (Low Byte first).

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	02	02
3	Type	8C	8C
4	Index	00 – 03 ²	00 – 03 ²
5	Data 1 (Low Byte)	00	→ table 57
6	Data 2 (High Byte)	00	→ table 57
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response: Read successful	–	C2
2	Len	02	02
3	Type	8C	8C
4	Index	02 ¹	02 ¹
5	Data 1	00	4B
6	Data 2	00	03
7	Data 3	00	00
8	Data 4	00	00

1) 02 = Analog input I11

Byte 5 – Data 1 (Low Byte): 4B_{hex}

Byte 6 – Data 2 (High Byte): 03_{hex}

→ corresponding 16-bit value: 034B_{hex} = 843

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

$$\frac{10 \text{ V}}{1023} \times \text{IO bit value} \Rightarrow \frac{10 \text{ V}}{1023} \times 843 = 8.24 \text{ V}$$

Markers: M1 – M16/N1 – N16**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Write	8C	–
	Response:		
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type ²		
	With M marker	86	86
	With N marker	87	87
4	Index ²	00 – 0F	00 – 0F
5	Data 1 (Low Byte) ³	00/01	00/01
6 – 8	Data 2 – 4	00	00

- 1) Possible causes → page 150
- 2) There are 16 M markers and 16 N markers. The markers are addressed by Type and Index:
Use Type to select the M or N marker.
Use Index to select the marker number.
- 3) The marker is set if a value is written to the data byte that does not equal zero. The marker is reset accordingly if the value 0 is written to data byte Data 1.

Example:
Marker M13 is set.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Write	8C	–
	Response:		
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type		
	M marker	86	86
4	Index	0C	0C
5	Data 1	01	00
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Markers: M1 – M16/N1 – N16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type		
	M marker	86	86
	N marker	87	87
4	Index ²	00	00
5	Data 1 (Low Byte)	00	→ table 58
6	Data 2 (Low Byte)	00	→ table 58
7 – 8	Data 3 – 4	00	00

- 1) Possible causes → page 150
- 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 58: Byte 5 to 6: Data 1 to 2

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

Example: The N markers are read:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	-
	Response:		
	Read successful	-	C2
	Command rejected	-	C0 ¹
2	Len	01	01
3	Type		
	N marker	87	87
4	Index	00	00
5	Data 1 (Low Byte)	00	04
6	Data 2 (Low Byte)	00	84
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

The markers N3, N11 and N16 are set.

Operating hours counters: 01 – 04

The following commands are used to read the logic state of the operating hours counters 01 – 04.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	EF	EF
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 59
6 – 8	Data 2 – 4	00	00

- 1) Possible causes → page 150

Table 59: Byte 5: Data 1

Data 1	Bit 7	6	5	4	3	2	1	0
01								0/1
02								0/1
03								0/1
04								0/1
...	

Local P buttons: P1 – P4

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



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

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	8A	8A
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 60
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Table 60: Byte 5: 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 PROFIBUS-DP fieldbus.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹⁾
2	Len	01	01
3	Type	85	85
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 61
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Table 61: Byte 5: Data 1

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

Example:

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

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

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C01)
2	Len	01	01
3	Type		
	for R data	88	88
	for S data	89	89
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 62
6	Data 2 (Low Byte)	00	→ table 62
7 – 8	Data 3 – 4	00	00

- 1) Possible causes → page 150

Table 62: Byte 5 to 6: Data 1 to 2

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

Timing relays: T1 – T16

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C01
2	Len	01	01
3	Type	ED	ED
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 63
6	Data 2 (Low Byte)	00	→ table 63
7 – 8	Data 3 – 4	00	00

1) Possible causes → page 150

Table 63: Byte 5 to 6: Data 1 to 2

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

Year time switch: Y1 – Y8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	91	91
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 64
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Table 64: Byte 5: Data 1

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

Example:

Data 1 = 1_{hex} → HY2 is active

Master reset: Z1 – Z3**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0 ¹
2	Len	01	01
3	Type	93	93
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 65
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Table 65: Byte 5: 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	0		

7-day time switch: ⊗1 – ⊗8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command: Read	88	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C01
2	Len	01	01
3	Type	90	90
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 66
6 – 8	Data 2 – 4	00	00

1) Possible causes → page 150

Table 66: Byte 5: 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 (AWB2528-1508GB) or in the EASY-SOFT Help.

General notes

Always note the following when working with function blocks:

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

Overview

Operands	Meaning	Read/ write	Type	Page
A1 – A16	"Analog value comparator/threshold comparator: A1 – A16"	Read/write	8D	132
C1 – C16	"Counter relays: C1 – C16"	Read/write	8F	135
O1 – O4	"Operating hours counters: O1 – O4"	Read/write	92	138
T1 – T16	"Timing relays: T1 – T16"	Read/write	8E	140
Y1 – Y8	"Year time switch: Y1 – Y8"	Read/write	A2	144
⊕1 – ⊕8	"7-day time switch: ⊕1 – ⊕8"	Read/write	A1	147

**Analog value comparator/threshold comparator:
A1 – A16****Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C01
2	Type	8D	8D
3	Instance ²	00 – 0F	00 – 0F
4	Index	→ table 67	
5 – 8	Data 1 – 4	depending on index, → table 68	

- 1) Possible causes → page 150
- 2) easy provides 16 analog comparators A1 to A16 for use as required. These can be addressed using the instance (0 – F).

Table 67: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 68		×	
01	Control byte → table 69		×	
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. It should be remembered that the low byte is in Data 1 (Byte 5) and the high byte 2 (byte 8) in Data 2.
Example: 5327_{dec} = 14CF_{hex} → Data 1 = 0xCF, Data 2 = 0x14

Table 68: Index 00 – Parameters

Meaning	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Appears in the Parameters menu																	
Yes/no																	0/1
Compare																	
FB not used														0	0	0	
EQ (=)														0	0	1	
GE (\geq)														0	1	0	
LE (\leq)														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 5) = 0xA3, Data 2 (Byte 6) = 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 Parameters menu;

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

Table 69: Index 01 – Control byte

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

1) Status 1 if comparison condition is fulfilled.

Counter relays: C1 – C16

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Type	8F	8F
3	Instance ²	00 – 0F	00 – 0F
4	Index	→ table 70	
5 – 8	Data 1 – 4	depending on index, → table 71	

1) Possible causes → page 150

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

Table 70: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 71		×	
01	Control byte → table 72		×	
02	Actual value	S1 ²	×	c ¹
03	Counter setpoint 2	S2 ²	×	c ¹

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

Table 71: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the Parameters 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 5) = 0x07

Meaning:

The values appear in the Parameters menu. The counter is used in the mode of the frequency counter. The counter setpoint S1 is not assigned to a constant and cannot therefore be written to.

Table 72: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	C4	RE ³	D ²	Q1 ¹

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

Example:

the actual value of C3 is to be read:

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

Explanation:

Data 1 = 12

Data 2 = 03

→ resulting 16-bit value = 0312_{hex} = 786_{dec}

Counter status = 786

Operating hours counters: O1 – O4

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C01
2	Type	92	92
3	Instance ²	00 – 03	00 – 03
4	Index	→ table 73	
5 – 8	Data 1 – 4	depending on index, → table 74	

- 1) Possible causes → page 150
- 2) easy provides 4 operating hours counters O1 to O4. These can be addressed using the instance (0 – 3).

Table 73: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 74		×	
01	Control byte → table 75		×	
02	Actual value	S1 ²	×	c1
03	Counter setpoint 2	S2 ²	×	c1

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

Table 74: Index 00 – Parameters

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

Example:

Data 1 (Byte 5) = 0x01

Meaning:

The values appear in the Parameters menu.

Table 75: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		-	-	-	-	-	RE ³	EN ²	Q11

1) Contact type

2) Enable, the timing relay is started (trigger coil)

3) Reset, the timing relay is reset (reset coil)

Example:

Index 02/03

Transferred values: Data 1 0x21

Data 2 0x23

Data 3 0x40

Data 4 0x00

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

Timing relays: T1 – T16**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Type	8E	8E
3	Instance ²	00 – 0F	00 – 0F
4	Index	→ table 76	
5 – 8	Data 1 – 4	depending on index, → table 77	

- 1) Possible causes → page 150
- 2) easy provides 16 timing relays T1 to T16 for use as required. These can be addressed using the instance (0 – F).

Table 76: Operand overview

Index (hex)	Operand		Read	Write
00	Parameters → table 77		×	
01	Control byte → table 78		×	
02	Actual value 1	T	×	c ¹
03	Time setpoint value 1	S1 ²	×	c ¹
04	Time setpoint 2	S2 ²	×	c ¹

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

Table 77: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the Parameters 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 generator					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 5) = 0xAC

Meaning:

The values appear in the Parameters menu. The timer is used in the pulse generator 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 78: Index 01 – Control byte

	Bit	7	6	5	4	3	2	1	0
FB input/output Data 3		–	–	–	–	ST ⁴	RE ³	EN ²	Q11

- 1) Contact type
- 2) Enable, the timing relay is started (trigger coil)
- 3) Reset, the timing relay is reset (reset coil)
- 4) Stop, the timing relay is stopped (Stop coil)

Example:

The time setpoint 1 is to be read:

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

Explanation:

Data 1 = 4C

Data 2 = 06

→ resulting 16-bit value = 064C_{hex} = 1612_{dec}

Meaning depending on set time base:

Millisecond	S	16120 ms	16.120 s
Seconds	M:S	1620 s	26:52 Minutes
Minute	H:M	1612 min	67:04 Hours

Year time switch: Y1 – Y8**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Type	A2	A2
3	Instance ²	00 – 07	00 – 07
4	Index	→ table 79	
5 – 8	Data 1 – 4	depending on index, → table 80	

- 1) Possible causes → page 150
- 2) easy provides 8 year time switches Y1 to Y8 for use as required. These can be addressed using the instance (0 – 7).

Table 79: Operand overview

Index (hex)	Operand	Read	Write
00	Parameters → table 80	×	
01	Control byte → table 81	×	
	Channel A	×	c ¹
11	ON time	×	c ¹
12	Time point OFF	×	c ¹
	Channel B	×	c ¹
21	ON time	×	c ¹
22	OFF time	×	c ¹
	Channel C	×	c ¹
31	ON time	×	c ¹
32	OFF time	×	c ¹
	Channel D	×	c ¹
41	ON time	×	c ¹
42	OFF time	×	c ¹

- 1) The value can only be written if it is assigned to a constant in the program.
- 2) The switch points are transferred in data bytes Data 1 – Data 3.

Table 80: Index 00 – Parameters

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

Example:

Data 1 (Byte 5) = 0x03 → The values of the year time switch of channel A and B appear in the Parameters menu.

Table 81: Index 01 – Control byte

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

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

Channel A, Index 11/12

Index 0x11 channel A ON time

Index 0x12 channel A OFF time

 Data 1 (Byte 5) – Day

 Data 2 (Byte 6) – Month

 Data 3 (Byte 7) – Year

Example:

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

Index = 0x11

 Data 1 = 0x15

 Data 2 = 0x04

 Data 3 = 0x04

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

Index = 0x22

 Data 1 = 0x05

 Data 2 = 0x0B

 Data 3 = 0x0C

7-day time switch: $\Theta 1 - \Theta 8$ **Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 102	
1	Command:		
	Read	89	–
	Write	8D	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0 ¹
2	Type	A1	A1
3	Instance ²	00 – 07	00 – 07
4	Index	→ table 82	
5 – 8	Data 1 – 4	depending on index, → table 83	

1) Possible causes → page 150

2) easy provides 8 seven-day time switches $\Theta 1$ to $\Theta 8$ use as required.
These can be addressed using the instance (0 – 7).

Table 82: Operand overview

Index (hex)	Operand	Read	Write
00	Parameters → table 83	×	
01	Control byte → table 84	×	
11	Channel A Day on/off	×	c1
12	On time	×	c1
13	Off time	×	c1
21	Channel B Day on/off	×	c1
22	On time	×	c1
23	Off time	×	c1
31	Channel C Day on/off	×	c1
32	On time	×	c1
33	Off time	×	c1
41	Channel D Day on/off	×	c1
42	On time	×	c1
43	Off time	×	c1

- 1) The value can only be written if it is assigned to a constant in the program.
- 2) A 16-bit value is transferred in data bytes Data 1 – Data 4. It should be remembered that Data 1 is the low byte and Data 2 the high byte.

Table 83: Index 00 – Parameters

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

Example:

Data 1 (Byte 5) = 0x03

Meaning:

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

Table 84: Index 01 – Control byte

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

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

Channel A, Index 11/12/13

Index 0x11 channel A Weekday on/off

Data 1 (Byte 5) – Weekday on

Data 2 (Byte 6) – Weekday off

0x01 = Sunday ... 0x07 = Saturday

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

Index 0x12 – On time (2 Byte)

Index 0x13 – Off time (2 Byte)

Data 1 (Byte 5) – Hour

Data 2 (Byte 6) – Minute

Example: On time at 13:43 p.m.

Data 1 = 0x0D

Data 2 = 0x2B

**Analysis – error codes via
EASY-LINK**

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

Telegram structure

Byte	Meaning	Slave transmits (value hex)
0	Toggle byte	→ page 102
1	Response	
	Command rejected	C0
2	Type	00
3	Instance	00
4	Index	00
5	Error code	→ table 85

Table 85: Error codes

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

9 Control Commands for easy800/ MFD (DPV0)

Data exchange procedure

The Control commands 9 bytes module allows extended data exchange of the easy800 and the MFD-Titan on the PROFIBUS-DP communication bus. This allows you to transfer services from the following areas:

- “Read/write date and time” (page 156)
- “Read/write image data” (page 160) and
- “Read/write function block data” (page 181).

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



Attention!

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



Caution!

Only those values specified for the command code should be used.

Check the values that you write in order to avoid malfunctions.

Requirement:

The “Control commands 9 byte” module must have been selected.

The master initiates the data exchange of the control commands and the addressed slave responds.

During communication 9 data bytes (byte 0 = toggle byte, bytes 1 to 8 information bytes) are sent via PROFIBUS.

The basic telegram structure is shown in the following diagram.

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
--------	--------	--------	--------	--------	--------	--------	--------	--------

Byte 0 – Toggle byte

Byte 0 is used to activate the sending of a control command with the toggle function.

Bit	7	6	5	4	3	2	1	0
01hex/ 861hex	0/1	0	0	0	0	0	0	1
Toggle bit	fixed							

Procedure

- ▶ To send a command, bit 7 must be toggled, i.e. set either from 1 to 0 or from 0 to 1.
- ▶ Then poll the toggle bit for the coupling modules response until it has the same status as the toggle bit sent. This status indicates to the master that the response to the sent command is valid.
- ▶ Do not send a new command until you have received a response (changing of the toggle bit), otherwise the response of the previous command will be overwritten before it can be read.



In order to use input/output data and control commands simultaneously:

Only after the “Control commands” data exchange has been completed will the I/O data be refreshed.

All specified commands and parameters must be transferred in hexadecimal format.

The following tables show the different control commands possible. These essential control commands fall into three essential categories – real-time clock, image and function blocks.

Version history

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

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

Read/write date and time

Please also note the relevant description of the real-time clock provided in the easy800 manual.

The latest edition of the manual is available as a PDF file from the Internet at: <http://www.moeller.net/support>:
Search Term: AWB2528-1423GB

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Len	05	05
3	Index	00	00
4 – 8	Data 1 – 5		
	Read operation	00	→ table 86
	Write operation	→ table 86	00

Table 86: Byte 4 – 8: Data 1 – 5

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

Winter/summer time, DST**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command		
	Read	93	–
	Write	B3	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Len	05	05
3	Index	01	01
4 – 8	Data 1 – 5		
	Read operation	00	→ table 87
	Write operation	→ table 87	00

Table 87: Byte 4 – 8: Data 1 – 5

Byte	Contents	Value (hex)
4	Data 1 Area	
		None 00
		Manual 01
		Automatic EU 02
		Automatic GB 03
		Automatic US 04
5	Data 2 ¹ Set summer time day (1 to 28, 29, 30, 31 depending on month and year)	00 – 3B
6	Data 3 ¹ Set Summer time month (1 to 12)	01 – 1F
7	Data 4 ¹ Set winter time day (1 to 28, 29, 30, 31 depending on month and year)	01 – 0C
8	Data 5 ¹ Set winter time month (1 to 12)	00 – 63

- 1) The additional parameters Data 2 to Data 5 for automatic DST change are only relevant if you have set the "Manual" parameter for Data 1.

Example

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Write	B3	-
	Response: Write successful	-	C1
2	Len	05	05
3	Index	00	00
4	Data 1	0E	00
5	Data 2	24	00
6	Data 3	17	00
7	Data 4	05	00
8	Data 5	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 or in the EASY-SOFT Help.

The latest edition of the manual is available as a PDF file from the Internet at: <http://www.moeller.net/support>:
Search Term: AWB2528-1423GB

The information provided in section "General notes on working with image data" on page 64 also applies to easy800.

Overview

Operands	Meaning	Read/ write	Comm and	Page
IW0	"Read local inputs IW0"	Read	01	161
IW1 – IW8	"Read inputs of the stations IW1 to IW8"	Read	01	163
IA1 – IA4	"Read local analog inputs IA1 to IA4"	Read	02	164
ID1 – ID16	"Read local diagnostics ID1 to ID16"	Read	03	166
QW0, QW1 – QW8	"Read and write local QW0 outputs/outputs of the stations QW1 to QW8"	Read/write	04	168
QA1	"Read and write local analog output QA1"	Read/write	05	170
P1 – P4	"Read local P buttons"	Read	06	171
R1 – R16 S1 – S8	"Read RW.. inputs/SW.. outputs from EasyLink"	Read	07/09	173
RN1 – RN32 SN1 – SN32	"Read receive data network RN1 .. RN32/send data network SN1 .. SN32"	Read	08/0A	175
M...	"Read and write markers"	Read/write	0B – 0E	177

Read local inputs IWO

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	01	01
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 88
6	Data 2 (High Byte)	00	→ table 88
7 – 8	Data 3 – 4	00	00

Table 88: Byte 5 to 6: Data 1 to 2

Data 1	Bit	7	6	5	4	3	2	1	0
I1									0/1
I2									0/1
..									..
I8									0/1

Data 2	Bit	7	6	5	4	3	2	1	0
I9									0/1
I10									0/1
..									..
I16									0/1

Example: Read local inputs IW0

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Read	91	-
	Response: Read successful	-	C2
2	Len	02	02
3	Type	01	01
4	Index	00	00
5	Data 1	00	C4
6	Data 2	00	02
7	Data 3	00	00
8	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.

Read inputs of the stations IW1 to IW8

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	01	01
4	Index	01 – 08 ¹	01 – 08 ¹
5	Data 1 (Low Byte)	00	→ table 88 on page 161.
6	Data 2 (High Byte)	00	
7 – 8	Data 3 – 4	00	00

1) Corresponds to address of network station

Read local analog inputs IA1 to IA4

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

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	02	02
4	Index	01 – 04 ¹	01 – 04 ¹
5	Data 1 (Low Byte)	00	See example
6	Data 2 (High Byte)	00	See example
7 – 8	Data 3 – 4	00	00

1) 01 = Analog input I7

02 = Analog input I8

03 = Analog input I11

04 = Analog input I12

Example

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Read	91	-
	Response: Read successful	-	C2
2	Len	02	02
3	Type	02	02
4	Index	01 ¹	01 ¹
5	Data 1	00	D9
6	Data 2	00	02
7	Data 3	00	00
8	Data 4	00	00

1) 01 = Analog input 1

Byte 5 – Data 1 (Low Byte): D9_{hex}

Byte 6 – Data 2 (High Byte): 02_{hex}

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

Read local diagnostics ID1 to 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
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	03	03
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 89
6	Data 2 (High Byte)	00	→ table 89
7 – 8	Data 3 – 4	00	00

Table 89: Byte 5 to 6: 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 and write local QW0 outputs/outputs of the stations QW1 to QW8

You can read and write the local outputs directly via PROFIBUS-DP. 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 160.

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command		
	Read	91	–
	Write	B1	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Len	02	02
3	Type	04	04
4	Index ¹	00/01 – 08	00/01 – 08
5	Data 1		
	Read operation	00	→ table 89
	For write operation	→ table 90	00
6 – 8	Data 2 – 4	00	00

1) 00 = Local output

01 – 08 = Outputs of network stations 1 – 8

Table 90: Byte5: 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							

Read and write 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, however, the value will only be output externally if the device concerned is in Run mode and the image concerned has not been overwritten by the actual program → section “Read/write image data” on page 160.

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command		
	Read	91	—
	Write	B1	—
	Response		
	Read successful	—	C2
	Write successful	—	C1
	Command rejected	—	C0
2	Len	02	02
3	Type	05	05
4	Index	00	00
5 – 6	Data 1 – 2		
	Read operation	00	See example
	For write operation	See example	00
7 – 8	Data 3 – 4	00	00

Example

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

500 = 01F4_{hex} Byte 5 – Data 1 (LowByte) : F4_{hex}
 Byte 6 – Data 2 (HighByte): 01_{hex}

Read local P buttons

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



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

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	06	06
4	Index	00	00
5	Data 1 (Low Byte)	00	→ table 91
6 – 8	Data 2 – 4	00	00

Table 91: Byte 5: 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

Read RW.. inputs/SW.. outputs from EasyLink

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
0	Toggle byte	→ page 154	
1	Command: Read	91	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Len	02	02
3	Type	For RW: 07 For SW: 09	For RW: 07 For SW: 09
4	Index	00/01 – 08 ¹	00/01 – 08 ¹
5	Data 1 (Low Byte)	00	→ table 92
6	Data 2 (High Byte)	00	→ table 92
7 – 8	Data 3 – 4	00	00

1) 00 = Local input/output

01 – 08 = Address of network station (NET-ID 1 – 8)

Table 92: Byte 5 to 6: Data 1 to 2

Data 1		Bit	7	6	5	4	3	2	1	0
RW	SW									
R1	S1									0/1
R2	S2									0/1
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

Read 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 is 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	
0	Toggle byte	→ page 154		
1	Command: Read	91	–	
	Response:			
	Read successful	–	C2	
	Command rejected	–	C0	
2	Len	04	04	
3	Type	For RN1 – RN32: 08 For SN1 – SN32: 0A		
4	Index	01 – 08 ¹	01 – 08 ¹	
5 – 8	Data 1 – 4	00	→ table 93	

1) Corresponds to NET-ID

Table 93: Byte 5 to 8: 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 and write markers

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command		
	Read	91	–
	Write	B1	–
	Response		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Len	→ table 94	→ table 94
3	Type		
4	Index		
5 – 8	Data 1 – 4		
	Read operation	00	→ "Example 1" on page 179
	For write operation	→ "Example 2" on page 180	00

Table 94: Byte 2 – 4: Len, Type, Index

Operand		Len	Type	Index
Marker bit	M1 .. M96	01 _{hex}	0B _{hex}	01 to 60 _{hex}
Marker byte	MB1 .. MB96	01 _{hex}	0C _{hex}	01 to 60 _{hex}
Marker word	MW1 .. MW96	02 _{hex}	0D _{hex}	01 to 60 _{hex}
Marker double word	MD1 .. MD96	04 _{hex}	0E _{hex}	01 to 60 _{hex}

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

**Attention!**

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

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



The relevant marker values are transferred in Intel format. In other words, the first byte is the low byte (Byte 5) and the last byte the high byte.

Example 1

Read marker bit M62

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Read	91	-
	Response: Read successful	-	C2
2	Len	01	01
3	Type	0B	0B
4	Index	3E	3E
5	Data 1	00	01
6	Data 2	00	00
7	Data 3	00	00
8	Data 4	00	00

Result: Data 1 = 01_{hex} → M62 was set

Example 2

Write marker word MW32 with 823

 $823_{\text{dec}} = 337_{\text{hex}}$ → Data 1 = 37_{hex} , Data 2 = 03_{hex}

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Write	B1	–
	Response: Write successful	–	C1
2	Len	02	02
3	Type	0D	0D
4	Index	20	20
5	Data 1	37	00
6	Data 2	03	00
7	Data 3	00	00
8	Data 4	00	00

Read/write function block data



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

The latest edition of the manual is available as a PDF file from the Internet at: <http://www.moeller.net/support>:
Search Term: AWB2528-1423GB)

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.
- All 32-bit values are treated as signed values. When transferring 32-bit values, ensure that the appropriate value range is suitable for long integers, i.e. signed.
32-bit value: -2 147 483 648 .. 0 .. +2 147 483 647

Overview

Operands	Meaning	Read/write	Type	Page
A01 – A32	"Analog value comparators A01 .. A32"	Read/write	11	183
AR01 – AR32	"Arithmetic function blocks AR01 .. AR32"	Read/write	12	185
BC01 – BC32	"Block compare function blocks BC01 .. BC32"	Read/write	25	187
BT01 – BT32	"Block transfer function blocks BT01 .. BT32"	Read/write	26	189
BV01 – BV32	"Boolean operation function blocks BV01 .. BV32"	Read/write	13	191
C01 – C32	"Counters C01 .. C32"	Read/write	14	193
CF01 – CF04	"Frequency counters CF01 .. CF04"	Read/write	15	195
CH01 – CH04	"High-speed counters CH01 .. CH04"	Read/write	16	197
CI01 – CI02	"Incremental encoder counters CI01 .. CI02"	Read/write	17	199
CP01 – CP32	"Comparators CP01 .. CP32"	Read/write	18	201
D01 – D32	"Text output function blocks D01 .. D32"	Read/write	19	203
DB01 – DB32	"Data function blocks DB01 .. DB32"	Read/write	1A	206
DC01 – DC32	"PID controllers DC01 .. DC32"	Read/write	27	208
FT01 – FT32	"Signal smoothing filters FT01 .. FT32"	Read/write	28	211
GT01 – GT32	"Receive network data function blocks GT01 .. GT32"	Read	1B	213
HW01 – HW32	"7-day time switches HW01 .. HW32"	Read	1C	215
HY01 – HY32	"Year time switches HY01 .. HY32"	Read	1D	218
LS01 – LS32	"Value scaling function blocks LS01 .. LS32"	Read/write	29	221
MR01 – MR32	"Master reset function blocks MR01 .. MR32"	Read	0F	223
NC01 – NC32	"Numerical converters NC01 .. NC32"	Read/write	2A	225
OT01 – OT04	"Operating hours counters OT01 .. OT04"	Read/write	1E	227
PT01 – PT32	"Send network data function blocks PT01 .. PT32"	Read	1F	229
PW01 – PW02	"Pulse width modulation function blocks PW01 .. PW02"	Read/write	2B	231
SC01	"Synchronize clock function block SC01"	Read	20	233
ST01	"Set cycle time function block ST01"	Read/write	2C	234
T01 – T32	"Timing relays T01 .. T32"	Read/write	21	236
VC01 – VC32	"Value limitation function blocks VC01 .. VC32"	Read/write	2D	239

Analog value comparators A01 .. A32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	11	11
3	Instance	01 – 20	01 – 20
4	Index	→ table 95	→ table 95
5 – 8	Data 1 – 4	00	depending on index, → table 96, 97

Table 95: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 96	×	
01	Mode, → table 97	×	
02	Comparison value 1	I1	c1
03	Gain factor for I1 ($I1 = F1 \times \text{value}$)	F1	c1
04	Comparison value 2	I2	c1
05	Gain factor for I2 ($I2 = F2 \times \text{value}$)	F2	c1
06	Offset for the value I1	OS	c1
07	Switching hysteresis for value I2 (the value of HY is for both positive and negative hysteresis.)	HY	c1

- 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 96: Index 0: Bit IO

FB output Data 3	Bit	7	6	5	4	3	2	1	0
	–	–	–	–	–	–	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 97: Index 1 - Mode

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

Arithmetic function blocks AR01 .. AR32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	12	12
3	Instance	01 – 20	01 – 20
4	Index	→ table 98	→ table 98
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 99, 100
	Write operation	depending on index, → table 99, 100	00

Table 98: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 99		×
01	Mode, → table 100		×
02	First operand	I1	×
03	Second operand	I2	×
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 99: Index 0: Bit IO

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

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

Table 100: Index 1 - Mode

Data 1 (hex)		
00	ADD	Add ($I1 + I2 = QV$)
01	SUB	Subtract ($I1 - I2 = QV$)
02	MUL	Multiply ($I1 \times I2 = QV$)
03	DIV	Divide ($I1 : I2 = QV$)

Block compare function blocks BC01 .. BC32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	25	25
3	Instance	01 – 20	01 – 20
4	Index	→ table 101	→ table 101
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 102, 103
	Write operation	depending on index, → table 102, 103	00

Table 101: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 102	×	
01	Mode, → table 103	×	
02	Source range 1 I1	×	c ¹
03	Target range 2 I2	×	c ¹
04	Number of elements to compare: 8 (max. 192 bytes)	×	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 102: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
		–	–	–	–	–	–	–	EN ¹
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 103: Index 1 - Mode

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

Block transfer function blocks BT01 .. BT32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	26	26
3	Instance	01 – 20	01 – 20
4	Index	→ table 104	→ table 104
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 105, 106
	Write operation	depending on index, → table 105, 106	00

Table 104: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 105	×	
01	Mode, → table 106	×	
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 .. Data 2 - High Byte).

Table 105: Index 0 – Bit IO

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

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

Error outputs

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

Table 106: Index 1 - Mode

Data 1 (hex)	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 function blocks BV01 .. BV32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	13	13
3	Instance	01 – 20	01 – 20
4	Index	→ table 107	→ table 107
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 108, 109
	Write operation	depending on index, → table 108, 109	00

Table 107: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 108	×	
01	Mode, → table 109	×	
02	First operand I1	×	c ¹
03	Second operand I2	×	c ¹
04	Operation result QV	×	

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



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

Table 108: Index 0: Bit IO

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

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

Table 109: Index 1 - Mode

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

Counters C01 .. C32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	14	14
3	Instance	01 – 20	01 – 20
4	Index	→ table 110	→ table 110
5 – 8	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	Value	Read	Write
00	Bit IO	→ table 111	×	
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 111: Index 0: Bit IO

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

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

Frequency counters CF01 .. CF04**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	15	15
3	Instance	01 – 04	01 – 04
4	Index	→ table 112	→ table 112
5 – 8	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	Read	Write
00	Bit IO, → table 113	×	
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 113: Index 0: Bit IO

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

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

High-speed counters CH01 .. CH04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	16	16
3	Instance	01 – 04	01 – 04
4	Index	→ table 114	→ table 114
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table
	Write operation	depending on index, → table	00

Table 114: Operand overview

Index (hex)	Operand	Value	Read	Write
00	Bit IO	→ table 115	×	
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 115: Index 0: Bit IO

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

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

Incremental encoder counters CI01 .. CI02**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	17	17
3	Instance	01 – 02	01 – 02
4	Index	→ table 116	→ table 116
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 117
	Write operation	depending on index, → table 117	00

Table 116: Operand overview

Index (hex)	Operand	Value	Read	Write
00	Bit IO	→ table 117	×	
01	Mode/Parameter	–	–	–
02	Upper setpoint	SH		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 117: Index 0 – Bit IO

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

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

Comparators CP01 .. CP32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	18	18
3	Instance	01 – 20	01 – 20
4	Index	→ table 118	→ table 118
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 119
	Write operation	depending on index, → table 119	00

Table 118: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 119	×	
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 119: Index 0 – Bit IO

FB output Data 3	Bit	7	6	5	4	3	2	1
		-	-	-	-	GT ¹	EQ ²	LT ³

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

Text output function blocks D01 ..D32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	19	19
3	Instance	01 – 20	01 – 20
4	Index	→ table 120	→ table 120
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 121
	Write operation	depending on index, → table 121	00

Table 120: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 121	×	
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	Read	Write
27	Scaling maximum value 2	×	
28	Scaling maximum value 3	×	
29	Scaling maximum value 4	×	
30	Control information line 1	×	
31	Control information line 2	×	
32	Control information line 3	×	
33	Control information line 4	×	

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



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

Table 121: Index 0 – Bit IO

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

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

Data function blocks DB01 .. DB32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	1A	1A
3	Instance	01 – 20	01 – 20
4	Index	→ table 122	→ table 122
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 123
	Write operation	depending on index, → table 123	00

Table 122: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 123	×	
01	Mode/Parameter	–	–
02	Input value: value that I1 is transferred to the QV output when the FB is triggered.	×	c ¹
03	Output value QV	×	

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



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

Table 123: Index 0 – Bit IO

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

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

PID controllers DC01 .. DC32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	27	27
3	Instance	01 – 20	01 – 20
4	Index	→ table 124	→ table 124
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 125, 126
	Write operation	depending on index, → table 125, 126	

Table 124: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 125	×	
01	Mode, → table 126	×	
02	Setpoint: -32 768 to +32 767	I1	×
03	Actual value: -32 768 to +32 767	I2	×
04	Proportional gain [%], Value range: 0 to 65 535	KP	×
05	Reset time [0.1 s], Value range: 0 to 65 535	TN	×
06	Rate time [0.1 s], Value range: 0 to 65 535	TV	×
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	×
08	Manual manipulated variable, value range: -4 096 to +4 095	MV	×
09	Manipulated variable <ul style="list-style-type: none"> • Mode: UNI, value range: 0 to +4 095 (12 bit) • Mode: BIP, value range: -4 096 to +4 095 (13 bit) 	QV	×

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



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

Table 125: 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 manipulated variable was exceeded

Table 126: Index 1 - Mode

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

Signal smoothing filters FT01 .. FT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	28	28
3	Instance	01 – 20	01 – 20
4	Index	→ table 127	→ table 127
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 128
	Write operation	depending on index, → table 128	00

Table 127: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 128	×	
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 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 128: Index 0 – Bit IO

FB output Data 3	Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	EN ¹	

- 1) Activates the function block on status 1.

Receive network data function blocks GT01 .. GT32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	1B	1B
3	Instance	01 – 20	01 – 20
4	Index	→ table 129	
5 – 8	Data 1 – 4	00	depending on index, → table 130, 131

Table 129: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 130	×	–
01	Mode/Parameters, → table 131	×	–
02	Output value: actual QV value from the network	×	–



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

Table 130: 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 131: 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
Parameter	Data 3	Instance ²	
		0	PT01
	
		31	PT32

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

7-day time switches HW01 .. HW32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	1C	1C
3	Instance	01 – 20	01 – 20
4	Index	→ table 132	
5 – 8	Data 1 – 4	00	depending on index, → table 133

Table 132: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 133	×	
01	Mode/Parameter	–	–
02	Parameters → table 134 Channel A	×	
03	Channel B		
04	Channel C		
05	Channel D		

Table 133: 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 134: 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
	Weekday				Hour				Minute							

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

m5 to m0: Minute (0 to 59)

h4 to h0: Hour (0 to 23)

d5 to d0: Weekday (0 = Sunday to 6 = Saturday)

Example

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

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Read	92	-
	Response: Read successful	-	C2
2	Type	1C	1C
3	Instance	13	13
4	Index	02	02
5	Data 1	00	62
6	Data 2	00	0B
7	Data 3	00	7B
8	Data 4	00	25

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

Switch-on time:

Weekday = 01_{hex} .. Monday

Hour = 0D_{hex} .. 1300 hours

Minute = 22_{hex} .. 34 minutes

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

Switch-off time:

Weekday = 04_{hex} .. Thursday

Hour = 15_{hex} .. 2100 hours

Minute = 59_{hex} .. 34 minutes

Year time switches HY01 .. HY32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	1D	1D
3	Instance	01 – 20	01 – 20
4	Index	→ table 135	
5 – 8	Data 1 – 4	00	depending on index, → table 136

Table 135: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO → table 136	×	
01	Mode/Parameter	–	–
02	Parameters → table 137	×	
	Channel A		
03	Channel B		
04	Channel C		
05	Channel D		

Table 136: 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 137: 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							

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							

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							

Switch-on time:

Day = 14 = 0E_{hex} = 0000 1110b

Month = 6 (June) = 06_{hex} = 0000 0110b

Year = 2003 = 03_{hex} = 0000 0011b

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								
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				

Switch-off time:

Day = 3 = 03_{hex} = 0000 0011bMonth = 10 (October) = 0A_{hex} = 0000 1010bYear = 2012 = 0C_{hex} = 0000 1100b

Resulting telegram:

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	81	80
1	Command: Write	B2	–
	Response: Write successful	–	C1
2	Type	1D	1D
3	Instance	0E	0E
4	Index	02	02
5	Data 1	8E	00
6	Data 2	06	00
7	Data 3	43	00
8	Data 4	19	00



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

Value scaling function blocks LS01 .. LS32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	29	29
3	Instance	01 – 20	01 – 20
4	Index	→ table 138	→ table 138
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 139
	Write operation	depending on index, → table 139	

Table 138: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 139	×	
01	Mode/Parameter	—	—
02	Input value, I1 value range: 32 bit	×	c ¹
03	Interpolation point 1, X coordinate, value range: 32 bit	×	c ¹
04	Interpolation point 1, Y coordinate, value range: 32 bit	×	c ¹
05	Interpolation point 2, X coordinate, value range: 32 bit	×	c ¹
06	Interpolation point 2, Y coordinate, value range: 32 bit	×	c ¹
07	Output value: contains the scaled input value	×	

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

Table 139: Index 0 – Bit IO

FB output Data 3	Bit	7	6	5	4	3	2	1	0
	—	—	—	—	—	—	—	EN ¹	

- 1) Activates the function block on status 1.

Master reset function blocks MR01 .. MR32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	0F	0F
3	Instance	01 – 20	01 – 20
4	Index		
	Bit IO	00	00
	Mode	01	01
5 – 8	Data 1 – 4	00	depending on index, → table 140, 141

Table 140: 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 141: Index 1 - Mode

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

Numerical converters NC01 .. NC32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	2A	2A
3	Instance	01 – 20	01 – 20
4	Index	→ table 142	→ table 142
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 143, 144
	Write operation	depending on index, → table 143, 144	00

Table 142: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 143	×	
01	Mode, → table 144	×	
02	Input value: operand to be converted	×	c ¹
03	Output value: contains the conversion result	×	

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



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

Table 143: Index 0 – Bit IO

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

- 1) Activates the function block on status 1.

Table 144: Index 1 - Mode

Data 1 (hex)	
00	BCD
01	BIN

Converts a BCD coded decimal value to an integer value.

Converts an integer value to a BCD coded decimal value.

Operating hours counters OT01 .. OT04

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	1E	1E
3	Instance	01 – 04	01 – 04
4	Index	→ table 145	→ table 145
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 146
	Write operation	depending on index, → table 146	00

Table 145: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 146	×	
01	Mode/Parameter	–	–
02	Upper threshold value I1	×	c ¹
03	Actual value of operating QV hours counter	×	

- 1) 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		–	–	–	–	–	–	RE ¹	EN ²
FB output Data 3		–	–	–	–	–	–	–	Q1 ³

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



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

Send network data function blocks PT01 .. PT32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	1F	1F
3	Instance	01 – 20	01 – 20
4	Index	→ table 147	
5 – 8	Data 1 – 4	00	depending on index, → table 148

Table 147: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 148	×	
01	Mode/Parameter	–	–
02	Input value: Setpoint I1 that it transmitted to the NET network	×	



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

Table 148: Index 0 – Bit IO

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

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

Pulse width modulation function blocks
PW01 .. PW02

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	2B	2B
3	Instance	01 – 02	01 – 02
4	Index	→ table 149	→ table 149
5 – 8	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	Read	Write
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 to 65535	PD	c ¹
04	Minimum on duration [ms], Value range: 0 to 65535	ME	c ¹

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

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

Synchronize clock function block SC01

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	
1	Command: Read	92	–
	Response:		
	Read successful	–	C2
	Command rejected	–	C0
2	Type	20	20
3	Instance	01	01
4	Index	→ table 151	
5 – 8	Data 1 – 4	00	depending on index, → table 152

Table 151: Operand overview

Index (hex)	Operand	Read	Write
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	–	–	–	–	–	–	–	–	T ¹
FB output Data 3	–	–	–	–	–	–	–	–	Q12

- 1) Trigger coil. If the coil is triggered (rising edge), the current date, weekday and time of the sending station are automatically sent to the NET network.
- 2) Status 1 if the trigger coil SC01T_ is also 1.

Set cycle time function block ST01**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	2C	2C
3	Instance	01	01
4	Index	→ table 153	→ table 153
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 154
	Write operation	depending on index, → table 154	00

Table 153: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 154	×	
01	Mode/Parameter	–	–
02	Cycle time in ms, value range: 0 – 1000	I1	c ¹

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

Table 154: Index 0 – Bit IO

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

- 1) Activates the function block on status 1.

Timing relays T01 .. T32**Telegram structure**

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	21	21
3	Instance	01 – 20	01 – 20
4	Index	→ table 155	→ table 155
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 156, 157
	Write operation	depending on index, → table 156, 157	

Table 155: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 156	×	
01	Mode/Parameters, → table 157	×	
02	Setpoint 1: Time setpoint 1	×	c ¹
03	Setpoint 2: Time setpoint 2 (with timing relay with 2 setpoints)	×	c ¹
04	Actual value: Time elapsed in Run mode	×	

- 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 156: 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) Contact type

Table 157: Index 1 - Mode/Parameters

Mode	Data 1	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 generator
	7	Flashing relay (two time setpoints)
	8	Off-delayed, retriggerable (easy600 mode)
	9	Off-delayed with random setpoint, retriggerable (easy600 mode)
Parameter	Data 3	Mode
	0	S (milliseconds)
	1	M:S (seconds)
	2	H:M (minutes)

Value limitation function blocks VC01 .. VC32

Telegram structure

Byte	Meaning	Value (hex), sent by	
		Master	Slave
0	Toggle byte	→ page 154	→ page 154
1	Command:		
	Read	92	–
	Write	B2	–
	Response:		
	Read successful	–	C2
	Write successful	–	C1
	Command rejected	–	C0
2	Type	2D	2D
3	Instance	01 – 20	01 – 20
4	Index	→ table 158	→ table 158
5 – 8	Data 1 – 4		
	Read operation	00	depending on index, → table 159
	Write operation	depending on index, → table 159	

Table 158: Operand overview

Index (hex)	Operand	Read	Write
00	Bit IO, → table 159	×	
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 QV value present at input I1 within the set limits.	×	

- 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	–	–	–	–	–	–	–	–	EN ¹

- 1) Activates the function block on status 1.

10 Process Data for easy600 (DPV1)

The PROFIBUS-DPV1 functionality of the EASY204-DP (from device version 07) allows you to access the following data ranges of an easy600 via a class 1 or class 2 DPV1 master without any extensive programming required:

- Identification
- Mode
- Image
- Function blocks
- Date and time
- DST

The data from these ranges is located in Process Data Objects (data records) and can be read and/or written with the DPV1 “Read” and “Write” services.

Object overview

The following overview shows all the objects contained in the EASY204-DP that are available for a connected easy600.



The attributes API, Slot Number and Index form the address information for the DP master in order to access an object. Refer to the following section for further information about this.

Table 160: Process data objects in the EASY204-DP for easy600

Object name	API	Slot number	Index	Data length (octets)	Read (R) Write (W)	→ page
Easy operating mode	0	0	100	1	R/W	247
Easy identification	0	1	100	50	R	245
Image data						
EASY 600 inputs (I1 – I16)	0	0	197	4	R	249
EASY 600 outputs and markers (Q1 – Q8, M1 – M16, D1 – D8)	0	0	198	4	R	251
EASY 600 function relays (T1 – T8, C1 – C8, O1 – O4, A1 – A8)	0	0	199	4	R	254
Easy600 pushbuttons	0	0	200	1	R	257
Easy Link input data (R1 – R16)	0	0	98	2	R/W	259
Easy Link output data (S1 – S8)	0	0	99	1	R	261
Function blocks						
EASY 600 parameters Timing relays T1 to T8	0	0	211 – 218	6 Read 5 Write	R/W	263
EASY 600 parameters Counters C1 to C8	0	0	221 – 228	2 Read 3 Write	R/W	268
EASY 600 parameters Analog value comparators A1 – A8	0	0	231 – 238	3	R/W	271

Object name	API	Slot number	Index	Data length (octets)	Read (R) Write (W)	→ page
EASY 600 parameters 7-day time switch ④ 1 channel A	0	0	239	8	R/W	273
...			...			
EASY 600 parameters 7-day time switch ④1 channel D	0	0	242	8	R/W	273
EASY 600 parameters 7-day time switch ④2 channel A	0	0	243	8	R/W	273
...			...			
EASY 600 parameters 7-day time switch ④2 channel D	0	0	246	8	R/W	273
EASY 600 parameters 7-day time switch ④3 channel A	0	0	247	8	R/W	273
...			...			
EASY 600 parameters 7-day time switch ④3 channel D	0	0	250	8	R/W	273
EASY 600 parameters 7-day time switch ④4 channel A	0	0	251	8	R/W	273
...			...			
EASY 600 parameters 7-day time switch ④4 channel D	0	0	254	8	R/W	273
Easy clock	0	0	97	8	R/W	276
EASY 600 DST setting	0	0	210	1	R/W	279

Accessing objects

The Process Data Objects in the EASY204-DP are accessed via the DPV1 "Read" and "Write" services using the functions provided for this by the DP master system. Refer to the documentation of the manufacturer concerning this.

Function blocks are normally used to carry out the access. In IEC 61131-3 based systems, the function blocks RDREC (Read) and WRREC (Write) defined by the PROFIBUS Users' Organisation in regulation 2.182 are normally made available. These also enable optimum access to complex data structures.

The following is required to access the objects:

- The address of the local DPV1 master interface
- The station address of the EASY204-DP to be accessed
- The application identifier (API) in the EASY204-DP (only required for class 2 DPV1 masters)
- The module to be addressed in EASY204-DP (Slot Number)
- The address (Index) of the required Process Data Object in the addressed EASY204-DP module
- The data length of the required Process Data Object
- A defined variable (memory range) in the local application to which the read data is to be assigned to or which contains the data to be written.

Refer to the topology of your master system for the address of the local DPV1 master interface. Refer to the PROFIBUS topology for the station address of the EASY204-DP.

The API (only required for class 2 DPV1 masters) must be stated for all Process Data Objects as 0.

The slot number, the index, the data length and a sample variable definition (declaration) for an IEC 61131-3 system is provided in the following sections for each of the Process Data Objects provided in the EASY-204-DP for a connected easy600.

Read identification**"Easy Identification" object**

This object contains the device identification of the EASY204-DP and the basic unit connected (ASCII string).

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to obtain the identification of the device:

- Type of easy basic unit
- Firmware version of the easy basic unit
- Build number of the firmware of the easy basic unit
- Type of communication module
- Firmware version of the communication module.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 100.

Length of the object

The data to be read is 50 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variables (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_Ident	STRUCT
Type	ARRAY [1..14] OF CHAR or STRING [14]
SW_Version	ARRAY [1..10] OF CHAR or STRING [10]
SW_Build	ARRAY [1..10] OF CHAR or STRING [10]
Modul_Type	ARRAY [1..10] OF CHAR or STRING [10]
Modul_SW_Version	ARRAY [1..6] OF CHAR or STRING [6]

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 161: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_Ident.Type	Octet 1 – 14	Type of easy basic unit
Easy_Ident.SW_Version	Octet 15 – 24	Firmware version of the easy basic unit ¹⁾
Easy_Ident.SW_Build	Octet 25 – 34	Firmware version build number of the easy basic unit ²⁾
Easy_Ident.Modul_Type	Octet 35 – 44	Type of communication module
Easy_Ident.Modul_SW_Version	Octet 45 – 50	Firmware version of the communication module.

- 1) Cannot be determined for easy800/MFD. The corresponding octets will therefore contain the text "n.a." (not available).
- 2) Cannot be determined for easy600/800/MFD. The corresponding octets will therefore contain the text "n.a." (not available).

All data contents are coded as ASCII strings.

Read/write mode**“Easy operating mode” object**

This object contains the current operating mode of an easy basic unit (Run/Stop).

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query (Read) the current operating mode (Run/Stop) of the easy basic unit or transfer a new operating mode (Write) to the easy basic unit.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 100.

Length of the object

The length of the data to be read or written is 1 octet. When calling the “Write” service do not enter any other length or enter a small length for the “Read” service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_Mode	BYTE or USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 162: Address location and meaning of the data

Variable access (example)	Data position	Designation
Easy_Mode	Octet 1	Operating mode of the easy basic unit → table 163

Table 163: Coding of the operating mode

Operating mode code (hex)	Meaning
00	"Stop" mode
01	"Run" mode

Read/write image**"Easy 600 inputs" object**

This object contains the current status of the easy600 inputs.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the digital inputs (I1 to I16) and the current analog value inputs I7 and I8 of the easy600.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 197.

Length of the object

The data to be read is 4 octets in length. Do not enter a smaller length for the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variables (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_600_Inputs	STRUCT
Analog_7	BYTE or USINT
Analog_8	BYTE or USINT
Digital	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 164: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Inputs.Analog_7	Octet 1	Analog value of I7
Easy_600_Inputs.Analog_8	Octet 2	Analog value of I8
Easy_600_Inputs.Digital[1]	Octet 3	Bit 0 Status I1
Easy_600_Inputs.Digital[2]		Bit 1 Status I2
Easy_600_Inputs.Digital[3]		Bit 2 Status I3
Easy_600_Inputs.Digital[4]		Bit 3 Status I4
Easy_600_Inputs.Digital[5]		Bit 4 Status I5
Easy_600_Inputs.Digital[6]		Bit 5 Status I6
Easy_600_Inputs.Digital[7]		Bit 6 Status I7
Easy_600_Inputs.Digital[8]		Bit 7 Status I8
Easy_600_Inputs.Digital[9]	Octet 4	Bit 0 Status I9
Easy_600_Inputs.Digital[10]		Bit 1 Status I10
Easy_600_Inputs.Digital[11]		Bit 2 Status I11
Easy_600_Inputs.Digital[12]		Bit 3 Status I12
		Bit 4 Not used
		Bit 5 Not used
		Bit 6 Status I15
		Bit 7 Status I16
Easy_600_Inputs.Digital[15]		
Easy_600_Inputs.Digital[16]		



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 600 outputs and M markers" object

This object contains the current status of the outputs, markers and text markers of the easy600.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the digital outputs (Q1 to Q8), the markers (M1 to M16) and the text markers (D1 to D8) of the easy600.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 198.

Length of the object

The data to be read is 4 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_600_Marker_Outputs	ARRAY [1..32] OF BOOL or ARRAY [1..4] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 165: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Marker_Outputs[1]	Octet 1	Bit 0
Easy_600_Marker_Outputs[2]		Bit 1
Easy_600_Marker_Outputs[3]		Bit 2
Easy_600_Marker_Outputs[4]		Bit 3
Easy_600_Marker_Outputs[5]		Bit 4
Easy_600_Marker_Outputs[6]		Bit 5
Easy_600_Marker_Outputs[7]		Bit 6
Easy_600_Marker_Outputs[8]		Bit 7
Easy_600_Marker_Outputs[9]	Octet 2	Bit 0
Easy_600_Marker_Outputs[10]		Bit 1
Easy_600_Marker_Outputs[11]		Bit 2
Easy_600_Marker_Outputs[12]		Bit 3
Easy_600_Marker_Outputs[13]		Bit 4
Easy_600_Marker_Outputs[14]		Bit 5
Easy_600_Marker_Outputs[15]		Bit 6
Easy_600_Marker_Outputs[16]		Bit 7

Variable access (example)	Data position		Meaning
Easy_600_Marker_Outputs[17]	Octet 3	Bit 0	Status Q1
Easy_600_Marker_Outputs[18]		Bit 1	Status Q2
Easy_600_Marker_Outputs[19]		Bit 2	Status Q3
Easy_600_Marker_Outputs[20]		Bit 3	Status Q4
Easy_600_Marker_Outputs[21]		Bit 4	Status Q5
Easy_600_Marker_Outputs[22]		Bit 5	Status Q6
Easy_600_Marker_Outputs[23]		Bit 6	Status Q7
Easy_600_Marker_Outputs[24]		Bit 7	Status Q8
Easy_600_Marker_Outputs[25]	Octet 4	Bit 0	Status D1
Easy_600_Marker_Outputs[26]		Bit 1	Status D2
Easy_600_Marker_Outputs[27]		Bit 2	Status D3
Easy_600_Marker_Outputs[28]		Bit 3	Status D4
Easy_600_Marker_Outputs[29]		Bit 4	Status D5
Easy_600_Marker_Outputs[30]		Bit 5	Status D6
Easy_600_Marker_Outputs[31]		Bit 6	Status D7
Easy_600_Marker_Outputs[32]		Bit 7	Status D8

→ A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 600 function relay" object

This object contains the current status of the easy600 function relays.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the timing relays (T1 to T8), counters (C1 to C8), 7-day time switches (T1 to T4) and the analog value comparators (A1 to A8) of the easy600 .

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 199.

Length of the object

The data to be read is 4 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_600_Function_Relay	ARRAY [1..32] OF BOOL or ARRAY [1..4] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 166: Address location and meaning of the data

Variable access (example)	Octet	Data position	Meaning
Easy_600_Function_Relay[1]	Octet 1	Bit 0	Status T1
Easy_600_Function_Relay[2]		Bit 1	Status T2
Easy_600_Function_Relay[3]		Bit 2	Status T3
Easy_600_Function_Relay[4]		Bit 3	Status T4
Easy_600_Function_Relay[5]		Bit 4	Status T5
Easy_600_Function_Relay[6]		Bit 5	Status T6
Easy_600_Function_Relay[7]		Bit 6	Status T7
Easy_600_Function_Relay[8]		Bit 7	Status T8
Easy_600_Function_Relay[9]	Octet 2	Bit 0	Status C1
Easy_600_Function_Relay[10]		Bit 1	Status C2
Easy_600_Function_Relay[11]		Bit 2	Status C3
Easy_600_Function_Relay[12]		Bit 3	Status C4
Easy_600_Function_Relay[13]		Bit 4	Status C5
Easy_600_Function_Relay[14]		Bit 5	Status C6
Easy_600_Function_Relay[15]		Bit 6	Status C7
Easy_600_Function_Relay[16]		Bit 7	Status C8

Variable access (example)	Data position	Meaning
Easy_600_Function_Relay[17]	Octet 3	Bit 0 Status B1
Easy_600_Function_Relay[18]		Bit 1 Status B2
Easy_600_Function_Relay[19]		Bit 2 Status B3
Easy_600_Function_Relay[20]		Bit 3 Status B4
		Bit 4 Not used
		Bit 5 Not used
		Bit 6 Not used
		Bit 7 Not used
Easy_600_Function_Relay[25]	Octet 4	Bit 0 Status A1
Easy_600_Function_Relay[26]		Bit 1 Status A2
Easy_600_Function_Relay[27]		Bit 2 Status A3
Easy_600_Function_Relay[28]		Bit 3 Status A4
Easy_600_Function_Relay[29]		Bit 4 Status A5
Easy_600_Function_Relay[30]		Bit 5 Status A6
Easy_600_Function_Relay[31]		Bit 6 Status A7
Easy_600_Function_Relay[32]		Bit 7 Status A8

→ A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 600 pushbuttons" object

This object contains the current status of the easy600 pushbuttons.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the pusbuttons (P1 to P4, ESC, DEL, ALT, OK) of the easy600.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 200.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_600_Pushbuttons	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 167: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Pushbuttons[1]	Octet 1	Bit 0 Status P1
Easy_600_Pushbuttons[2]		Bit 1 Status P2
Easy_600_Pushbuttons[3]		Bit 2 Status P3
Easy_600_Pushbuttons[4]		Bit 3 Status P4
Easy_600_Pushbuttons[5]		Bit 4 Status ESC
Easy_600_Pushbuttons[6]		Bit 5 Status OK
Easy_600_Pushbuttons[7]		Bit 6 Status DEL
Easy_600_Pushbuttons[8]		Bit 7 Status ALT



A set bit corresponds to an actuated (depressed) pushbutton, an unset bit corresponds to an unactuated pushbutton.

"Easy Link input data" object

This object contains the current status of the input data on the Easy-Link of an easy basic unit.

This object can be read (Read) and written (Write). It can only be accessed by a class 2 DPV1 master.

Use

Use this object to

- to query the status of the Easy-Link input data (R1 to R16) that is transferred by the class 1 DP master in the cyclical DP data
- transfer new input data on the Easy-Link to the connected easy basic unit.



You can only write to this object if there is **no** class 1 DP master is running a cyclical data exchange with the addressed EASY204-DP.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 98.

Length of the object

The length of the data to be read or written is 2 octets.

Ensure that a different length is not entered when the "Write" service is called, and that a shorter length is entered when the "Read" service is called. Otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_Link_Inputs	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 168: Address location and meaning of the data

Variable access (example)	Data position	Meaning	
Easy_Link_Inputs[1]	Octet 1	Bit 0 Status R1	
Easy_Link_Inputs[2]		Bit 1 Status R2	
Easy_Link_Inputs[3]		Bit 2 Status R3	
Easy_Link_Inputs[4]		Bit 3 Status R4	
Easy_Link_Inputs[5]		Bit 4 Status R5	
Easy_Link_Inputs[6]		Bit 5 Status R6	
Easy_Link_Inputs[7]		Bit 6 Status R7	
Easy_Link_Inputs[8]		Bit 7 Status R8	
Easy_Link_Inputs[9]		Octet 2	Bit 0 Status R9
Easy_Link_Inputs[10]			Bit 1 Status R10
Easy_Link_Inputs[11]			Bit 2 Status R11
Easy_Link_Inputs[12]			Bit 3 Status R12
Easy_Link_Inputs[13]			Bit 4 Status R13
Easy_Link_Inputs[14]			Bit 5 Status R14
Easy_Link_Inputs[15]			Bit 6 Status R15
Easy_Link_Inputs[16]			Bit 7 Status R16



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy Link output data" object

This object contains the current status of the output data on the Easy-Link of an easy basic unit.

This object can only be read (Read). It can only be accessed by a class 2 DPV1 master.

Use

Use this object to query the current status of the Easy-Link output data (S1 to S8) that is transferred by the class 1 DP master in the cyclical DP data.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 99.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_Link_Outputs	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 169: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_Link_Outputs[1]	Octet 1	Bit 0	Status S1
Easy_Link_Outputs[2]		Bit 1	Status S2
Easy_Link_Outputs[3]		Bit 2	Status S3
Easy_Link_Outputs[4]		Bit 3	Status S4
Easy_Link_Outputs[5]		Bit 4	Status S5
Easy_Link_Outputs[6]		Bit 5	Status S6
Easy_Link_Outputs[7]		Bit 6	Status S7
Easy_Link_Outputs[8]		Bit 7	Status S8



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

Read/write function block data “Easy 600 timing relay parameters T1 to T8” objects

These 8 objects contain the parameters of the 8 timing relays of the easy600.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current parameters of the timing relays (T1 to T8) of an easy600
- transfer new parameters for the timing relays of an easy600.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 211 to 218 for the timing relays T1 to T8.

Length of objects

The data to be read is 6 octets in length. Do not enter a smaller length when calling the “Read” service, otherwise this will generate an error message.

The data to be written is 5 octets in length. Do not enter any different length when calling the “Write” service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read the parameters of a timing relay:

Name	Data type
Easy_600_Timer_Act_Val	STRUCT
Function	BYTE or USINT
Time range	BYTE or USINT
Parameter_menu	BYTE or USINT
Control status	BYTE or USINT
Act_Val	WORD or UINT

Declare the following variable (data function block) in an IEC 61131-3 based system in order to write the parameters of a timing relay:

Name	Data type
Easy_600_Timer_Setpoint	ARRAY [1..5] OF BYTE or ARRAY [1..5] OF USINT

Data contents of the objects

The following tables shows the address location and meaning of the data contents of the objects when reading the parameters of a timing relay. It also illustrates in an example how to access the data using the defined variable.

Table 170: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Timer_Act_Val.Function	Octet 1	Function of the timing relay → table 172
Easy_600_Timer_Act_Val.Time_Range	Octet 2	Time range of the timing relay → table 173
Easy_600_Timer_Act_Val.Parameter_Menu	Octet 3	Timing relay in the Parameters menu → table 174
Easy_600_Timer_Act_Val.Controlstatus	Octet 4	Control status of the timing relay → table 175
Easy_600_Timer_Act_Val.TimeValue	Octet 5, 6	Time value of the timing relay [1/100 s] with a time range in seconds [s] with a time range in minutes: seconds [min] with a time range in hours: minutes



When accessing the data of the "timing relay time value" observe the Motorola coding format (octet N: High byte, octet N+1: Low byte) used in PROFIBUS-DP. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

The following tables shows the address location and meaning of the data contents of the objects when writing the parameters of a timing relay. It also illustrates in an example how to access the data using the defined variable.

Table 171: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Timer_Setpoint[1]	Octet 1	Function of the timing relay → table 172
Easy_600_Timer_Setpoint[2]	Octet 2	Time range of the timing relay → table 173
Easy_600_Timer_Setpoint[3]	Octet 3	Timing relay in the Parameters menu → table 174
Easy_600_Timer_Setpoint[4]	Octet 4	Time setpoint of the timing relay
Easy_600_Timer_Setpoint[5]	Octet 5	→ table 176

Table 168 shows the coding of the data contents for the function of the timing relay.

Table 172: Function of the timing relay

Value (hex)	Function
00	On-delayed
01	Off-delayed
02	On-delayed with random switching
03	Off-delayed with random switching
04	Single pulse
05	Flashing

Table 173: Time range of the timing relay

Value (hex)	Time range
00	Seconds
01	Minutes:Seconds
02	Hours:Seconds

Table 174: Parameters menu of the timing relay

Value (hex)	Parameters menu
00	Timing relay appears in the Parameters menu
01	Timing relay does not appear in the Parameters menu

Table 175: Control status of the timing relay

Value (hex)	Control status
00	Timing relay not used by operating system
01	Timing relay used by operating system

Table 176: Time setpoint of the timing relay

Data position		Meaning	Explanation
Octet 4	Bit 4 to 7	Value of least significant unit, tens digit	1/100 seconds (00 to 99) with seconds time range; Seconds (00 to 59)
	Bit 0 to 3	Value of least significant unit, ones digit	with minutes time range:Seconds; Minutes (00 to 59) with hours time range:hours minutes
Octet 5	Bit 4 to 7	Value of most significant unit, tens digit	Seconds (00 to 99) with seconds time range; Minutes (00 to 99)
	Bit 0 to 3	Value of most significant unit, ones digit	with minutes:seconds time range; Hours (00 to 99) with hours:minutes time range

"Easy 600 counters C1 to C8 parameters" objects

These 8 objects contain the parameters of the 8 counters of the easy600.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current parameters of the counters (C1 to C8) of an easy600
- transfer new parameters for the counters of an easy600

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 221 to 228 for the counters C1 to C8.

Length of objects

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

The data to be written is 3 octets in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read the parameters of a counter:

Name	Data type
Easy_600_Counter_Act_Val	WORD or UINT

Declare the following variable (data function block) in an IEC 61131-3 based system in order to write the parameters of a counter:

Name	Data type
Easy_600_Counter_Setpoint	STRUCT
Counter value	WORD or UINT
Parameter_menu	BYTE or USINT

Data contents of the objects

The following tables shows the address location and meaning of the data contents of the objects when reading the parameters of a counter. It also illustrates in an example how to access the data using the defined variable.

Table 177: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Counter_Act_Val	Octet 1 and 2	Actual value of counter



When accessing the data of the “timing relay time value” observe the Motorola coding format (octet N: High byte, octet N+1: Low byte) used in PROFIBUS-DP. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

The following tables shows the address location and meaning of the data contents of the objects when writing the parameters of a counter. It also illustrates in an example how to access the data using the defined variable.

Table 178: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_Counter_Setpoint.CounterValue	Octet 1 and 2	Setpoint of the counter
Easy_600_Counter_Setpoint.Parameter_Menu	Octet 3	Counter in the Parameters menu → table 179



When accessing the data of the “timing relay time value” observe the Motorola coding format (octet N: High byte, octet N+1: Low byte) used in PROFIBUS-DP. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

Table 179: Parameters menu of the counter

Value (hex)	Parameters menu
00	Counter in the Parameters menu
01	Counter should not appear in the Parameters menu

"Easy 600 analog value comparator A1 to A8 parameter" objects

These eight objects contain the respective parameters of the eight analog value comparators of the easy600.

These objects can only be written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to transfer new parameters for the analog value comparators of an easy600

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 231 to 238 for the analog value comparators A1 to A8.

Length of objects

The data to be written is 3 octets in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system in order to write the parameters of an analog value comparator:

Name	Data type
Easy_600_AnalogVal_Cmp	ARRAY [1..3] OF BYTE or ARRAY [1..3] OF USINT

Data contents of the objects

The following tables shows the address location and meaning of the data contents of the objects when writing the parameters of an analog value comparator. It also illustrates in an example how to access the data using the defined variable.

Table 180: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_600_AnalogVal_Cmp[1]	Octet 1	Function of the analog value comparator → table 181
Easy_600_AnalogVal_Cmp[2]	Octet 2	Comparison value
Easy_600_AnalogVal_Cmp[3]	Octet 3	Analog value comparator in the Parameters menu → table 182

Table 181: Function of the analog value comparator

Value (hex)	Function
00	Comparison between I7 \leq I8
01	Comparison between I7 \geq I8
02	Comparison between I7 \leq comparison value
03	Comparison between I7 \geq comparison value
04	Comparison between I8 \leq comparison value
05	Comparison between I8 \geq comparison value

Table 182: Parameters menu of the analog value comparator

Value (hex)	Parameters menu
00	Counter in the Parameters menu
01	Counter should not appear in the Parameters menu

"Easy 600 ④1 channel A to 7-day time switch ④4 channel D parameter" objects

These sixteen objects contain the respective parameters of the four channels of the four 7-day time switches of the easy600.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current parameters of the 7-day time switch channels (④1 to ④4) of an easy600
- transfer new parameters for the channels of the easy600 7-day time switches

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 239 to 242 for channel A to channel D of the 7-day time switch ④1
- Index is 243 to 246 for channel A to channel D of the 7-day time switch ④2
- Index is 247 to 250 for channel A to channel D of the 7-day time switch ④1
- Index is 251 to 254 for channel A to channel D of the 7-day time switch ④1

Length of objects

The length of the data to be read or written is 8 octets. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read and write the parameters of 7-day time switch channel:

Name	Data type
Easy_600_Channel_TimeSwitch	ARRAY [1..8] OF BYTE or ARRAY [1..8] OF USINT

Data contents of the objects

The following tables shows the address location and meaning of the data contents of the objects when writing the parameters of a 7-day time switch channel. It also illustrates in an example how to access the data using the defined variable.

Table 183: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_600_Channel_TimeSwitch[1]	Octet 1		Weekday starting → table 184
Easy_600_Channel_TimeSwitch[2]	Octet 2		Weekday ending → table 184
Easy_600_Channel_TimeSwitch[3]	Octet 3	Bit 4 to 7	Switch on minute tens digit
		Bit 0 to 3	Switch on minute ones digit
Easy_600_Channel_TimeSwitch[4]	Octet 4	Bit 4 to 7	Switch on hour tens digit
		Bit 0 to 3	Switch on hour ones digit
Easy_600_Channel_TimeSwitch[5]	Octet 5	Bit 4 to 7	Switch off minute tens digit
		Bit 0 to 3	Switch off minute ones digit
Easy_600_Channel_TimeSwitch[6]	Octet 6	Bit 4 to 7	Switch off hour tens digit
		Bit 0 to 3	Switch off hour ones digit
Easy_600_Channel_TimeSwitch[7]	Octet 7		Switch time → table 185
Easy_600_Channel_TimeSwitch[8]	Octet 8		Channel of the time switch in the Parameters menu → table 186

Table 184: Weekday of the channel of a 7-day time switch

Value (hex)	Meaning
00	None set
01	Monday
02	Tuesday
03	Wednesday
04	Thursday
05	Friday
06	Saturday
07	Sunday

Table 185: Switch time of the channel of a 7-day time switch

Value (hex)	Meaning
00	Switch time On < Off
01	Switch time On > Off

Table 186: Parameters menu of the channel of a 7-day time switch

Value (hex)	Meaning
00	Channel of the time switch appears in the Parameters menu
01	Channel of the time switch does not appear in the Parameters menu

Read/write date and time "Easy clock" object

This object contains date and time of the real-time clock of an easy basic unit.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read out the current time and the current date of an easy600
- set the time and date of an easy600.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 97.

Length of the object

The length of the data to be read or written is 8 octets. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read and write the parameters of the real-time clock:

Name	Data type
Easy_Clock	DATE_AND_TIME

Data content of the object

The following table shows the address location and meaning of the data contained in the object when reading and writing. It also illustrates in an example how to access the data using the defined variable.

Table 187: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_Clock	Octet 1 – 8	Time and date of easy → table 188

Table 188: BCD coding of the time and date data

Data position		Meaning	Explanation
Octet 1	Bit 4 to 7	Year number, tens digit	Value 90 same as 1990 Value 99 same as 1999
	Bit 0 to 3	Year number, ones digit	Value 00 same as 2000 Value 89 same as 2089
Octet 2	Bit 4 to 7	Month number, tens digit	
	Bit 0 to 3	Month number, ones digit	
Octet 3	Bit 4 to 7	Day number, tens digit	
	Bit 0 to 3	Day number, ones digit	
Octet 4	Bit 4 to 7	Hour number, tens digit	
	Bit 0 to 3	Hour number, ones digit	
Octet 5	Bit 4 to 7	Month number, tens digit	
	Bit 0 to 3	Month number, ones digit	
Octet 6	Bit 4 to 7	Seconds number, tens digit	
	Bit 0 to 3	Seconds number, ones digit	

Data position		Meaning	Explanation
Octet 7	Bit 4 to 7	Milliseconds number, hundreds digit	
	Bit 0 to 3	Milliseconds number, tens digit	
Octet 8	Bit 4 to 7	Milliseconds number, ones digit	
	Bit 0 to 3	Weekday	Value 1 corresponds to Sunday Value 2 corresponds to Monday Value 7 corresponds to Saturday



The time in the easy600 does not internally use year, month, day and milliseconds. The corresponding data fields are therefore assigned with "0" when the object is read. The values in these data fields are ignored when the object is written.

Read/write DST**Easy 600 DST setting objects**

This object contains the DST setting of the easy600.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read the current DST setting of an easy600 clock.
- set the current DST setting of an easy600 clock.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 210.

Length of the object

The length of the data to be read or written is 1 octet. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read and write the DST setting:

Name	Data type
Easy_600_Clock_Setting	BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object when reading and writing. It also illustrates in an example how to access the data using the defined variable.

Table 189: Address location and meaning of the data

Variable access (example)	Data position	Designation
Easy_600_Clock_Setting	Octet 1	Setting of the easy clock to DST → table 190

Table 190: Coding of the easy600 DST setting

Value (hex)	Effect
00	Easy is / will be set to winter time
01	Easy is / will be set to summer time

11 Process Data for easy700 (DPV1)

The PROFIBUS-DPV1 functionality of the EASY204-DP (from device version 07) allows you to access the following data ranges of an easy700 without any extensive programming required via a class 1 or class 2 DPV1 master:

- Identification
- Mode
- Image
- Function blocks
- Date and time
- Summer/winter time

The data from these ranges is located in Process Data Objects (data records) and can be read and/or written with the DPV1 “Read” and “Write” services.

Object overview

The following overview shows all the objects contained in the EASY204-DP that are available for a connected easy700.



The attributes API, Slot Number and Index form the address information for the DP master in order to access an object. Refer to the following section for further information about this.

Table 191: Process data objects EASY204-DP for easy700

Object name	API	Slot Number	Index	Data length (octets)	Read (R) Write (W)	→ Page
Easy operating mode	0	0	100	1	R/W	247
Easy identification	0	1	100	50	R	245
Image data						
EASY 700/800 inputs (I1 – I16)	0	1	250	2	R	286
Easy 700/800 analog inputs IA1 to IA4	0	1	197 – 200	2	R	288
Easy 700/800 outputs (Q1 – Q8)	0	1	251	1	R	290
Easy700/800 pushbuttons	0	1	249	1	R	292
Easy 700 analog value comparators	0	2	1	2	R	294
Easy 700 7-day time switches	0	2	2	1	R	296
Easy 700 year time switches	0	2	3	1	R	298
Easy 700 master reset	0	2	4	1	R	300
Easy 700 text function blocks	0	2	5	2	R	302
Easy 700 timing relays	0	2	6	2	R	304
Easy 700 counters	0	2	7	2	R	306
Easy 700 operating hours counters	0	2	8	1	R	308

Object name	API	Slot Number	Index	Data length (octets)	Read (R) Write (W)	→ Page
Easy 700 actual value M markers	0	2	11	2	R	310
Easy 700 actual value N markers	0	2	12	2	R	312
Easy 700 new value markers M1 to M16	0	2	101 – 116	1	W	316
Easy 700 new value markers N1 to N16	0	0	117 – 132	1	W	316
Easy Link input data (R1 – R16)	0	0	98	2	R/W	259
Easy Link output data (S1 – S8)	0	0	99	1	R	261

Function blocks

Easy 700/800 write data function block	0	1	97	8	W	323
Easy 700/800 selection of the data function block	0	1	99	4	W	319
Easy 700/800 read data function block	0	1	98	4	R	321
Easy clock	0	0	97	8	R/W	276
Easy 700 DST setting	0	2	9	16	R/W	338

Accessing objects

The Process Data Objects in the EASY204-DP are accessed via the DPV1 “Read” and “Write” services using the functions provided for this by the DP master system. Refer to the documentation of the manufacturer concerning this. Function blocks are normally used to carry out the access. In IEC 61131-3 based systems, the function blocks RDREC (Read) and WRREC (Write) defined by the PROFIBUS Users' Organisation in regulation 2.182 are normally made available. These also enable the access of complex data structures if required.

The following is required to access the objects:

- The address of the local DPV1 master interface
- The station address of the EASY204-DP to be accessed
- The application identifier (API) in the EASY204-DP (only required for class 2 DPV1 masters)
- The module to be addressed in EASY204-DP (Slot Number)
- The address (Index) of the required Process Data Object in the addressed EASY204-DP module
- The data length of the required Process Data Object
- A defined variable (memory range) in the local application that is to be assigned to the read data or containing the data to be written.

Refer to the topology of your master system for the address of the local DPV1 master interface. Refer to the PROFIBUS topology for the station address of the EASY204-DP.

The API (only required for class 2 DPV1 masters) must be stated for all Process Data Objects.

- The slot number, the index, the data length and a sample variable definition (declaration) for an IEC 61131-3 based system is provided in the following sections for each of the Process Data Objects provided in the EASY-204-DP for a connected easy700.

Read identification**"Easy identification" object**

The description of this object is identical to the easy600 (DPV1), → page 245.

Read/write mode**"Easy mode" object**

The description of this object is identical to the easy600 (DPV1), → page 247.

Read/write image

"Easy 700/800 inputs" object

This object contains the current status of the inputs of an easy700 or an easy800/MFD.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the digital inputs (I1 to I16) of an easy700 or easy800/MFD.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 250.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_Inputs	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 192: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_700_800_Inputs[1]	Octet 1	Bit 0	Status I1
Easy_700_800_Inputs[2]		Bit 1	Status I2
Easy_700_800_Inputs[3]		Bit 2	Status I3
Easy_700_800_Inputs[4]		Bit 3	Status I4
Easy_700_800_Inputs[5]		Bit 4	Status I5
Easy_700_800_Inputs[6]		Bit 5	Status I6
Easy_700_800_Inputs[7]		Bit 6	Status I7
Easy_700_800_Inputs[8]		Bit 7	Status I8
Easy_700_800_Inputs[9]	Octet 2	Bit 0	Status I9
Easy_700_800_Inputs[10]		Bit 1	Status I10
Easy_700_800_Inputs[11]		Bit 2	Status I11
Easy_700_800_Inputs[12]		Bit 3	Status I12
Easy_700_800_Inputs[13]		Bit 4	Status I13
Easy_700_800_Inputs[14]		Bit 5	Status I14
Easy_700_800_Inputs[15]		Bit 6	Status I15
Easy_700_800_Inputs[16]		Bit 7	Status I16



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700/800 analog inputs IA1 to IA4" objects

These objects contain the current status of the analog inputs of an easy700 or an easy800/MFD.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the analog inputs (IA1 to IA4) of an easy700 or easy800/MFD.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 197 to 200 for the analog inputs IA1 to IA4.

Length of objects

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_Analog_I	Word or UINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 193: Address location and meaning of the data of the objects

Variable access (example)	Data position	Meaning
Easy_700_800_Analog_I	Octet 1 and 2	Value of the analog input IA



When accessing the data of the "value of the analog input IA" observe the Motorola coding format (octet N: High byte, octet N+1: Low byte) used in PROFIBUS-DP. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

"Easy 700/800 outputs" object

This object contains the current status of the outputs of an easy700 or an easy800/MFD.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the digital outputs (Q1 to Q8) of an easy700 or easy800/MFD.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 251.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_Outputs	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 194: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_700_800_Outputs[1]	Octet 1	Bit 0	Status Q1
Easy_700_800_Outputs[2]		Bit 1	Status Q2
Easy_700_800_Outputs[3]		Bit 2	Status Q3
Easy_700_800_Outputs[4]		Bit 3	Status Q4
Easy_700_800_Outputs[5]		Bit 4	Status Q5
Easy_700_800_Outputs[6]		Bit 5	Status Q6
Easy_700_800_Outputs[7]		Bit 6	Status Q7
Easy_700_800_Outputs[8]		Bit 7	Status Q8



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700/800 pushbuttons" object

This object contains the current status of the pushbuttons of an easy700 or an easy800/MFD.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the pushbuttons (P1 to P4) of an easy700 or easy800/MFD.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 249.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_Pushbuttons	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 195: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_800_Pushbuttons[1]	Octet 1	Bit 0
Easy_700_800_Pushbuttons[2]		Bit 1
Easy_700_800_Pushbuttons[3]		Bit 2
Easy_700_800_Pushbuttons[4]		Bit 3
		Bit 4
		Bit 5
		Bit 6
		Bit 7



A set bit corresponds to an actuated (depressed) pushbutton, an unset bit corresponds to an unactuated pushbutton.

"Easy 700 analog value comparators" object

This object contains the current status of the analog value comparators of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the analog value comparators (A1 to A16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 1.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_AnalogVal_Cmp	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 196: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_700_AnalogVal_Cmp[1]	Octet 1	Bit 0	Status A1
Easy_700_AnalogVal_Cmp[2]		Bit 1	Status A2
Easy_700_AnalogVal_Cmp[3]		Bit 2	Status A3
Easy_700_AnalogVal_Cmp[4]		Bit 3	Status A4
Easy_700_AnalogVal_Cmp[5]		Bit 4	Status A5
Easy_700_AnalogVal_Cmp[6]		Bit 5	Status A6
Easy_700_AnalogVal_Cmp[7]		Bit 6	Status A7
Easy_700_AnalogVal_Cmp[8]		Bit 7	Status A8
Easy_700_AnalogVal_Cmp[9]	Octet 2	Bit 0	Status A9
Easy_700_AnalogVal_Cmp[10]		Bit 1	Status A10
Easy_700_AnalogVal_Cmp[11]		Bit 2	Status A11
Easy_700_AnalogVal_Cmp[12]		Bit 3	Status A12
Easy_700_AnalogVal_Cmp[13]		Bit 4	Status A13
Easy_700_AnalogVal_Cmp[14]		Bit 5	Status A14
Easy_700_AnalogVal_Cmp[15]		Bit 6	Status A15
Easy_700_AnalogVal_Cmp[16]		Bit 7	Status A16



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 7-day time switches" object

This object contains the current status of the 7-day time switches of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the 7-day time switches (① to ⑧) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 2.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_7DayTimeSwitch	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 197: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_7DayTimeSwitch[1]	Octet 1	Bit 0 Status @1
Easy_700_7DayTimeSwitch[2]		Bit 1 Status @2
Easy_700_7DayTimeSwitch[3]		Bit 2 Status @3
Easy_700_7DayTimeSwitch[4]		Bit 3 Status @4
Easy_700_7DayTimeSwitch[5]		Bit 4 Status @5
Easy_700_7DayTimeSwitch[6]		Bit 5 Status @6
Easy_700_7DayTimeSwitch[7]		Bit 6 Status @7
Easy_700_7DayTimeSwitch[8]		Bit 7 Status @8



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 year time switches" objects

This object contains the current status of the year time switches of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the year time switches (Y1 to Y8) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 3.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Year_TimeSwitch	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 198: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_Year_TimeSwitch[1]	Octet 1	Bit 0 Status Y1
Easy_700_Year_TimeSwitch[2]		Bit 1 Status Y2
Easy_700_Year_TimeSwitch[3]		Bit 2 Status Y3
Easy_700_Year_TimeSwitch[4]		Bit 3 Status Y4
Easy_700_Year_TimeSwitch[5]		Bit 4 Status Y5
Easy_700_Year_TimeSwitch[6]		Bit 5 Status Y6
Easy_700_Year_TimeSwitch[7]		Bit 6 Status Y7
Easy_700_Year_TimeSwitch[8]		Bit 7 Status Y8



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 master reset" object

This object contains the current status of the master reset of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the master reset (M1 to M4) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 4.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Master_Reset	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 199: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_700_Master_Reset[1]	Octet 1	Bit 0	Status Z1
Easy_700_Master_Reset[2]		Bit 1	Status Z2
Easy_700_Master_Reset[3]		Bit 2	Status Z3
Easy_700_Master_Reset[4]		Bit 3	Status Z4
		Bit 4	Not used
		Bit 5	Not used
		Bit 6	Not used
		Bit 7	Not used



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 text function blocks" object

This object contains the current status of the text function blocks of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the text function blocks (D1 to D16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 5.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_TextFB	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 200: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_700_TextFB[1]	Octet 1	Bit 0	Status D1
Easy_700_TextFB[2]		Bit 1	Status D2
Easy_700_TextFB[3]		Bit 2	Status D3
Easy_700_TextFB[4]		Bit 3	Status D4
Easy_700_TextFB[5]		Bit 4	Status D5
Easy_700_TextFB[6]		Bit 5	Status D6
Easy_700_TextFB[7]		Bit 6	Status D7
Easy_700_TextFB[8]		Bit 7	Status D8
Easy_700_TextFB[9]		Octet 2	Status D9
Easy_700_TextFB[10]		Bit 0	Status D10
Easy_700_TextFB[11]		Bit 1	Status D11
Easy_700_TextFB[12]		Bit 2	Status D12
Easy_700_TextFB[13]		Bit 3	Status D13
Easy_700_TextFB[14]		Bit 4	Status D14
Easy_700_TextFB[15]		Bit 5	Status D15
Easy_700_TextFB[16]		Bit 6	Status D16



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 timing relays" objects

This object contains the current status of the timing relays of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the timing relays (T1 to T16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 6.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Timers	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 201: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_Timer[1]	Octet 1	Bit 0
Easy_700_Timer[2]		Bit 1
Easy_700_Timer[3]		Bit 2
Easy_700_Timer[4]		Bit 3
Easy_700_Timer[5]		Bit 4
Easy_700_Timer[6]		Bit 5
Easy_700_Timer[7]		Bit 6
Easy_700_Timer[8]		Bit 7
Easy_700_Timer[9]	Octet 2	Bit 0
Easy_700_Timer[10]		Bit 1
Easy_700_Timer[11]		Bit 2
Easy_700_Timer[12]		Bit 3
Easy_700_Timer[13]		Bit 4
Easy_700_Timer[14]		Bit 5
Easy_700_Timer[15]		Bit 6
Easy_700_Timer[16]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 counters" object

This object contains the current status of the counters of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the counters (C1 to C16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 7.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Counter	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 202: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_Counter[1]	Octet 1	Bit 0
Easy_700_Counter[2]		Bit 1
Easy_700_Counter[3]		Bit 2
Easy_700_Counter[4]		Bit 3
Easy_700_Counter[5]		Bit 4
Easy_700_Counter[6]		Bit 5
Easy_700_Counter[7]		Bit 6
Easy_700_Counter[8]		Bit 7
Easy_700_Counter[9]	Octet 2	Bit 0
Easy_700_Counter[10]		Bit 1
Easy_700_Counter[11]		Bit 2
Easy_700_Counter[12]		Bit 3
Easy_700_Counter[13]		Bit 4
Easy_700_Counter[14]		Bit 5
Easy_700_Counter[15]		Bit 6
Easy_700_Counter[16]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 operating hours counters" object

This object contains the current status of the operating hours counters of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the operating hours counters (O1 to O4) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 8.

Length of the object

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Hours_of_Operation	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 203: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_Hours_of_Operation[1]	Octet 1	Bit 0
Easy_700_Hours_of_Operation[2]		Bit 1
Easy_700_Hours_of_Operation[3]		Bit 2
Easy_700_Hours_of_Operation[4]		Bit 3
		Bit 4
		Bit 5
		Bit 6
		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 actual value M markers" object

This object contains the current status of the M markers of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the M markers (M1 to M16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 11.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_M_Marker	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 204: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_M_Marker[1]	Octet 1	Bit 0
Easy_700_M_Marker[2]		Bit 1
Easy_700_M_Marker[3]		Bit 2
Easy_700_M_Marker[4]		Bit 3
Easy_700_M_Marker[5]		Bit 4
Easy_700_M_Marker[6]		Bit 5
Easy_700_M_Marker[7]		Bit 6
Easy_700_M_Marker[8]		Bit 7
Easy_700_M_Marker[9]	Octet 2	Bit 0
Easy_700_M_Marker[10]		Bit 1
Easy_700_M_Marker[11]		Bit 2
Easy_700_M_Marker[12]		Bit 3
Easy_700_M_Marker[13]		Bit 4
Easy_700_M_Marker[14]		Bit 5
Easy_700_M_Marker[15]		Bit 6
Easy_700_M_Marker[16]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 actual value N markers" object

This object contains the current status of the N markers of an easy700.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the N markers (N1 to N16) of an easy700.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 12.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_N_Marker	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 205: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_700_N_Marker[1]	Octet 1	Bit 0
Easy_700_N_Marker[2]		Bit 1
Easy_700_N_Marker[3]		Bit 2
Easy_700_N_Marker[4]		Bit 3
Easy_700_N_Marker[5]		Bit 4
Easy_700_N_Marker[6]		Bit 5
Easy_700_N_Marker[7]		Bit 6
Easy_700_N_Marker[8]		Bit 7
Easy_700_N_Marker[9]	Octet 2	Bit 0
Easy_700_N_Marker[10]		Bit 1
Easy_700_N_Marker[11]		Bit 2
Easy_700_N_Marker[12]		Bit 3
Easy_700_N_Marker[13]		Bit 4
Easy_700_N_Marker[14]		Bit 5
Easy_700_N_Marker[15]		Bit 6
Easy_700_N_Marker[16]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 700 new value markers M1 to M16" objects

These objects enable the status of the M markers of an easy700 to be controlled.

These objects can only be written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to control the current status of the M markers (M1 to M16) of an easy700.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 101 to 116 for the markers M1 to M16.

Length of objects

The data to be written is 1 octet in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Set_M_Marker	BYTE or USINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 206: Address location and meaning of the data of the objects

Variable access (example)	Data position	Meaning
Easy_700_Setp_M_Marker	Octet 1	Setpoint for M markers: 00 = FALSE FF = TRUE

"Easy 700 new value markers N1 to N16" objects

These objects enable the status of the N markers of an easy700 to be controlled.

These objects can only be written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to control the current status of the N markers (N1 to N16) of an easy700.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 117 to 132 for the markers N1 to N16.

Length of objects

The data to be written is 1 octet in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_Setp_N_Marker	BYTE or USINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 207: Address location and meaning of the data of the objects

Variable access (example)	Data position	Meaning
Easy_700_Setp_N_Marker	Octet 1	Setpoint for N markers: 00 = FALSE FF = TRUE

"Easy Link input data" object

The description of this object is identical to the easy600 (DPV1), → page 259.

"Easy Link output data" object

The description of this object is identical to the easy600 (DPV1), → page 261.

Read/write function block data Procedure

The following generic objects simplify the reading and writing of function block data:

- “Easy 700/800 select data function block”
- “Easy 700/800 read data function block”
- “Easy 700/800 write data function block”.

Two object accesses are required in order to read the data of a function block:

- Selection of the required function block data for the subsequent read operation:
→ “Easy 700/800 function block data selection” object, page 319.
- Reading of the selected function block data:
→ “Easy 700/800 read function block data” object, page 321.

Only one object access is required in order to write the data of a function block: The selection of the required function block data and the transfer of the new values for this are executed simultaneously.

→ “Easy 700/800 write function block data” object, page 323.

"Easy 700/800 function block data selection" object

This object is used to select the data of an easy700 function block that you later wish to read.

This object can only be written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to select the required data of an easy700 function block for one or several subsequent read accesses with the "Easy 700/800 read function block data" object. The selection is valid for all subsequent read accesses until you have selected new data by writing to the "Easy 700/800 function block data selection" object.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 99.

Length of the object

The data to be written is 4 octets in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_Ad_FB_Read	ARRAY [1..4] OF BYTE or ARRAY [1..4] OF USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 208: Address location and meaning of the data

Variable access (example)	Data position	Meaning	
Easy_700_800_Ad_FB_Read[1]	Octet 1	Comm and	8D _{hex} for easy700
			B2 _{hex} for easy800/MFD
Easy_700_800_Ad_FB_Read[2]	Octet 2	Type	→ Function Block Description
Easy_700_800_Ad_FB_Read[3]	Octet 3	Instance	
Easy_700_800_Ad_FB_Read[4]	Octet 4	Index	

Octet 2 to octet 4 (Type, Instance, Index) identify which function block data is to be read using the “Easy 700/800 read function block data” object. This information is specific for the function block data concerned and can be obtained from the function block description.

"Easy 700/800 read function block data" object

This object is used to read the required data of an easy700 function block.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to read the required data of an easy700 function block.



You must have selected the required data beforehand by writing the "Easy 700/800 select function block data" object.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 98.

Length of the object

The data to be read is 8 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_FB_Read	ARRAY [1..8] OF BYTE or ARRAY [1..8] OF USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 209: Address location and meaning of the data

Variable access (example)	Data position	Meaning	
Easy_700_800_FB_Read[1]	Octet 1	Command	89 _{hex} for easy700 92 _{hex} for easy800/MFD
Easy_700_800_FB_Read[2]	Octet 2	Type	→ Function Block Description
Easy_700_800_FB_Read[3]	Octet 3	Instance	
Easy_700_800_FB_Read[4]	Octet 4	Index	
Easy_700_800_FB_Read[5]	Octet 5	Data 1	
Easy_700_800_FB_Read[6]	Octet 6	Data 2	
Easy_700_800_FB_Read[7]	Octet 7	Data 3	
Easy_700_800_FB_Read[8]	Octet 8	Data 4	

Octet 2 to octet 4 (type, instance, index) identify which function block data was read.



This information is the same as the selection you have made by writing the "Easy 700/800 select function block data" object.

Octet 5 to octet 8 (Data 1 to Data 4) contain the actual value of the read function block data. The structure and meaning of these values are specific for the read function block data. Refer to the relevant description of the function block about this (from page 325).

"Easy 700/800 write function block data" object

Use this object to select the required data of an easy700 function block and transfer new values to it.

This object can only be written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to assign new values to the required data of an easy700 function block.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 97.

Length of the object

The data to be written is 8 octets in length. Do not enter any different length when calling the "Write" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_700_800_FB_Write	ARRAY [1..8] OF BYTE or ARRAY [1..8] OF USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 210: Address location and meaning of the data

Variable access (example)	Data position	Meaning	
Easy_700_800_FB_Write[1]	Octet 1	Comma nd	8D _{hex} for easy700
			B2 _{hex} for easy800/MFD
Easy_700_800_FB_Write[2]	Octet 2	Type	→ Function Block Description
Easy_700_800_FB_Write[3]	Octet 3	Instance	
Easy_700_800_FB_Write[4]	Octet 4	Index	
Easy_700_800_FB_Write[5]	Octet 5	Data 1	
Easy_700_800_FB_Write[6]	Octet 6	Data 2	
Easy_700_800_FB_Write[7]	Octet 7	Data 3	
Easy_700_800_FB_Write[8]	Octet 8	Data 4	

Octet 2 to octet 4 (type, instance, index) identify which function block data is to be written. The corresponding information is specific for the function block data concerned and is explained in the following sections of the relevant function block.

Octet 5 to octet 8 (Data 1 to Data 4) contain the actual value of the function block data to be written. The structure and meaning of these values are specific for the function block data concerned and are explained in the following sections of the relevant function block. Note the information given in these section concerning which function block data can be written under which conditions.

**Analog value comparator/threshold comparator:
A1 – A16**

Operand selection	Value (hex)
Type	8D
Instance ¹⁾	00 – 0F
Index	00 – 07, → table 211

- 1) The value 00_{hex} selects the analog value comparator A1,
the value 0F_{hex} the analog value comparator A16.

Table 211: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Parameters	→ table 212	–	–	–	R
01	Control byte	–	–	→ table 213	–	R
02	Comparison value I1	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
03	Comparison value I2	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
04	Gain factor for F1 (I1 = F1*I1)	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
05	Gain factor for I2 (I2 = F2*I2)	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
06	Offset OS (I1 = OS + actual value at I1)	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
07	Switching hysteresis HY for value I2	WORD or UINT ¹⁾	–	–	–	R/W ²⁾

- 1) Value transferred in Intel format: Data 1 contains Low byte, Data 2 contains High byte
2) The value can only be written if it is assigned to a constant in the program.

Table 212: Index 00 – Parameters

Meaning	Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Appears in the Parameters menu																	
Yes/no																	0/1
Compare																	
FB not used														0	0	0	
EQ (=)														0	0	1	
GE (\geq)														0	1	0	
LE (\leq)														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											

Table 213: Index 01 – Control byte (Data 3)

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

1) Status 1 if comparison condition is fulfilled.



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

Counters C1 – C16

Operand selection	Value (hex)
Type	8F
Instance ¹⁾	00 – 0F
Index	00 – 0, → table 2143

- 1) The value 00_{hex} selects the counter C1,
the value 0F_{hex} the counter C16.

Table 214: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Parameters	→ table 215	–	–	–	R
01	Control byte	→ table 216	–	–	–	R
02	Actual value	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
03	Counter setpoint 2	WORD or UINT ¹⁾	–	–	–	R/W ²⁾

- 1) Value transferred in Intel format: Data 1 contains Low byte, Data 2 contains High byte
 2) The value can only be written if it is assigned to a constant in the program.

Table 215: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the Parameters 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		–	–	–	–				

Table 216: Index 01 – Control byte

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

- 1) Contact type
- 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



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

Operating hours counters 01 – 04

Operand selection	Value (hex)
Type	92
Instance ¹⁾	00 – 03
Index	00 – 03, → table 217

- 1) The value 00_{hex} selects the operating hours counter 01,
the value 03_{hex} the operating hours counter 04.

Table 217: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Parameters	→ table 218	–	–	–	R
01	Control byte	→ table 219	–	–	–	R
02	Actual value	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
03	Counter setpoint 2	WORD or UINT ¹⁾	–	–	–	R/W ²⁾

- 1) Value transferred in Intel format: Data 1 contains Low byte, Data 2 contains High byte
 2) The value can only be written if it is assigned to a constant in the program.

Table 218: Index 00 – Parameters

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

Table 219: Index 01 – Control byte

Data 1	Bit	7	6	5	4	3	2	1	0
FB output		–	–	–	–	–	RE ³	EN ²	Q11

- 1) Contact type
- 2) Enable, the timing relay is started (trigger coil)
- 3) Reset, the timing relay is reset (reset coil)



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

Timing relays T1 – T16

Operand selection	Value (hex)
Type	92
Instance ¹⁾	00 – 0F
Index	00 – 04, → table 220

- 1) The value 00_{hex} selects the timing relay T1,
the value 0F_{hex} the timing relay T16.

Table 220: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Parameters	→ table 221	–	–	–	R
01	Control byte	→ table 222	–	–	–	R
02	Actual value 1	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
03	Time setpoint value 1	WORD or UINT ¹⁾	–	–	–	R/W ²⁾
	Time setpoint 2	WORD or UINT ¹⁾	–	–	–	R/W ²⁾

- 1) Value transferred in Intel format: Data 1 contains Low byte, Data 2 contains High byte
 2) The value can only be written if it is assigned to a constant in the program.

Table 221: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
Appears in the Parameters 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 generator					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								

Table 222: Index 01 – Control byte

	Bit	7	6	5	4	3	2	1	0
FB input/output Data 3		–	–	–	–	ST ⁴	RE ³	EN ²	Q11

- 1) Contact type
- 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)



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

Year time switches Y1 – Y8

Operand selection	Value (hex)
Type	A2
Instance ¹⁾	00 – 07
Index	→ table 223

- 1) The value 00_{hex} selects the year time switch Y1,
the value 07_{hex} the year time switch Y8.

Table 223: Operand overview

Index (hex)	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Parameters	→ table 224	–	–	–	R
01	Control byte → table 225	→ table 225	–	–	–	R
	Channel A					
11	ON time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
12	OFF time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
	Channel B					
21	ON time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
22	OFF time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
	Channel C					
31	ON time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
32	OFF time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
	Channel D					
41	ON time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾
42	OFF time	Day ¹⁾	Month ¹⁾	Year ²⁾	–	R/W ³⁾

1) Value is transferred in hexadecimal code

2) Value transferred in hexadecimal code; 00 stands for the year 2000

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

Table 224: Index 00 – Parameters

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

Table 225: Index 01 – Control byte

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

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



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

7-day time switches ④1 to ④8

Operand selection	Value (hex)
Type	A1
Instance ¹⁾	00 – 07
Index	→ table 226

1) The value 00_{hex} selects the 7-day time switch ④1, the value 07_{hex} the 7-day time switch ④8.

Table 226: Operand overview

Index (hex)	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Parameters	→ table 227	–	–	–	R
01	Control byte	→ table 228	–	–	–	R
	Channel A					
11	Day on/off	Day on ¹⁾	Day off ¹⁾	–	–	R/W ³⁾
12	On time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾
13	Off time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾
	Channel B					
21	Day on/off	Day on ¹⁾	Day off ¹⁾	–	–	R/W ³⁾
22	ON time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾
23	OFF time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾
	Channel C					
31	Day on/off	Day on ¹⁾	Day off ¹⁾	–	–	R/W ³⁾
32	ON time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾
33	OFF time	Hour ²⁾	Minute ²⁾	–	–	R/W ³⁾

Index (hex)	Operand	Data 1	Data 2	Data 3	Data 4	Read/ Write
	Channel D					
41	Day on/off	Day off ¹⁾	Month ¹⁾	–	–	R/W ³⁾
42	ON time	Minute ²⁾	Month ¹⁾	–	–	R/W ³⁾
43	OFF time	Minute ²⁾	Month ¹⁾	–	–	R/W ³⁾

- 1) Value is transferred in hexadecimal code, 01 corresponds to Sunday ... 07 corresponds to Saturday
- 2) Value is transferred in hexadecimal code
- 3) The value can only be written if it is assigned to a constant in the program.

Table 227: Index 00 – Parameters

Meaning	Bit	7	6	5	4	3	2	1	0
---------	-----	---	---	---	---	---	---	---	---

Appears in the Parameters menu

Channel A									0/1
Channel B								0/1	
Channel C							0/1		
Channel D					0/1				
Unused bits	–	–	–	–					

Table 228: Index 01 – Control byte

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

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



Further information on this function block is provided in the easy700 manual (AWB2528-1508G) or in the EASY-SOFT Help.

Read/write date and time "Easy clock" object

The description of this object is identical to the easy600 (DPV1), → page 276.

Read/write DST "Easy 700 DST setting" objects

This object contains the DST setting of the easy700.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- transfer the current DST setting of an easy700 clock.
- transfer a new DST setting of an easy700 clock.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 2
- Index is 9.

Length of the object

The length of the data to be read or written is 16 octets. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read and write the DST setting:

Name	Data type
Easy_700_Clock_DSTSetting	ARRAY [1..16] OF BYTE or ARRAY [1..16] OF USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object when reading and writing. It also illustrates in an example how to access the data using the defined variable.

Table 229: Address location and meaning of the data

Variable access (example)	Data position	Meaning	Value range (decimal)
Easy_700_Clock_DST[1]	Octet 1	Changeover condition → table 230	0 to 4
Easy_700_Clock_DST[2]	Octet 2	Rule 1 for changeover to summer time → table 231	0 to 5
Easy_700_Clock_DST[3]	Octet 3	Weekday for changeover to summer time → table 232	0 to 6
Easy_700_Clock_DST[4]	Octet 4	Rule 2 for changeover to summer time → table 233	0 to 2
Easy_700_Clock_DST[5]	Octet 5	Day of changeover for change to summer time	1 to 31 BCD coding
Easy_700_Clock_DST[6]	Octet 6	Month of changeover for change to summer time	1 to 12 BCD coding
Easy_700_Clock_DST[7]	Octet 7	Hour of changeover for change to summer time	0 to 23 BCD coding
Easy_700_Clock_DST[8]	Octet 8	Minute of changeover for change to summer time	0 to 59 BCD coding
Easy_700_Clock_DST[9]	Octet 9	Time difference → table 234	0 to 5

Variable access (example)	Data position	Meaning	Value range (decimal)
Easy_700_Clock_ DST[10]	Octet 10	Rule 1 for changeover to winter time → table 231	0 to 4
Easy_700_Clock_ DST[11]	Octet 11	Weekday for changeover to winter time → table 232	0 to 5
Easy_700_Clock_ DST[12]	Octet 12	Rule 2 for changeover to winter time → table 233	0 to 6
Easy_700_Clock_ DST[13]	Octet 13	Day of changeover for change to winter time	1 to 31 BCD coding
Easy_700_Clock_ DST[14]	Octet 14	Month of changeover for change to winter time	1 to 12 BCD coding
Easy_700_Clock_ DST[15]	Octet 15	Hour of changeover for change to winter time	0 to 23 BCD coding
Easy_700_Clock_ DST[16]	Octet 16	Minute of changeover for change to winter time	0 to 59 BCD coding

Octet 1 contains the changeover condition for the summer/winter changeover, table 230 shows the possible changeover options.

Table 230: Changeover conditions for DST change

Value (hex)	Meaning	Note
00	No changeover	Octets 2 to 16 have no meaning
01	Changeover is according to: • Octet 2 to 9 for changeover to summer time • Octet 9 to 16 for changeover to winter time	
02	Changeover according to rules for EU	Octets 2 to 16 have no meaning
03	Changeover according to rules for GB	
04	Changeover according to rules for USA	

If the value 0 is entered in octet 1, there is no DST changeover. Selecting the values 2 to 4 causes an automatic DST change according to the legal requirements for the selected country.

If the value 1 is entered in octet 1, the DST changeover takes place according to the user-defined rules entered in octet 2 to 16.

Octet 2 contains the first rule for the changeover to summer time, octet 10 the rule for the changeover to winter time. Table 231 indicates the possible rules.

Table 231: Rule 1 for DST changeover

Value (hex)	Meaning	
00	The changeover takes place at the time defined in...	<ul style="list-style-type: none">... • Octet 5 to 8 for changeover to summer time• Octet 13 to 16 for changeover to winter time <p>In this case without meaning:</p> <ul style="list-style-type: none">• Octet 3 to 4 for changeover to summer time• Octet 11 to 12 for changeover to summer time
01	The changeover takes place on the first weekday defined in...	<ul style="list-style-type: none">... • Octet 3 for changeover to summer time• Octet 11 for changeover to winter time <p>in conjunction with the defined rule 2 in:</p> <ul style="list-style-type: none">• Octet 4 for changeover to summer time• Octet 12 for changeover to winter time
02 ¹⁾	The changeover takes place on the second weekday defined in...	
03 ¹⁾	The changeover takes place on the third weekday defined in...	
04 ¹⁾	The changeover takes place on the fourth weekday defined in...	
05 ¹⁾	The changeover takes place on the last weekday defined in...	

1) The meaning "in" must be entered in rule 2. Rule 2 is defined in:

- Octet 4 for changeover to summer time
- Octet 12 for changeover to winter time

If octet 2 or octet 10 contains one of the values 1 to 5, the second rule must be defined for the changeover to summer time (octet 4) or winter time (octet 12). table 232 indicates the possible rules.

Table 232: Rule 2 for DST changeover

Value (hex)	Meaning	Note
00 ¹⁾	Changeover takes place on the defined weekday and month	The weekday is defined in: • Octet 3 for changeover to summer time • Octet 11 for changeover to winter time
01	Changeover takes place on the defined weekday after the defined date	Month and date are defined in: • Octet 5 to 6 for changeover to summer time • Octet 13 to 14 for changeover to winter time
02	Changeover takes place on the defined weekday before the defined date	

1) In this case without meaning:

- Octet 5 for changeover to summer time
- Octet 13 for changeover to winter time

Table 233: Coding of the weekday

Value (hexadecimal)	Meaning
00	Sunday
01	Monday
02	Tuesday
03	Wednesday
04	Thursday
05	Friday
06	Saturday

Table 234: Coding of the time difference (octet 9)

Value (hexadecimal)	Time difference of the change is
00	30 minutes
01	1 hour
02	90 minutes
03	2 hours
04	150 minutes
05	3 hours

12 Process Data for easy800/MFD (DPV1)

The PROFIBUS-DPV1 functionality of the EASY204-DP (from device version 07) allows you to access the following data ranges of an easy800/MFD via a class 1 or class 2 DPV1 master without any extensive programming required:

- Identification
- Mode
- Image
- Function blocks
- Date and time
- DST

The data from these ranges is located in Process Data Objects (data records) and can be read and/or written with the DPV1 “Read” and “Write” services.

Object overview

The following overview shows all the objects contained in the EASY204-DP that are available for a connected easy800/MFD.



The attributes API, Slot Number and Index form the address information for the DP master in order to access an object. Refer to the following section for further information about this.

Table 235: Process data objects in EASY204-DP for easy800/MFD

Object name	API	Slot Number	Index	Data length (octets)	Read/Write	→ page
Easy operating mode	0	0	100	1	R/W	247
Easy identification	0	1	100	50	R	245
Easy 700/800 inputs (I1 – I16)	0	1	250	2	R	286
Easy 700/800 analog inputs IA1 to IA4	0	1	197 – 200	2	R	288
Easy 700/800 outputs (Q1 – Q8)	0	1	251	1	R	290
Easy 700/800 pushbuttons	0	1	249	1	R	292
Easy 800 analog output (QA1)	0	1	252	2	R	350
Easy 800 local diagnostics (ID1 - ID16)	0	1	253	2	R	352
Easy 800 inputs network stations IW1 to IW8	0	1	211 – 218	2	R	354
Easy 800 Link inputs network stations RW1 to RW8	0	1	221 – 228	2	R	356
Easy 800 outputs network stations QW1 to QW8	0	1	231 – 238	1	R	358
Easy 800 Link outputs network stations SW1 to SW8	0	1	241 – 248	1	R	360
Easy 800 receive data network stations RNW1 to RNW8	0	0	201 – 208	4	R	362
Easy 800 send data network SNW1 to SNW8	0	1	201 – 208	4	R	364

Object name	API	Slot Number	Index	Data length (octets)	Read/Write	→ page
Easy 800 bit markers M1 to M96	0	0	1 – 96	1	R/W	366
Easy 800 byte markers MB1 to MB96	0	0	101 – 196	1	R/W	368
Easy 800 word markers MW1 to MW96	0	1	1 – 96	2	R/W	370
Easy 800 double word markers MD1 to MD96	0	1	101 – 196	4	R/W	372
Easy 800 8 byte data	0	1	229	8	R/W	374
Easy 800 16 byte data	0	1	230	16	R/W	376
Easy 800 32 byte data	0	1	239	32	R/W	378
Easy 800 64 byte data	0	1	240	64	R/W	381
Easy Link input data (R1 – R16)	0	0	98	2	R/W	259
Easy Link output data (S1 – S8)	0	0	99	1	R	261
Easy 700/800 write data function block	0	1	97	8	W	323
Easy 700/800 select function block data	0	1	99	4	W	319
Easy 700/800 read data function block	0	1	98	4	R	321
Easy clock	0	0	97	8	R/W	276
EASY 800 DST setting	0	0	209	20	R/W	431

Accessing objects

The Process Data Objects in the EASY204-DP are accessed via the DPV1 “Read” and “Write” services using the functions provided for this by the DP master system. Refer to the documentation of the manufacturer concerning this. Function blocks are normally used to carry out the access. In IEC 61131-3 based systems, the function blocks RDREC (Read) and WRREC (Write) defined by the PROFIBUS Users' Organisation in regulation 2.182 are normally made available. These also enable optimum access to complex data structures.

The following is required to access the objects:

- The address of the local DPV1 master interface
- The station address of the EASY204-DP to be accessed
- The application identifier (API) in the EASY204-DP (only required for class 2 DPV1 masters)
- The module to be addressed in EASY204-DP (Slot Number)
- The address (Index) of the required Process Data Object in the addressed EASY204-DP module
- The data length of the required Process Data Object
- A defined variable (memory range) in the local application that is to be assigned to the read data or containing the data to be written.

Refer to the topology of your master system for the address of the local DPV1 master interface. Refer to the PROFIBUS topology for the station address of the EASY204-DP.

The API (only required for class 2 DPV1 masters) must be stated for all Process Data Objects as 0.

- The slot number, the index, the data length and a sample variable definition (declaration) for an IEC 61131-3 system are shown in the following sections for each of the Process Data Objects provided in the EASY-204-DP for a connected easy800/MFD.

Read identification**"Easy identification" object**

The description of this object is identical to the easy600 (DPV1), → page 245.

Read/write mode**"Easy mode" object**

The description of this object is identical to the easy600 (DPV1), → page 247.

Read/write image**"Easy 700/800 inputs" object**

The description of this object is identical to the easy700 (DPV1), → page 286.

"Easy 700/800 analog inputs IA1 to IA4" objects

The description of this object is identical to the easy700 (DPV1), → page 288.

"Easy 700/800 outputs" objects

The description of this object is identical to the easy700 (DPV1), → page 290.

"Easy 700/800 pushbuttons" object

The description of this object is identical to the easy700 (DPV1), → page 292.

"Easy 800 analog output" object

This object contains the current status of the analog output of an easy800/MFD.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the analog output (QA1) of an easy800/MFD.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 252.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_Analog_Output	Word or UINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 236: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_Analog_Output	Octet 1 and 2	Analog value of QA1



When accessing the data of the "analog value of QA1" observe the Motorola coding format (octet N: High byte, octet N+1: Low byte) used in PROFIBUS-DP. If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

"Easy 800 local diagnostics" object

This object contains the current diagnostics of the NET connections of an easy800/MFD.

This object can only be read (Read). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to query the current status of the NET stations of an easy800/MFD and the status of the connection to the remote station (only MFD).

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 253.

Length of the object

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Diag	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 237: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_800_NET_Diag[1]	Octet 1	Bit 0	Status ID1 (Status Net Station 1)
Easy_800_NET_Diag[2]		Bit 1	Status ID2 (Status Net Station 2)
Easy_800_NET_Diag[3]		Bit 2	Status ID3 (Status Net Station 3)
Easy_800_NET_Diag[4]		Bit 3	Status ID4 (Status Net Station 4)
Easy_800_NET_Diag[5]		Bit 4	Status ID5 (Status Net Station 5)
Easy_800_NET_Diag[6]		Bit 5	Status ID6 (Status Net Station 6)
Easy_800_NET_Diag[7]		Bit 6	Status ID7 (Status Net Station 7)
Easy_800_NET_Diag[8]		Bit 7	Status ID8 (Status Net Station 8)
Easy_800_NET_Diag[9]	Octet 2	Bit 0	Status ID9 (Remote Connection MFD)
Easy_800_NET_Diag[10]		Bit 1	Not used
Easy_800_NET_Diag[11]		Bit 2	Not used
Easy_800_NET_Diag[12]		Bit 3	Not used
Easy_800_NET_Diag[13]		Bit 4	Not used
Easy_800_NET_Diag[14]		Bit 5	Not used
Easy_800_NET_Diag[15]		Bit 6	Not used
Easy_800_NET_Diag[16]		Bit 7	Not used



A set bit corresponds to the status "Station present". A unset bit corresponds to the status "Station not present".

"Easy 800 inputs network stations IW1 to IW8" objects

These objects contain the current status of the inputs of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the inputs (I1 to I16) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 211 to 218 for the EASYNET station 1 to 8

Length of objects

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Inputs	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 238: Address location and meaning of the data

Variable access (example)	Octet 1	Octet 2	Meaning
Easy_800_NET_Inputs[1]	Bit 0		Status I1
Easy_800_NET_Inputs[2]	Bit 1		Status I2
Easy_800_NET_Inputs[3]	Bit 2		Status I3
Easy_800_NET_Inputs[4]	Bit 3		Status I4
Easy_800_NET_Inputs[5]	Bit 4		Status I5
Easy_800_NET_Inputs[6]	Bit 5		Status I6
Easy_800_NET_Inputs[7]	Bit 6		Status I7
Easy_800_NET_Inputs[8]	Bit 7		Status I8
Easy_800_NET_Inputs[9]	Octet 2	Bit 0	Status I9
Easy_800_NET_Inputs[10]		Bit 1	Status I10
Easy_800_NET_Inputs[11]		Bit 2	Status I11
Easy_800_NET_Inputs[12]		Bit 3	Status I12
Easy_800_NET_Inputs[13]		Bit 4	Status I13
Easy_800_NET_Inputs[14]		Bit 5	Status I14
Easy_800_NET_Inputs[15]		Bit 6	Status I15
Easy_800_NET_Inputs[16]		Bit 7	Status I16



A set bit corresponds to the status "Station present". A unset bit corresponds to the status "Station not present".

"Easy 800 Link inputs network stations RW1 to RW8" objects

These objects contain the current status of the Link inputs of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the Link inputs (R1 to R16) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 221 to 228 for the EASYNET station 1 to 8

Length of objects

The data to be read is 2 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Link_Inputs	ARRAY [1..16] OF BOOL or ARRAY [1..2] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 239: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_800_NET_Link_Inputs[1]	Octet 1	Bit 0	Status R1
Easy_800_NET_Link_Inputs[2]		Bit 1	Status R2
Easy_800_NET_Link_Inputs[3]		Bit 2	Status R3
Easy_800_NET_Link_Inputs[4]		Bit 3	Status R4
Easy_800_NET_Link_Inputs[5]		Bit 4	Status R5
Easy_800_NET_Link_Inputs[6]		Bit 5	Status R6
Easy_800_NET_Link_Inputs[7]		Bit 6	Status R7
Easy_800_NET_Link_Inputs[8]		Bit 7	Status R8
Easy_800_NET_Link_Inputs[9]		Octet 2	Bit 0
Easy_800_NET_Link_Inputs[10]		Bit 1	Status R9
Easy_800_NET_Link_Inputs[11]		Bit 2	Status R10
Easy_800_NET_Link_Inputs[12]		Bit 3	Status R11
Easy_800_NET_Link_Inputs[13]		Bit 4	Status R12
Easy_800_NET_Link_Inputs[14]		Bit 5	Status R13
Easy_800_NET_Link_Inputs[15]		Bit 6	Status R14
Easy_800_NET_Link_Inputs[16]		Bit 7	Status R15



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 800 outputs network station SW1 to SW8" objects

These objects contain the current status of the outputs of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the outputs (Q1 to Q8) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 231 to 238 for the EASYNET station 1 to 8

Length of objects

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Outputs	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 240: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_NET_Outputs[1]	Octet 1	Bit 0
Easy_800_NET_Outputs[2]		Bit 1
Easy_800_NET_Outputs[3]		Bit 2
Easy_800_NET_Outputs[4]		Bit 3
Easy_800_NET_Outputs[5]		Bit 4
Easy_800_NET_Outputs[6]		Bit 5
Easy_800_NET_Outputs[7]		Bit 6
Easy_800_NET_Outputs[8]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 800 Link outputs network station SW1 to SW8" objects

These objects contain the current status of the Link outputs of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the Link outputs (S1 to S8) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 241 to 248 for the EASYNET station 1 to 8

Length of objects

The data to be read is 1 octet in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Link_Outputs	ARRAY [1..8] OF BOOL or BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 241: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_NET_Link_Outputs[1]	Octet 1	Bit 0
Easy_800_NET_Link_Outputs[2]		Bit 1
Easy_800_NET_Link_Outputs[3]		Bit 2
Easy_800_NET_Link_Outputs[4]		Bit 3
Easy_800_NET_Link_Outputs[5]		Bit 4
Easy_800_NET_Link_Outputs[6]		Bit 5
Easy_800_NET_Link_Outputs[7]		Bit 6
Easy_800_NET_Link_Outputs[8]		Bit 7



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 800 receive data network stations RNW1 to RNW8" objects

These objects contain the current status of the receive data of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the receive data (RN1 to RN32) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 201 to 208 for the EASYNET station 1 to 8

Length of objects

The data to be read is 4 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Receive_Data	ARRAY [1..32] OF BOOL or ARRAY [1..4] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 242: Address location and meaning of the data

Variable access (example)	Data position		Meaning
Easy_800_NET_Receive_Data[1]	Octet 1	Bit 0	Status RN1
Easy_800_NET_Receive_Data[2]		Bit 1	Status RN2
...	
Easy_800_NET_Receive_Data[7]		Bit 6	Status RN7
Easy_800_NET_Receive_Data[8]		Bit 7	Status RN8
Easy_800_NET_Receive_Data[9]		Octet 2	Bit 0
Easy_800_NET_Receive_Data[10]		Bit 1	Status RN9
...	
Easy_800_NET_Receive_Data[15]		Bit 6	Status RN15
Easy_800_NET_Receive_Data[16]		Bit 7	Status RN16
Easy_800_NET_Receive_Data[17]	Octet 3	Bit 0	Status RN17
Easy_800_NET_Receive_Data[18]		Bit 1	Status RN18
...	
Easy_800_NET_Receive_Data[23]		Bit 6	Status RN23
Easy_800_NET_Receive_Data[24]		Bit 7	Status RN24
Easy_800_NET_Receive_Data[25]		Octet 4	Bit 0
Easy_800_NET_Receive_Data[26]		Bit 1	Status RN25
...	Octet 4
Easy_800_NET_Receive_Data[31]		Bit 6	Status RN26
Easy_800_NET_Receive_Data[32]		Bit 7	Status RN27



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 800 send data network SNW1 to SNW8" objects

These objects contain the current status of the send data of an EASYNET station.

These objects can only be read (Read). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to query the current status of the send data (SN1 to SN32) of an EASYNET station.

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 201 to 208 for the EASYNET station 1 to 8

Length of objects

The data to be read is 4 octets in length. Do not enter a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_NET_Senddata	ARRAY [1..32] OF BOOL or ARRAY [1..4] OF BYTE

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 243: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_NET_Senddata[1]	Octet 1	Bit 0 Status SN1
Easy_800_NET_Senddata[2]		Bit 1 Status SN2
...		...
Easy_800_NET_Senddata[7]		Bit 6 Status SN7
Easy_800_NET_Senddata[8]		Bit 7 Status SN8
Easy_800_NET_Senddata[9]		Octet 2 Bit 0 Status SN9
Easy_800_NET_Senddata[10]		Bit 1 Status SN10
...		...
Easy_800_NET_Senddata[15]	Octet 3	Bit 6 Status SN15
Easy_800_NET_Senddata[16]		Bit 7 Status SN16
Easy_800_NET_Senddata[17]		Bit 0 Status SN17
Easy_800_NET_Senddata[18]		Bit 1 Status SN18
...		...
Easy_800_NET_Senddata[23]		Bit 6 Status SN23
Easy_800_NET_Senddata[24]		Bit 7 Status SN24
Easy_800_NET_Senddata[25]		Octet 4 Bit 0 Status SN25
Easy_800_NET_Senddata[26]	Octet 4	Bit 1 Status SN26
...		...
Easy_800_NET_Senddata[31]		Bit 6 Status SN31
Easy_800_NET_Senddata[32]		Bit 7 Status SN32



A set bit corresponds to the On state. An unset bit corresponds to the Off state.

"Easy 800 bit markers M1 to M96" objects

These objects contain the current status of the bit markers of an easy800/MFD.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current status of the bit markers M1 to M96 of the easy800 (Read)
- or to transfer a new setpoint for the bit markers M1 to M96 (Write).

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 1 to 96 for the bit markers M1 to M96.

Length of objects

The length of the data to be read or written is 1 octet. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_Bit_Marker	BYTE or USINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 244: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_Bit_Marker	Octet 1	Value of bit marker: 00 _{hex} = FALSE FF _{hex} = TRUE

"Easy 800 byte markers MB1 to MB96" objects

These objects contain the current status of the byte markers of an easy800/MFD.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current status of the byte markers MB1 to MB96 of the easy800 (Read)
- or to transfer a new setpoint for the byte markers MB1 to MB96 (Write).

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 101 to 196 for the byte markers MB1 to MB96.

Length of objects

The length of the data to be read or written is 1 octet. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_Byte_Marker	BYTE or USINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 245: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_Byte__Marker	Octet 1	Value of byte marker

"Easy 800 word markers MW1 to MW96" objects

These objects contain the current status of the word markers of an easy800/MFD.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current status of the word markers MW1 to MW96 of the easy800 (Read)
- or to transfer a new setpoint for the word markers MW1 to MW96 (Write).

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 1 to 96 for the word markers MW1 to MW96.

Length of objects

The length of the data to be read or written is 2 octets. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_Word_Marker	WORD or UINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 246: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_Word_Marker	Octet 1 and 2	Value of word marker



When accessing the data content of “value word markers” remember the Motorola coding format used in PROFIBUS-DP (octet N: high byte, octet N+1: low byte). If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

"Easy 800 double word markers MD1 to MD96" objects

These objects contain the current status of the double word markers of an easy800/MFD.

These objects can be read (Read) and written (Write). They can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use these objects to

- query the current status of the double word markers MD1 to MD96 of the easy800 (Read)
- or to transfer a new setpoint for the double word markers MD1 to MD96 (Write).

Addressing the objects

Use the following information to address the objects:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 101 to 196 for the double word markers MD1 to MD96.

Length of objects

The length of the data to be read or written is 4 octets. Do not enter any different length when calling the "Write" service, or a smaller length when calling the "Read" service, otherwise this will generate an error message.

Variable definition (example) of the objects

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_Double_Word_Marker	DWORD or UDINT

Data contents of the objects

The following table shows the address location and meaning of the data contained in the objects. It also illustrates in an example how to access the data using the defined variable.

Table 247: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_Double_Word_Marker	Octet 1 – 4	Value of double word marker



When accessing the data content of “value double word markers” remember the Motorola coding format used in PROFIBUS-DP (octet N: high byte, octet N+1: low byte). If the data processing format in your DP master system is different to this, and the DP access commands are not converted automatically, the necessary conversion must be implemented in the user program. Refer to the documentation of your DP master system concerning this.

"Easy 800 8 byte data" object

This object is used to exchange up to 8 bytes of application-related data between a PROFIBUS-DPV1 master and an easy800/MFD.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read up to 8 bytes of application-related data (content of double word markers MD67 and MD68) from the easy800 (Read)
- transfer up to 8 bytes of application-related data (content of double word markers MD67 and MD68) to the easy800 (Write)

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 229.

Length of the object

The maximum length of the data to be read or written is 8 octets. You can also specify a smaller length (1 to 7 octets) when calling the "Read" or "Write" services if you do not wish to use the entire data range.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_8_Byte_Data	Application-related, e.g. ARRAY [1..8] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 248: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_8_Byte_Data[1]	Octet 1	Value MD 67 Byte 1
Easy_800_8_Byte_Data[2]	Octet 2	Value MD 67 Byte 2
Easy_800_8_Byte_Data[3]	Octet 3	Value MD 67 Byte 3
Easy_800_8_Byte_Data[4]	Octet 4	Value MD 67 Byte 4
Easy_800_8_Byte_Data[5]	Octet 5	Value MD 68 Byte 1
Easy_800_8_Byte_Data[6]	Octet 6	Value MD 68 Byte 2
Easy_800_8_Byte_Data[7]	Octet 7	Value MD 68 Byte 3
Easy_800_8_Byte_Data[8]	Octet 8	Value MD 68 Byte 4

"Easy 800 16 byte data" object

This object is used to exchange up to 16 bytes of application-related data between a PROFIBUS-DPV1 master and an easy800/MFD.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read up to 16 bytes of application-related data (content of double word markers MD69 to MD72) from the easy800 (Read)
- or transfer up to 16 bytes of application-related data (content of double word markers MD67 and MD68) to the easy800 (Write)

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 230.

Length of the object

The maximum length of the data to be read or written is 16 octets. You can also specify a smaller length (1 to 15 octets) when calling the "Read" or "Write" services if you do not wish to use the entire data range.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_16_Byte_Data	Application-related, e.g. ARRAY [1..16] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 249: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_16_Byte_Data[1]	Octet 1	Value MD 69 Byte 1
Easy_800_16_Byte_Data[2]	Octet 2	Value MD 69 Byte 2
Easy_800_16_Byte_Data[3]	Octet 3	Value MD 69 Byte 3
Easy_800_16_Byte_Data[4]	Octet 4	Value MD 69 Byte 4
Easy_800_16_Byte_Data[5]	Octet 5	Value MD 70 Byte 1
Easy_800_16_Byte_Data[6]	Octet 6	Value MD 70 Byte 2
Easy_800_16_Byte_Data[7]	Octet 7	Value MD 70 Byte 3
Easy_800_16_Byte_Data[8]	Octet 8	Value MD 70 Byte 4
Easy_800_16_Byte_Data[9]	Octet 9	Value MD 71 Byte 1
Easy_800_16_Byte_Data[10]	Octet 10	Value MD 71 Byte 2
Easy_800_16_Byte_Data[11]	Octet 11	Value MD 71 Byte 3
Easy_800_16_Byte_Data[12]	Octet 12	Value MD 71 Byte 4
Easy_800_16_Byte_Data[13]	Octet 13	Value MD 72 Byte 1
Easy_800_16_Byte_Data[14]	Octet 14	Value MD 72 Byte 2
Easy_800_16_Byte_Data[15]	Octet 15	Value MD 72 Byte 3
Easy_800_16_Byte_Data[16]	Octet 16	Value MD 72 Byte 4

"Easy 800 32 byte data" object

This object is used to exchange up to 32 bytes of application-related data between a PROFIBUS-DPV1 master and an easy800/MFD.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read up to 32 bytes of application-related data (content of double word markers MD73 to MD80) from the easy800 (Read)
- or transfer up to 32 bytes of application-related data (content of double word markers MD73 and MD80) to the easy800 (Write)

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 239.

Length of the object

The maximum length of the data to be read or written is 32 octets. You can also specify a smaller length (1 to 31 octets) when calling the "Read" or "Write" services if you do not wish to use the entire data range.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_32_Byte_Data	Application-related, e.g. ARRAY [1..32] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 250: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_32_Byte_Data[1]	Octet 1	Value MD 73 Byte 1
Easy_800_32_Byte_Data[2]	Octet 2	Value MD 73 Byte 2
Easy_800_32_Byte_Data[3]	Octet 3	Value MD 73 Byte 3
Easy_800_32_Byte_Data[4]	Octet 4	Value MD 73 Byte 4
Easy_800_32_Byte_Data[5]	Octet 5	Value MD 74 Byte 1
Easy_800_32_Byte_Data[6]	Octet 6	Value MD 74 Byte 2
Easy_800_32_Byte_Data[7]	Octet 7	Value MD 74 Byte 3
Easy_800_32_Byte_Data[8]	Octet 8	Value MD 74 Byte 4
Easy_800_32_Byte_Data[9]	Octet 9	Value MD 75 Byte 1
Easy_800_32_Byte_Data[10]	Octet 10	Value MD 75 Byte 2
Easy_800_32_Byte_Data[11]	Octet 11	Value MD 75 Byte 3
Easy_800_32_Byte_Data[12]	Octet 12	Value MD 75 Byte 4
Easy_800_32_Byte_Data[13]	Octet 13	Value MD 76 Byte 1
Easy_800_32_Byte_Data[14]	Octet 14	Value MD 76 Byte 2
Easy_800_32_Byte_Data[15]	Octet 15	Value MD 76 Byte 3
Easy_800_32_Byte_Data[16]	Octet 16	Value MD 76 Byte 4
Easy_800_32_Byte_Data[17]	Octet 17	Value MD 77 Byte 1
Easy_800_32_Byte_Data[18]	Octet 18	Value MD 77 Byte 2
Easy_800_32_Byte_Data[19]	Octet 19	Value MD 77 Byte 3
Easy_800_32_Byte_Data[20]	Octet 20	Value MD 77 Byte 4
Easy_800_32_Byte_Data[21]	Octet 21	Value MD 78 Byte 1
Easy_800_32_Byte_Data[22]	Octet 22	Value MD 78 Byte 2
Easy_800_32_Byte_Data[23]	Octet 23	Value MD 78 Byte 3

Variable access (example)	Data position	Meaning
Easy_800_32_Byte_Data[24]	Octet 24	Value MD 78 Byte 4
Easy_800_32_Byte_Data[25]	Octet 25	Value MD 79 Byte 1
Easy_800_32_Byte_Data[26]	Octet 26	Value MD 79 Byte 2
Easy_800_32_Byte_Data[27]	Octet 27	Value MD 79 Byte 3
Easy_800_32_Byte_Data[28]	Octet 28	Value MD 79 Byte 4
Easy_800_32_Byte_Data[29]	Octet 29	Value MD 80 Byte 1
Easy_800_32_Byte_Data[30]	Octet 30	Value MD 80 Byte 2
Easy_800_32_Byte_Data[31]	Octet 31	Value MD 80 Byte 3
Easy_800_32_Byte_Data[32]	Octet 32	Value MD 80 Byte 4

"Easy 800 64 byte data" object

This object is used to exchange up to 64 bytes of application-related data between a PROFIBUS-DPV1 master and an easy800/MFD.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read up to 64 bytes of application-related data (content of double word markers MD81 to MD97) from the easy800 (Read)
- or transfer up to 64 bytes of application-related data (content of double word markers MD81 and MD96) to the easy800 (Write)

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 1
- Index is 240.

Length of the object

The maximum length of the data to be read or written is 64 octets. You can also specify a smaller length (1 to 63 octets) when calling the "Read" or "Write" services if you do not wish to use the entire data range.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system:

Name	Data type
Easy_800_64_Byte_Data	Application-related, e.g. ARRAY [1..64] OF BYTE

Data content of the object

The following table shows the address location and meaning of the data contained in the object. It also illustrates in an example how to access the data using the defined variable.

Table 251: Address location and meaning of the data

Variable access (example)	Data position	Meaning
Easy_800_64[Byte>Data[1]	Octet 1	Value MD 81 Byte 1
Easy_800_64[Byte>Data[2]	Octet 2	Value MD 81 Byte 2
Easy_800_64[Byte>Data[3]	Octet 3	Value MD 81 Byte 3
Easy_800_64[Byte>Data[4]	Octet 4	Value MD 81 Byte 4
Easy_800_64[Byte>Data[5]	Octet 5	Value MD 82 Byte 1
Easy_800_64[Byte>Data[6]	Octet 6	Value MD 82 Byte 2
Easy_800_64[Byte>Data[7]	Octet 7	Value MD 82 Byte 3
Easy_800_64[Byte>Data[8]	Octet 8	Value MD 82 Byte 4
Easy_800_64[Byte>Data[9]	Octet 9	Value MD 83 Byte 1
Easy_800_64[Byte>Data[10]	Octet 10	Value MD 83 Byte 2
Easy_800_64[Byte>Data[11]	Octet 11	Value MD 83 Byte 3
Easy_800_64[Byte>Data[12]	Octet 12	Value MD 83 Byte 4
Easy_800_64[Byte>Data[13]	Octet 13	Value MD 84 Byte 1
Easy_800_64[Byte>Data[14]	Octet 14	Value MD 84 Byte 2
Easy_800_64[Byte>Data[15]	Octet 15	Value MD 84 Byte 3
Easy_800_64[Byte>Data[16]	Octet 16	Value MD 84 Byte 4
Easy_800_64[Byte>Data[17]	Octet 17	Value MD 85 Byte 1
Easy_800_64[Byte>Data[18]	Octet 18	Value MD 85 Byte 2
Easy_800_64[Byte>Data[19]	Octet 19	Value MD 85 Byte 3
Easy_800_64[Byte>Data[20]	Octet 20	Value MD 85 Byte 4
Easy_800_64[Byte>Data[21]	Octet 21	Value MD 86 Byte 1
Easy_800_64[Byte>Data[22]	Octet 22	Value MD 86 Byte 2
Easy_800_64[Byte>Data[23]	Octet 23	Value MD 86 Byte 3

Variable access (example)	Data position	Meaning
Easy_800_64[Byte>Data[24]	Octet 24	Value MD 86 Byte 4
Easy_800_64[Byte>Data[25]	Octet 25	Value MD 87 Byte 1
Easy_800_64[Byte>Data[26]	Octet 26	Value MD 87 Byte 2
Easy_800_64[Byte>Data[27]	Octet 27	Value MD 87 Byte 3
Easy_800_64[Byte>Data[28]	Octet 28	Value MD 87 Byte 4
Easy_800_64[Byte>Data[29]	Octet 29	Value MD 88 Byte 1
Easy_800_64[Byte>Data[30]	Octet 30	Value MD 88 Byte 2
Easy_800_64[Byte>Data[31]	Octet 31	Value MD 88 Byte 3
Easy_800_64[Byte>Data[32]	Octet 32	Value MD 88 Byte 4
Easy_800_64[Byte>Data[33]	Octet 33	Value MD 89 Byte 1
Easy_800_64[Byte>Data[34]	Octet 34	Value MD 89 Byte 2
Easy_800_64[Byte>Data[35]	Octet 35	Value MD 89 Byte 3
Easy_800_64[Byte>Data[36]	Octet 36	Value MD 89 Byte 4
Easy_800_64[Byte>Data[37]	Octet 37	Value MD 90 Byte 1
Easy_800_64[Byte>Data[38]	Octet 38	Value MD 90 Byte 2
Easy_800_64[Byte>Data[39]	Octet 39	Value MD 90 Byte 3
Easy_800_64[Byte>Data[40]	Octet 40	Value MD 90 Byte 4
Easy_800_64[Byte>Data[41]	Octet 41	Value MD 91 Byte 1
Easy_800_64[Byte>Data[42]	Octet 42	Value MD 91 Byte 2
Easy_800_64[Byte>Data[43]	Octet 43	Value MD 91 Byte 3
Easy_800_64[Byte>Data[44]	Octet 44	Value MD 91 Byte 4
Easy_800_64[Byte>Data[45]	Octet 45	Value MD 92 Byte 1
Easy_800_64[Byte>Data[46]	Octet 46	Value MD 92 Byte 2
Easy_800_64[Byte>Data[47]	Octet 47	Value MD 92 Byte 3
Easy_800_64[Byte>Data[48]	Octet 48	Value MD 92 Byte 4
Easy_800_64[Byte>Data[49]	Octet 49	Value MD 93 Byte 1
Easy_800_64[Byte>Data[50]	Octet 50	Value MD 93 Byte 2
Easy_800_64[Byte>Data[51]	Octet 51	Value MD 93 Byte 3

Variable access (example)	Data position	Meaning
Easy_800_64_Byte_Data[52]	Octet 52	Value MD 93 Byte 4
Easy_800_64_Byte_Data[53]	Octet 53	Value MD 94 Byte 1
Easy_800_64_Byte_Data[54]	Octet 54	Value MD 94 Byte 2
Easy_800_64_Byte_Data[55]	Octet 55	Value MD 94 Byte 3
Easy_800_64_Byte_Data[56]	Octet 56	Value MD 94 Byte 4
Easy_800_64_Byte_Data[57]	Octet 57	Value MD 95 Byte 1
Easy_800_64_Byte_Data[58]	Octet 58	Value MD 95 Byte 2
Easy_800_64_Byte_Data[59]	Octet 59	Value MD 95 Byte 3
Easy_800_64_Byte_Data[60]	Octet 60	Value MD 95 Byte 4
Easy_800_64_Byte_Data[61]	Octet 61	Value MD 96 Byte 1
Easy_800_64_Byte_Data[62]	Octet 62	Value MD 96 Byte 2
Easy_800_64_Byte_Data[63]	Octet 63	Value MD 96 Byte 3
Easy_800_64_Byte_Data[64]	Octet 64	Value MD 96 Byte 4

"Easy Link input data" object

The description of this object is identical to the easy600 (DPV1), → page 259.

"Easy Link output data" object

The description of this object is identical to the easy600 (DPV1), → page 261.

**Read/write function block
data Procedure**

The following generic objects simplify the reading and writing of function block data:

- “Easy 700/800 select data function block”
- “Easy 700/800 read data function block”
- “Easy 700/800 write data function block”.

Two object accesses are required in order to read the data of a function block:

- Selection of the required function block data for the subsequent read operation:
→ “Easy 700/800 function block data selection” object, page 319.
- Reading of the selected function block data: → “Easy 700/800 read function block data” object, page 321.

Only one object access is required in order to write the data of a function block. The selection of the required function block data and the transfer of the new values for this are executed simultaneously.

→ “Easy 700/800 write function block data” object, page 323.

Analog value comparators A1 to A32

Operand selection	Value (hex)
Type	11
Instance ¹⁾	01 – 20
Index	00 – 07, → table 252

- 1) The value 01_{hex} selects the analog value comparator A1, the value 20_{hex} the analog value comparator A32.

Table 252: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit I0	–	–	→ Tab. 253	–	R
01	Mode	→ Tab. 254	–	–	–	R
02	Comparison value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Gain factor for F1 (I1 = F1 * Value)	DWORD or UDINT ¹⁾				R/W ²⁾
04	Comparison value I2	DWORD or UDINT ¹⁾				R/W ²⁾
05	Gain factor for I2 (I2 = F2 * Value)	DWORD or UDINT ¹⁾				R/W ²⁾
06	Offset OS for I1	DWORD or UDINT ¹⁾				R/W ²⁾
07	Switching hysteresis HY for value I2	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 253: 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 254: Index 1 - Mode

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Arithmetic function blocks AR1 to AR32

Operand selection	Value (hex)
Type	12
Instance ¹⁾	01 – 20
Index	00 – 04, → table 255

- 1) The value 01_{hex} for the instance selects the arithmetic function block AR1, the value 20_{hex} the arithmetic function block AR32.

Table 255: Operand overview

Index	Operand	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	–	–	→ Tab. 256	–	R
01	Mode	→ Tab. 257	–	–	–	R
02	First operand I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Second operand I2	DWORD or UDINT ¹⁾				R/W ²⁾
04	Result QV	DWORD or UDINT 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 256: Index 0: Bit IO

	Bit	7	6	5	4	3	2	1
FB output Data 3		-	-	-	-	-	ZE ¹	CY1

- 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 257: Index 1 - Mode

Data 1 (hex)		
00	ADD	Add ($I1 + I2 = QV$)
01	SUB	Subtract ($I1 - I2 = QV$)
02	MUL	Multiply ($I1 \times I2 = QV$)
03	DIV	Divide ($I1 : I2 = QV$)



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Block Compare function blocks BC1 to BC32

Operand selection	Value (hex)
Type	25
Instance ¹⁾	01 – 20
Index	00 – 04, → table 258

- 1) The value 01_{hex} for the instance selects the Block compare function block BC1, the value 20_{hex} the Block compare function block BC32.

Table 258: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	→ Tab. 259	–	→ Tab. 259	–	R
01	Mode	→ Tab. 260	–	–	–	R
02	Source range I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Destination range I2	DWORD or UDINT ¹⁾				R/W ²⁾
04	Number of elements to compare NO (max. 192 bytes)	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 259: 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	E15

- 1) Activates the function block on status 1.
- 2) Status 1 if the data ranges are equal; status 0 if not equal
- 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 260: Index 1 - Mode

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Block Transfer function blocks BT1 to BT32

Operand selection	Value (hex)
Type	26
Instance ¹⁾	01 – 20
Index	00 – 04, → table 261

- 1) The value 01_{hex} for the instance selects the Block transfer function block BT1, the value 20_{hex} the Block transfer function block BT32.

Table 261: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	→ Tab. 262	–	→ Tab. 262	–	R
01	Mode	→ Tab. 263	–	–	–	R
02	Source range I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Destination range I2	DWORD or UDINT ¹⁾				R/W ²⁾
04	Number of elements to be transferred NO (max. 192 bytes)	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 262: Index 0 – Bit IO

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

- 1) Transfer of the source address specified at I1 to the target address specified at I2 on rising edge.
- 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 263: Index 1 - Mode

Data 1 (hex)	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.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Boolean operation function blocks BV1 to BV32

Operand selection	Value (hex)
Type	13
Instance ¹⁾	01 – 20
Index	00 – 04, → table 264

- 1) The value 01_{hex} for the instance selects the Boolean operation function block BV1, the value 20_{hex} the Boolean operation function block BV32.

Table 264: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	–	–	→ Tab. 265	–	R
01	Mode	→ Tab. 266	–	–	–	R
02	First operand I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Second operand I2	DWORD or UDINT ¹⁾				R/W ²⁾
04	Result QV	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 265: 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 266: 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.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Counters C1 – C32

Operand selection	Value (hex)
Type	14
Instance ¹⁾	01 – 20
Index	00 – 05, → table 267

- 1) The value 01_{hex} for the instance selects the counter function block C1, the value 20_{hex} the counter function block C32.

Table 267: Meaning of operands

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	→ Tab. 268	–	→ Tab. 268	–	R
02	Upper setpoint SH	DWORD or UDINT ¹⁾				R/W ²⁾
03	Lower setpoint SL	DWORD or UDINT ¹⁾				R/W ²⁾
04	Preset actual value SV	DWORD or UDINT ¹⁾				R/W ²⁾
05	Current actual value QV	DWORD or UDINT 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 268: Index 0: Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Frequency counters CF1 to CF4

Operand selection	Value (hex)
Type	15
Instance ¹⁾	01 – 04
Index	00 – 04, → table 269

- 1) The value 01_{hex} for the instance selects the frequency counter function block CF1, the value 04_{hex} the frequency counter function block CF4.

Table 269: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 270	–	→ Tab. 270	–	R
02	Upper setpoint SH	DWORD or UDINT ¹⁾				R/W ²⁾
03	Lower setpoint SL	DWORD or UDINT ¹⁾				R/W ²⁾
04	Current actual value QV	DWORD or UDINT 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 270: Index 0: Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

High-speed counters CH1 to CH4

Operand selection	Value (hex)
Type	16
Instance ¹⁾	01 – 04
Index	00 – 05, → table 271

- 1) The value 01_{hex} for the instance selects the high-speed counter function block CH1, the value 04_{hex} the high-speed counter function block CH4.

Table 271: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	→ Tab. 272	–	→ Tab. 272	–	R
02	Upper setpoint SH	DWORD or UDINT ¹⁾				R/W ²⁾
03	Lower setpoint SL	DWORD or UDINT ¹⁾				R/W ²⁾
04	Preset actual value SV	DWORD or UDINT ¹⁾				R/W ²⁾
05	Current actual value QV	DWORD or UDINT 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 272: Index 0: Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Incremental counters CI1 to CI4

Operand selection	Value (hex)
Type	17
Instance ¹⁾	01 – 04
Index	00 – 05, → table 273

- 1) The value 01_{hex} for the instance selects the incremental counter function block CI1, the value 04_{hex} the incremental counter function block CI4.

Table 273: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 274	–	→ Tab. 274	–	R
02	Upper setpoint SH	DWORD or UDINT ¹⁾				R/W ²⁾
03	Lower setpoint SL	DWORD or UDINT ¹⁾				R/W ²⁾
04	Preset actual value SV	DWORD or UDINT ¹⁾				R/W ²⁾
05	Current actual value QV	DWORD or UDINT 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 274: Index 0 – Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Comparators CP1 to CP32

Operand selection	Value (hex)
Type	18
Instance ¹⁾	01 – 20
Index	00 – 03, → table 275

- 1) The value 01_{hex} for the instance selects the comparator function block CP1, the value 20_{hex} the comparator function block CP32.

Table 275: Operand overview

Index	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
00	Bit IO	–	–	→ Tab. 276	–	R
02	Comparison value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Comparison value I2	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 276: Index 0 – Bit IO

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Text output function blocks D1 to D32

Operand selection	Value (hex)
Type	19
Instance ¹⁾	01 – 20
Index	00 – 33

- 1) The value 01_{hex} for the instance selects the text output function block D1, the value 20_{hex} the text output function block D32.

Table 277: Operand overview

Index hex	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 278	–	→ Tab. 278	–	R
02	Text line 1, column 1 - 4	DWORD or UDINT ¹⁾				R
03	Text line 1, column 5 - 8	DWORD or UDINT ¹⁾				R
04	Text line 1, column 9 - 12	DWORD or UDINT ¹⁾				R
05	Text line 1, column 13 - 16	DWORD or UDINT ¹⁾				R
06	Text line 2, column 1 - 4	DWORD or UDINT ¹⁾				R
07	Text line 2, column 5 - 8	DWORD or UDINT ¹⁾				R
08	Text line 2, column 9 - 12	DWORD or UDINT ¹⁾				R
09	Text line 2, column 13 - 16	DWORD or UDINT ¹⁾				R
10	Text line 3, column 1 - 4	DWORD or UDINT ¹⁾				R
11	Text line 3, column 5 - 8	DWORD or UDINT ¹⁾				R
12	Text line 3, column 9 - 12	DWORD or UDINT ¹⁾				R
13	Text line 3, column 13 - 16	DWORD or UDINT ¹⁾				R
14	Text line 4, column 1 - 4	DWORD or UDINT ¹⁾				R
15	Text line 4, column 5 - 8	DWORD or UDINT ¹⁾				R
16	Text line 4, column 9 - 12	DWORD or UDINT ¹⁾				R
17	Text line 4, column 13 - 16	DWORD or UDINT ¹⁾				R
18	Variable 1	DWORD or UDINT ¹⁾				R/W ²⁾

Index hex	Operand	Data 1	Data 2	Data 3	Data 4	Read/Write
19	Variable 2	DWORD or UDINT ¹⁾				R/W ²⁾
20	Variable 3	DWORD or UDINT ¹⁾				R/W ²⁾
21	Variable 4	DWORD or UDINT ¹⁾				R/W ²⁾
22	Scaling minimum value 1	DWORD or UDINT ¹⁾				R
23	Scaling minimum value 2	DWORD or UDINT ¹⁾				R
24	Scaling minimum value 3	DWORD or UDINT ¹⁾				R
25	Scaling minimum value 4	DWORD or UDINT ¹⁾				R
26	Scaling maximum value 1	DWORD or UDINT ¹⁾				R
27	Scaling maximum value 2	DWORD or UDINT ¹⁾				R
28	Scaling maximum value 3	DWORD or UDINT ¹⁾				R
29	Scaling maximum value 4	DWORD or UDINT ¹⁾				R
30	Control information line 1	DWORD or UDINT ¹⁾				R
31	Control information line 2	DWORD or UDINT ¹⁾				R
32	Control information line 3	DWORD or UDINT ¹⁾				R
33	Control information line 4	DWORD or UDINT 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 278: Index 0 – Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Data function blocks DB1 to DB32

Operand selection	Value (hex)
Type	1A
Instance ¹⁾	01 – 20
Index	00 – 03, → table 279

- 1) The value 01_{hex} for the instance selects the data function block DB1, the value 20_{hex} the data function block DB32.

Table 279: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit I0	→ Tab. 280	–	→ Tab. 280	–	R
02	Input value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Output value QV	DWORD or UDINT 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 280: Index 0 – Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

PID controllers DC1 to DC32

Operand selection	Value (hex)
Type	27
Instance ¹⁾	01 – 20
Index	00 – 09, → table 281

- 1) The value 01_{hex} for the instance selects the PID controller DC1, the value 20_{hex} the PID controller DC32.

Table 281: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 282	–	→ Tab. 282	–	R
01	Mode	→ Tab. 283	–	–	–	R
02	Setpoint I1: -32768 to +32767	DWORD or UDINT ¹⁾				R/W ²⁾
03	Actual value I2: -32768 to +32767	DWORD or UDINT ¹⁾				R/W ²⁾
04	Proportional gain KP [%], Value range: 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
05	Reset time TN [0.1 s], Value range 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
06	Rate time TV [0.1 s], Value range 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
07	Scan time TC Value range 0.1s to 65535.5s Value 0: time is same as program cycle	DWORD or UDINT ¹⁾				R/W ²⁾
08	Manual manipulated variable MV, Value range: -4096 to +4095	DWORD or UDINT ¹⁾				R/W ²⁾
09	Manipulated variable QV Mode UNI, value range 0 to +4095 (12 bit) Mode BIP, value range -4096 to +4095 (13 bit)	DWORD or UDINT 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 282: 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 manipulated variable was exceeded

Table 283: Index 1 - Mode

Data 1	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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Signal smoothing filters FT1 to FT32

Operand selection	Value (hex)
Type	28
Instance ¹⁾	01 – 20
Index	00 – 05, → table 284

- 1) The value 01_{hex} for the instance selects the signal smoothing filter FT1, the value 20_{hex} the signal smoothing filter FT32.

Table 284: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/write
00	Bit IO	–	–	→ Tab. 285	–	R
02	Input value I1: -32768 to +32767	DWORD or UDINT ¹⁾				R/W ²⁾
03	Recovery time TG [0.1 s], Value range 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
04	Proportional gain KP [%], Value range: 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
05	Delayed output value QV, value range -32768 to +32767	DWORD or UDINT 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 285: Index 0 – Bit IO

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

- 1) Activates the function block on status 1.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Get network data function blocks GT1 to GT32

Operand selection	Value (hex)
Type	1B
Instance ¹⁾	01 – 20
Index	00 – 02, → table 286

- 1) The value 01_{hex} for the instance selects the Get network data function block GT1, the value 20_{hex} the Get network data function block GT32.

Table 286: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	–	–	→ Tab. 287	–	R
01	Mode/Parameter	→ Tab. 288	–	→ Tab. 288	–	R
02	Actual value from the network QV	DWORD or UDINT ¹⁾				R

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

Table 287: 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 288: 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
		PT01
		..
		PT32

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

7-day time switches HW1 to HW32

Operand selection	Value (hex)
Type	1C
Instance ¹⁾	01 – 20
Index	00 – 05, → table 289

- 1) The value 01_{hex} for the instance selects the 7-day time switch HW1, the value 20_{hex} the 7-day time switch HW32.

Table 289: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	–	–	→ table 290	–	R
02	Parameters Channel A	→ table 291				R
03	Parameters Channel B	→ table 291				R
04	Parameters Channel C	→ table 291				R
05	Parameters Channel D	→ table 291				R

Table 290: Index 0 – Bit IO

FB output Data 3	Bit	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	-	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 291: 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
Weekday								Hour								Minute							

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

m5 to m0: Minute (0 to 59)

h4 to h0: Hour (0 to 23)

d5 to d0: Weekday (0 = Sunday to 6 = Saturday)



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Year time switches HY1 – HY32

Operand selection	Value (hex)
Type	1D
Instance ¹⁾	01 – 20
Index	00 – 05, → table 292

1) The value 01_{hex} for the instance selects the year time switch HY1, the value 20_{hex} the year time switch HY32.

Table 292: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit I0	–	–	→ Tab. 293	–	R
02	Parameters Channel A	→ Tab. 294				R
03	Parameters Channel B	→ Tab. 294				R
04	Parameters Channel C	→ Tab. 294				R
05	Parameters Channel D	→ Tab. 294				R

Table 293: 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 294: 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)



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Value scaling function blocks LS1 to LS32

Operand selection	Value (hex)
Type	29
Instance ¹⁾	01 – 20
Index	00 – 07, → table 295

- 1) The value 01_{hex} for the instance selects the value scaling function block LS1, the value 20_{hex} the value scaling function block LS32.

Table 295: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	–	–	→ Tab. 296	–	R
02	Input value I1: Value range 32 bit	DWORD or UDINT ¹⁾				R/W ²⁾
03	Interpolation point X1: (X coordinate) value range: 32 bit	DWORD or UDINT ¹⁾				R/W ²⁾
04	Interpolation point Y1: (Y coordinate) value range: 32 bit	DWORD or UDINT ¹⁾				R/W ²⁾
05	Interpolation point X2: (X coordinate) value range: 32 bit	DWORD or UDINT ¹⁾				R/W ²⁾
06	Interpolation point Y2: (Y coordinate) value range: 32 bit	DWORD or UDINT ¹⁾				R/W ²⁾
07	Scaled input value QV	DWORD or UDINT 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 296: 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.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Master reset MR1 to MR32

Operand selection	Value (hex)
Type	0F
Instance ¹⁾	01 – 20
Index	00 – 01, → table 297

- 1) The value 01_{hex} for the instance selects the master reset function block MR1, the value 20_{hex} the master reset function block MR32.

Table 297: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/ Write
0	Bit IO	→ Tab. 298	–	→ Tab. 298	–	R
1	Mode/Parameter	→ Tab. 299	–	–	–	R

Table 298: 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 299: Index 1 - Mode

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Numerical converters NC1 to NC32

Operand selection	Value (hex)
Type	2A
Instance ¹⁾	01 – 20
Index	00 – 03, → table 300

- 1) The value 01_{hex} for the instance selects the numerical converter NC1, the value 20_{hex} the numerical converter NC32.

Table 300: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	–	–	→ Tab. 301	–	R
01	Mode	→ Tab. 302	–	–	–	R
02	Input value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Output value QV	DWORD or UDINT 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 301: 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.

Table 302: Index 1 - Mode

Data 1 (hex)	
00	BCD
01	BIN



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Operating hours counters OT1 – OT4

Operand selection	Value (hex)
Type	1E
Instance ¹⁾	01 – 04
Index	00 – 03, → table 303

- 1) The value 01_{hex} for the instance selects the operating hours counter OT1, the value 04_{hex} the operating hours counter OT4.

Table 303: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 304	–	→ Tab. 304	–	R
02	Upper threshold value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Actual value QV	DWORD or UDINT 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 304: Index 0 – Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Put network data function blocks PT1 to PT32

Operand selection	Value (hex)
Type	1F
Instance ¹⁾	01 – 20
Index	00 – 02
Data 1 – 4	depending on index, → table 305

- 1) The value 01_{hex} for the instance selects the Put network data function block PT1, the value 20_{hex} the Put network data function block PT32.

Table 305: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
0	Bit IO	→ Tab. 306	–	→ Tab. 306	–	R
2	Setpoint for the network QV	DWORD or UDINT ¹⁾				R

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

Table 306: Index 0 – Bit IO

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

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



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Pulse width modulation PW1 to PW2

Operand selection	Value (hex)
Type	2B
Instance ¹⁾	01 – 02
Index	00 – 04, → table 307

- 1) The value 01_{hex} for the instance selects the pulse width modulation function block PW1, the value 02_{hex} the pulse width modulation function block PW2.

Table 307: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 308	–	→ Tab. 308	–	R
02	Manipulated variable SV Value range 0 to 4095 (12 bit)	DWORD or UDINT ¹⁾				R/W ²⁾
03	Period duration PD [ms], Value range: 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾
04	Minimum on duration ME [ms], Value range: 0 to 65535	DWORD or UDINT ¹⁾				R/W ²⁾

- 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 308: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
FB input Data 1		–	–	–	–	–	–	–	EN1
FB output Data 3		–	–	–	–	–	–	–	E12

- 1) Activates the function block on status 1.
- 2) Status 1 if below the minimum on duration or minimum off duration



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Synchronize clock function block SC1

Operand selection	Value (hex)
Type	20
Instance	01
Index	00, → table 309

Table 309: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	→ Tab. 310	–	→ Tab. 310	–	R

Table 310: Index 0 – Bit IO

	Bit	7	6	5	4	3	2	1	0
		–	–	–	–	–	–	–	T1
FB input Data 1		–	–	–	–	–	–	–	Q1 ²
FB output Data 3		–	–	–	–	–	–	–	Q1 ²

- 1) Trigger coil. If the coil is triggered (rising edge), the current date, weekday and time of the sending station are automatically sent to the NET network.
- 2) Status 1 if the trigger coil SC01T_ is also 1.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Set cycle time function block ST1

Operand selection	Value (hex)
Type	2C
Instance	01
Index	00 – 02, → table 311

Table 311: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
0	Bit IO	–	–	→ Tab. 312	–	R
2	Cycle time [ms], Value range: 0 to 1000	DWORD or UDINT ¹⁾				R

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

Table 312: Index 0 – Bit IO

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

- 1) Activates the function block on status 1.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Timing relays T1 – T32

Operand selection	Value (hex)
Type	21
Instance ¹⁾	01 – 20
Index	00 – 04, → table 313

- 1) The value 01_{hex} for the instance selects the timing relay T1,
the value 20_{hex} the timing relay T32.

Table 313: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
0	Bit IO	→ Tab. 314	–	→ Tab. 314	–	R
01	Mode/Parameter	→ Tab. 315	–	→ Tab. 315	–	R
02	Time setpoint value 1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Time setpoint 2	DWORD or UDINT ¹⁾				R/W ²⁾
04	Elapsed time	DWORD or UDINT 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 314: 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) Contact type

Table 315: Index 1 - Mode/Parameters

Mode	Data 1	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 generator
	7	Flashing relay (two time setpoints)
	8	Off-delayed, retriggerable (easy600 mode)
	9	Off-delayed with random setpoint, retriggerable (easy600 mode)
Parameter	Data 3	Mode
	0	S (milliseconds)
	1	M:S (seconds)
	2	H:M (minutes)



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Value limitation VC1 to VC32

Operand selection	Value (hex)
Type	2D
Instance ¹⁾	01 – 20
Index	00 – 05, → table 316

- 1) The value 01_{hex} for the instance selects the value limitation function block VC1, the value 20_{hex} the value limitation function block VC32.

Table 316: Operand overview

Index hex	Data	Data 1	Data 2	Data 3	Data 4	Read/Write
00	Bit IO	–	–	→ Tab. 317	–	R
02	Input value I1	DWORD or UDINT ¹⁾				R/W ²⁾
03	Upper limit value SH	DWORD or UDINT ¹⁾				R/W ²⁾
04	Lower limit value SL	DWORD or UDINT ¹⁾				R/W ²⁾
05	Output value QV	DWORD or UDINT 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 317: Index 0 – Bit IO

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

- 1) Activates the function block on status 1.



Further information on this function block is provided in the easy800 manual (AWB2528-1423G) or in the EASY-SOFT Help.

Read/write date and time “Easy clock” object

The description of this object is identical to the easy600 (DPV1), → page 276.

Read/write DST**“Easy 800 DST setting” objects**

This object contains the DST setting of the easy800/MFD.

This object can be read (Read) and written (Write). It can be accessed by a class 1 and/or class 2 DPV1 master.

Use

Use this object to

- read the current DST setting of an easy800/MFD clock.
- transfer a new DST setting of an easy800/MFD clock.

Addressing the object

Use the following information for addressing the object:

- API is 0 (only required for class 2 DPV1 masters)
- Slot number is 0
- Index is 209.

Length of the object

The length of the data to be read or written is 20 octets. Do not enter any different length when calling the “Write” service, or a smaller length when calling the “Read” service, otherwise this will generate an error message.

Variable definition (example) of the object

Declare the following variable (data function block) in an IEC 61131-3 based system in order to read and write the DST setting:

Name	Data type
Easy_800_Clock_DSTSetting	ARRAY [1..20] OF BYTE or ARRAY [1..20] OF USINT

Data content of the object

The following table shows the address location and meaning of the data contained in the object when reading and writing. It also illustrates in an example how to access the data using the defined variable.

Table 318: Address location and meaning of the data

Variable access (example)	Data position	Meaning	Value range (decimal)
Easy_800_Clock_ DST[1]	Octet 1	Changeover condition → table 319	0 to 5
Easy_800_Clock_ DST[2]	Octet 2	Day of the changeover to summer time (the time change occurs at 2.00 a.m. and changes to 3.00 a.m.)	1 to 31 BCD coding
Easy_800_Clock_ DST[3]	Octet 3	Month of changeover for change to summer time	1 to 12 BCD coding
Easy_800_Clock_ DST[4]	Octet 4	Day of the changeover to winter time (the time change occurs at 3.00 a.m. and changes to 2.00 a.m.)	1 to 31 BCD coding
Easy_800_Clock_ DST[5]	Octet 5	Month of changeover for change to winter time	1 to 12 BCD coding
Easy_800_Clock_ DST[6]	Octet 6	Rule 1 for changeover to summer time → table 320	0 to 5
Easy_800_Clock_ DST[7]	Octet 7	Weekday for changeover to summer time → table 322	0 to 6
Easy_800_Clock_ DST[8]	Octet 8	Rule 2 for changeover to summer time → table 321	0 to 2
Easy_800_Clock_ DST[9]	Octet 9	Day of changeover for change to summer time	1 to 31 BCD coding
Easy_800_Clock_ DST[10]	Octet 10	Month of changeover for change to summer time	1 to 12 BCD coding

Variable access (example)	Data position	Meaning	Value range (decimal)
Easy_800_Clock_DST[11]	Octet 11	Hour of changeover for change to summer time	0 to 23 BCD coding
Easy_800_Clock_DST[12]	Octet 12	Minute of changeover for change to summer time	0 to 59 BCD coding
Easy_800_Clock_DST[13]	Octet 13	Time difference	0 to 5
Easy_800_Clock_DST[14]	Octet 14	Rule 1 for changeover to winter time → table 320	0 to 5
Easy_800_Clock_DST[15]	Octet 15	Weekday for changeover to winter time → table 322	0 to 6
Easy_800_Clock_DST[16]	Octet 16	Rule 2 for changeover to winter time → table 321	0 to 2
Easy_800_Clock_DST[17]	Octet 17	Day of changeover for change to winter time	1 to 31 BCD coding
Easy_800_Clock_DST[18]	Octet 18	Month of changeover for change to winter time	1 to 12 BCD coding
Easy_800_Clock_DST[19]	Octet 19	Hour of changeover for change to winter time	0 to 23 BCD coding
Easy_800_Clock_DST[20]	Octet 20	Minute of changeover for change to winter time	0 to 59 BCD coding

Table 319: Changeover conditions for DST change

Value (hex)	Meaning	Note
00	No changeover	Octets 2 to 20 have no meaning
01	The changeover is carried out according to octet 2 and 3 for the change to summer time octet 4 and 5 for the change to winter time	Octets 6 to 20 have no meaning
02	Changeover according to rules for EU	Octets 2 to 20 have no meaning
03	Changeover according to rules for GB	Octets 2 to 20 have no meaning
04	Changeover according to rules for USA	Octets 2 to 20 have no meaning
05	The changeover is carried out according to octet 6 to 13 for the change to summer time octet 13 to 20 for the change to winter time	Octets 2 to 5 have no meaning

If the value 0 is entered in octet 1, there is no DST changeover. Selecting the values 2 to 4 causes an automatic DST change according to the legal requirements for the selected country.

If the value 1 is entered in octet 1, the DST changeover takes place according to the user-defined rules entered in octet 2 to 5.

If the value 1 is entered in octet 5, the DST changeover takes place according to the user-defined rules entered in octet 6 to 20.

Octet 6 contains the first rule for the changeover to summer time, octet 14 the rule for the changeover to winter time. Table 320 indicates the possible rules.

Table 320: Rule 1 for DST changeover

Value (hex)	Meaning	
00	The changeover takes place at the time defined in...	... • Octet 9 to 12 for changeover to summer time • Octet 17 to 20 for changeover to winter time In this case without meaning: • Octet 7 to 8 for changeover to summer time • Octet 15 to 16 for changeover to summer time
01	The changeover takes place on the first weekday defined in...	... • Octet 7 for changeover to summer time • Octet 15 for changeover to winter time in conjunction with the defined rule 2 in: • Octet 8 for changeover to summer time • Octet 16 for changeover to winter time
02 ¹⁾	The changeover takes place on the second weekday defined in...	
03 ¹⁾	The changeover takes place on the third weekday defined in...	
04 ¹⁾	The changeover takes place on the fourth weekday defined in...	
05 ¹⁾	The changeover takes place on the last weekday defined in...	

1) The meaning "in" must be entered in rule 2. Rule 2 is defined in:

- Octet 8 for changeover to summer time • Octet 16 for changeover to winter time

If octet 6 or octet 14 contains one of the values 1 to 5, the second rule must be defined for the changeover to summer time (octet 8) or winter time (octet 16). table 321 indicates the possible rules.

Table 321: Rule 2 for DST changeover

Value (hex)	Meaning	Note
00 ¹⁾	Changeover takes place on the defined weekday and month	The weekday is defined in: • Octet 7 for changeover to summer time • Octet 15 for changeover to winter time
01	Changeover takes place on the defined weekday after the defined date	Month and date are defined in: • Octet 9 to 10 for changeover to summer time • Octet 17 to 18 for changeover to winter time
02	Changeover takes place on the defined weekday before the defined date	

1) In this case without meaning:

- Octet 9 for changeover to summer time
- Octet 17 for changeover to winter time

Table 322: Coding of the weekday

Value (hexadecimal)	Meaning
00	Sunday
01	Monday
02	Tuesday
03	Wednesday
04	Thursday
05	Friday
06	Saturday

Table 323: Coding of the time difference (octet 13)

Value (hexadecimal)	Time difference of the change is
01	30 minutes
02	1 hour
03	90 minutes
04	2 hours
05	150 minutes
06	3 hours

DPV1 error messages

The EASY204-DP generates DPV1 error messages shown in the following table.

Table 324: DPV1 error messages

Error message/code	Meaning	Meaning/Remedy
Access denied (Code B6 hexadecimal)	The addressed object is disabled for the DP master type	You have addressed an object with a class 1 DP master that can only be addressed by a class 2 DPV1 master. Observe the requirements of the object.
	The addressed object is disabled for the selected service.	You have either accessed an object with a "Write" service that can only be read or accessed an object with a "Read" service that can only be written. Observe the requirements of the object.
	The connected Easy/MFD has generated an error code for the selected service	Observe the requirements of the object.
Invalid index (Code B0 hexadecimal)	The index specified in the "Read" or "Write" service is invalid for the EASY204-DP	Use the index specified in the object description
Invalid parameter (Code B8 hexadecimal)	The data length used with the "Read" service is smaller than the data length of the addressed object	Use the data length specified in the object description
	The parameters used for establishing a connection for a class 2 DPV1 master do not match those of the EASY204-DP	The class 2 DPV1 master must set the parameters "Features Supported" to 1 and "Profile Ident Number" to 0 for the connection establishment

Error message/code	Meaning	Meaning/Remedy
Invalid range (Code B7 hexadecimal)	An impermissible value was used for a data content with the "Write" service	Observe the value ranges of the data contents specified in the object description
Invalid slot (Code B2 hexadecimal)	The slot specified in the "Read" or "Write" service is invalid for the EASY204-DP	Use the slot specified in the object description
Resource unavailable (Code C3 hexadecimal)	The addressed object is currently not available since the communication between EASY204-DP and the connected Easy/MFD is faulty	Check the connection between the devices
Write length error (Code B1 hexadecimal)	The data length used with the "Write" service does not match the data length of the addressed object	Use the data length specified in the object description

Appendix

What Happens If ...?

Event	Explanation	Remedy
POW LED not lit	No power supply	Connect and switch on power supply
POW LED flashing	Data transfer via EASY-LINK OK	
BUS LED not lit	No PROFIBUS-DP data communication	Connect and start PROFIBUS-DP
BUS LED lit	Data transfer via PROFIBUS-DP OK	
Slave not signalling	<ul style="list-style-type: none"> – No slave address set – No bus terminating resistor present – Cable, plug faulty – No power supply 	<ul style="list-style-type: none"> – Set slave address – Set bus terminating resistors – Check connection – Provide power supply to device
Write command rejected	<ul style="list-style-type: none"> – Command not permissible – EASY display not on the Status display 	<ul style="list-style-type: none"> – Change command – Show Status display
Actual value is zero	No actual value present	Function relay does not have an actual value or not triggered

Overview of commands**easy600**

The commands are sorted in ascending order:

Command value hex	
01	Write T1 timing relay setpoint
02	Write T2 timing relay setpoint
03	Write T3 timing relay setpoint
04	Write T4 timing relay setpoint
05	Write T5 timing relay setpoint
06	Write T6 timing relay setpoint
07	Write T7 timing relay setpoint
08	Write T8 timing relay setpoint
09	Write C1 counter relay setpoint
0A	Write C2 counter relay setpoint
0B	Write C3 counter relay setpoint
0C	Write C4 counter relay setpoint
0D	Write C5 counter relay setpoint
0E	Write C6 counter relay setpoint
0F	Write C7 counter relay setpoint
10	Write C8 counter relay setpoint
12	Write time switch 1 channel A
13	Write time switch 1 channel B
14	Write time switch 1 channel C
15	Write time switch 1 channel D
16	Write time switch 2 channel A
17	Write time switch 2 channel B
18	Write time switch 2 channel C
19	Write time switch 2 channel D
1A	Write time switch 3 channel A

Command value hex	
1B	Write time switch 3 channel B
1C	Write time switch 3 channel C
1D	Write time switch 3 channel D
1E	Write time switch 4 channel A
1F	Write time switch 4 channel B
20	Write time switch 4 channel C
21	Write time switch 4 channel D
22	Write analog value comparator A1
23	Write analog value comparator A2
24	Write analog value comparator A3
25	Write analog value comparator A4
26	Write analog value comparator A5
27	Write analog value comparator A6
28	Write analog value comparator A7
29	Write analog value comparator A8
2A	Write time
2B	Read time switch 1 channel A
2C	Read time switch 1 channel B
2D	Read time switch 1 channel C
2E	Read time switch 1 channel D
2F	Read time switch 2 channel A
30	Read time switch 2 channel B
31	Read time switch 2 channel C
32	Read time switch 2 channel D
33	Read time switch 3 channel A
34	Read time switch 3 channel B
35	Read time switch 3 channel C
36	Read time switch 3 channel D

Command value hex	
37	Read time switch 4 channel A
38	Read time switch 4 channel B
39	Read time switch 4 channel C
3A	Read time switch 4 channel D
3C	Read time
3D	Read status of analog and digital inputs
3E	Read status of P buttons and operator buttons
3F	Read status of timing relays, counter relays, time switches and analog value comparators
40	Read status of markers, digital outputs and text display markers
41	Read T1 actual value
42	Read T2 actual value
43	Read T3 actual value
44	Read T4 actual value
45	Read T5 actual value
46	Read T6 actual value
47	Read T7 actual value
48	Read T8 actual value
49	Read C1 counter relay actual value
4A	Read C2 counter relay actual value
4B	Read C3 counter relay actual value
4C	Read C4 counter relay actual value
4D	Read C5 counter relay actual value
4E	Read C6 counter relay actual value
4F	Read C7 counter relay actual value
50	Read C8 counter relay actual value

easy800/MFD

		Byte 1	Byte 2	Byte 3
Date and time		Command (hex)	Len ¹ (hex)	Index (hex)
Read/write date and time		93/B3	05	00
Winter/summer time, DST				01
Image data		Byte 1 Command (hex)	Byte 2 Len ¹ (hex)	Byte 3 Type (hex)
Read/write image data	91/B1			
"Local inputs: I1 – I16"		2	01	0
"Read inputs of the stations IW1 to IW8"				1 – 8
"Read local analog inputs IA1 to IA4"			02	1 – 4
"Read local diagnostics ID1 to ID16"			03	0
"Read and write local QW0 outputs/outputs of the stations QW1 to QW8"			04	0/1 – 8
"Read and write local analog output QA1"			05	0
"Read local P buttons"		1	06	0
"Read RW.. inputs/SW.. outputs from EasyLink"		2	07/09	0
"Read receive data network RN1 .. RN32/send data network SN1 .. SN32"				1 – 8
"Read receive data network RN1 .. RN32/send data network SN1 .. SN32"		4	08/0A	1 – 8
Marker bit M1 .. M96		1	0B	1 – 96
Marker byte MB1 .. MB96			0C	1 – 96
Marker word MW1 .. MW96		2	0D	1 – 96
Marker double word MD1 .. MD96		4	0E	1 – 96

1) Len... stands for the number of data bytes to be sent.

Function blocks	Byte 1 Command (hex)	Byte 2 Type (hex)	Byte 3 Instance (hex)
Read/write function blocks	92/B2		
"Receive network data function blocks GT01 .. GT32"		0F	1 – 20
"Analog value comparators A01 .. A32"		11	1 – 20
"Arithmetic function blocks AR01 .. AR32"		12	1 – 20
"Boolean operation function blocks BV01 .. BV32"		13	1 – 20
"Counters C01 .. C32"		14	1 – 20
"Frequency counters CF01 .. CF04"		15	1 – 20
"High-speed counters CH01 .. CH04"		16	1 – 4
"Incremental encoder counters CI01 .. CI02"		17	1 – 2
"Comparators CP01 .. CP32"		18	1 – 20
"Text output function blocks D01 .. D32"		19	1 – 20
"Data function blocks DB01 .. DB32"		1A	1 – 20
"Receive network data function blocks GT01 .. GT32"		1B	1 – 20
"7-day time switches HW01 .. HW32"		1C	1 – 20
"Year time switches HY01 .. HY32"		1D	1 – 20
"Operating hours counters OT01 .. OT04"		1E	1 – 4
"Send network data function blocks PT01 .. PT32"		1F	1 – 20
"Synchronize clock function block SC01"		20	1
"Set cycle time function block ST01"		21	1 – 20
"Block compare function blocks BC01 .. BC32"		25	1 – 20
"Block transfer function blocks BT01 .. BT32"		26	1 – 20
"PID controllers DC01 .. DC32"		27	1 – 20
"Signal smoothing filters FT01 .. FT32"		28	1 – 20
"Value scaling function blocks LS01 .. LS32"		29	1 – 20
"Numerical converters NC01 .. NC32"		2A	1 – 20
"Pulse width modulation function blocks PW01 .. PW02"		2B	1 – 2
"Set cycle time function block ST01"		2C	1
"Value limitation function blocks VC01 .. VC32"		2D	1 – 20

Technical Data**General**

Standards and regulations	EN 55011, EN 55022, IEC/EN 61-4, IEC 60068-2-27, IEC 61158
Dimensions (W × H × D)	35.5 × 90 × 56.5
Weight	150 g
Mounting	Top-hat rail to DIN 50022, 35 mm Screw fixing with fixing brackets ZB4-101-GF1 (accessories)

Ambient temperatures

Ambient temperature Installed horizontally/vertically	Cold to IEC 60068-2-1 Heat to IEC 60068-2-2	-25 to 55 °C
Condensation		Prevent condensation with suitable measures
Storage/transport temperature		-40 to +70 °C
Relative air humidity	IEC 60068-2-30	5 to 95 %, non-condensing
Air pressure (operation)		795 to 1080 hPa
Corrosion resistance	IEC 60068-2-42 IEC 60068-2-43	SO ₂ 10 cm ³ /m ³ , 4 days H ₂ S 1 cm ³ /m ³ , 4 days

Ambient mechanical conditions

Pollution degree		2
Degree of protection	EN 50178 IEC 60529 VBG4	IP20
Oscillations	IEC 60068-2-6	10 to 57 Hz (constant amplitude 0.15 mm) 57 to 150 Hz (constant acceleration 2 g)
Shocks	IEC 60068-2-27	18 shocks (semi-sinusoidal 15 g/11 ms)
Drop	IEC 60068-2-31	Drop height 50 mm
Free fall, packaged	IEC 60068-2-32	1 m

Electromagnetic compatibility (EMC)

Electrostatic discharge	IEC/EN 61 000-4-2, degree of severity 3	8 kV air discharge, 6 kV contact discharge
Electromagnetic fields	IEC/EN 61 000-4-3	Field strength 10 V/m
Radio interference suppression	EN 55011, EN 55022	Limit class A
Burst	IEC/EN 61 000-4-4, degree of severity 3	2 kV supply lines, 1 kV signal lines
High-energy pulses (surge)		
EASY...-DC...	IEC/EN 61000-4-5, degree of severity 2	0.5 kV power cable symmetrical
Line-conducted interference	IEC/EN 61000-4-6	10 V

Dielectric strength

Measurement of the air clearance and creepage distance	EN 50178, UL 508, CSC C22.2 No 142
Dielectric strength	EN 50178

Tools and cable cross-sections

Solid	
min.	0.2 mm ² , AWG 22
max.	4 mm ² , AWG 12
Flexible with ferrule	
min.	0.2 mm ² , AWG 22
max.	2.5 mm ² , AWG 12
Slot-head screwdriver, width	3.5 × 0.8 mm
Tightening torque max.	0.5 Nm

Power supply

Rated voltage	
Rated value	24 V DC, -15 %, +20 %
Permissible range	20.4 to 28.8 V DC
Residual ripple	< 5 %
Input current at 24 V DC	Normally 200 mA
Voltage dips (IEC/EN 61131-2)	10 ms
Power dissipation at 24 V DC	Normally 4.8 W

LEDs

Power LED (POW)	green
PROFIBUS-DP LED (BUS)	green

PROFIBUS-DP

Device connection	SUB-D 9-pole, socket
Electrical isolation	Bus to power supply (simple) Bus and power supply to "easy" basic unit (safe isolation)
Function	PROFIBUS-DP slave
Interface	RS 485
Bus protocol	PROFIBUS-DP
Baud rates	Automatic search up to 12 MBd
Bus terminating resistors	Connectable via plug
Bus addresses	1 to 126 addressable via "easy" basic unit with display or EASY-SOFT
Services	
Inputs module	All data S1 to S8 (EASY6..)
Outputs module	All data R1 to R16 (EASY6..)
Control commands module	Read/Write Time, day, summer/winter time (DST) All parameters of the EASY function relays

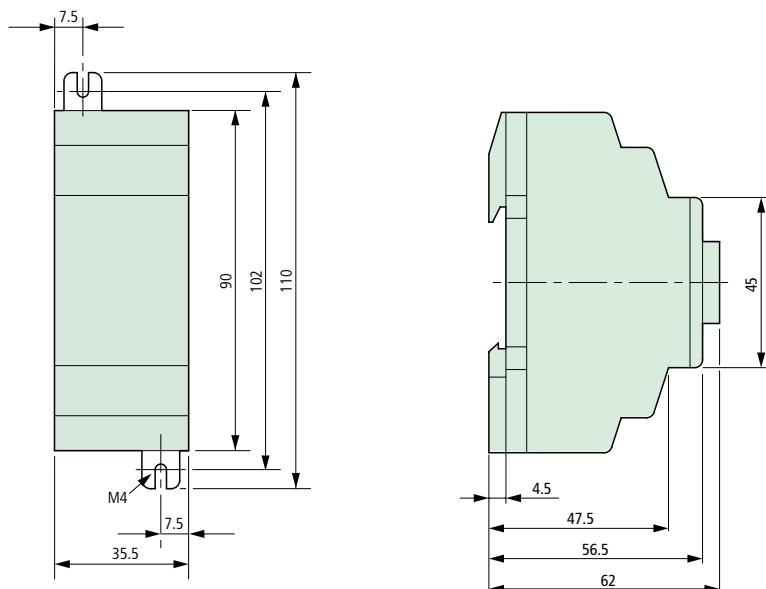
Dimensions

Figure 9: Dimensions EASY204-DP

Glossary

This glossary refers to subjects relating to PROFIBUS-DP.

Bus terminating resistor	Resistor at the beginning and end of a bus line for preventing disturbance caused by signal reflections and for adapting bus cables. Bus terminating resistors must always be the last unit at the end of a bus segment.
Acknowledge	Acknowledgement returned by the receiving station after having received a signal.
Address	Number, for example, for identifying a memory location, a system or a module within a network.
Addressing	Assignment or setting of an address such as for a module in a network.
Active metal component	Conductor or conductive component that is live when in operation.
Automation device	Control device with inputs and outputs that is connected to a technical process. Programmable controllers (PLCs) are a special group of automation devices.
Analog	Value, such as voltage, that is infinitely variable and proportional. Analog signals can acquire any value within specific limits.
Baud	Unit for the data transfer rate. One baud corresponds to the transmission of one bit per second (bit/s).
Baud rate	Unit of measure of the data transmission speed in bit/s.
Electrical equipment	All objects that are used for the generation, conversion, transfer, distribution and use of electric power, such as conductors, cables, machines, control devices.

Reference ground	Ground potential in the area of grounding devices. Unlike "ground", which always has zero potential, it may have any potential except zero.
Reference potential	Represents a reference point for measuring and/or visualising the voltage of any connected electrical circuits.
Bidirectional	Operation in both directions.
Lightning protection	Represents all measures for preventing system damage due to overvoltage caused by lightning strike.
Bus	Bus cable system for data exchange between CPU, memory and I/O level. A bus can consist of several parallel segments, such as 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	The entirety of all units which communicate across a bus.
Bus cycle time	Time interval in which a master will serve all slaves or stations in a bus system, i.e. writes their outputs and reads their inputs.
CPU	Abbreviation for "Central Processing Unit". Central unit for data processing, which represents the core element of a computer.
Digital	A value, for example voltage, that can only be represented by a certain number of states within a defined range, usually defined as 0 and 1.
DIN	Abbreviation for "Deutsches Institut für Normungen e.V." (German standards institute).
EMC	Abbreviation for "Electromagnetic Compatibility". The ability of electrical equipment to function trouble-free within a particular environment without a negative effect on the environment concerned.
EN	Abbreviation for "European Norm" or European standard.

Ground	In electrical engineering the name for conductive grounding with an electrical potential at any point equal to zero. In the environment of grounding devices, the electrical ground potential may not equal zero. This is called a "reference ground".
ground (verb)	Represents the connection of an electrically conductive component to the equipotential earth via a grounding device.
Ground connection	One or several components that have a direct and good contact with the ground.
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 their highly reliable transfer of signals and real-time response.
Field supply	Voltage supply to field devices as well as signal voltage.
Galvanic coupling	A galvanic coupling occurs when two circuits use the same cable. Typical sources of interference are, for example, starting motors, static discharges, clocked devices, and a potential difference between the housing of components and the common power supply.
GND	Abbreviation for "GROUND" (0 potential).
GSD	The device master data files (GSD) contain standardized PROFIBUS station descriptions. They are used to simplify the configuration of the DP master and DP slaves.
hexadecimal	Number system with base 16. Counting from 0 to 9 and then with the letters A, B, C, D, E and F.
I/O	Abbreviation for "Input/Output".
Impedance	Apparent resistance that a component or circuit of several components has for an alternating current at a particular frequency.

Low impedance connection	Connection with low alternating-current resistance.
Inactive metal parts	Conductive parts that cannot be touched and which are insulated from active metal parts. They can, however, carry voltage in the event of a fault.
Inductive coupling	Inductive (magnetic) coupling occurs between two current carrying conductors. The magnetism produced by the currents induces an interference voltage. Typical interference sources are, for example transformers, motors, mains cables installed parallel and RF signal cables.
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-capable modules	Command-capable modules are modules with an internal memory that are capable of executing particular commands (such as output substitute values).
Configuring	Systematic arrangement of the I/O modules of a station.
Protected against short-circuit	Property of electrical equipment. Short-circuit-proof equipment has the ability to withstand the thermal and dynamic loads that may occur at the location of installation on account of a short-circuit.
LSB	Abbreviation for "Least Significant Bit". Bit with the least significant value.
Chassis ground	Entirety of all interconnected inactive equipment parts that do not have any contact voltage, even in the event of a fault.
Grounding strip	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 proximity switches represent a special category of 2-wire proximity switches. They are highly resistant to interference and reliable due to their special construction, e.g. low internal resistance, few components and short design.
Overhead	System management time required in the system in each transmission cycle.
Parameter assignment	Assignment of parameters in the configuration software of the DP master for the individual stations on the bus and their modules.
Equipotential bonding	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.

PROFIBUS-DP	PROFIBUS bus system with the DP protocol. DP stands for "decentralized periphery". PROFIBUS-DP is based on DIN 19245 Part 1+4, and was integrated in the European fieldbus standard EN 50170. It is used for high-speed data exchange between the central DP master and the decentralized peripheral devices, the DP slaves. The comprehensive use is implemented by means of a multi-master concept.
PROFIBUS-DP address	Each PROFIBUS-DP station is assigned an unambiguous PROFIBUS-DP address by means of which it can be addressed by the master.
PROFIBUS-DP master	The PROFIBUS-DP master is the central station and controls the PROFIBUS access of all PROFIBUS-DP slaves.
PROFIBUS-DP slave	PROFIBUS-DP slaves are addressed by the PROFIBUS-DP master and exchange data with it at its request.
Response time	In a bus system the time interval between the sending of a read job and the receipt of the response. Within an input module, it represents the time interval between the signal change at an input and its output to the bus system.
Repeater	Amplifier for signals transferred across a bus.
RS 485	Serial interface in accordance with the EIA standard for high-speed data transmission via several transmitters.
Shield	Term that describes the conductive covering of cables, cubicles and cabinets.
Shielding	All measures and equipment used for connecting system parts with the shield.
Protective conductor	A conductor required for the protection against dangerous currents, designated by the letters PE (abbreviation of "Protective Earth").
Serial	Describes an information transfer technique. Data is transferred in a bit-stream across the cables.

Slave	Station 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.
Radiated coupling	Radiated coupling occurs when an electromagnetic wave makes contact with a conductor structure. The impact of the wave induces currents and voltages. Typical interference sources are, for example ignition circuits (spark plugs, commutators of electrical motors) and transmitters (e.g. radio-operated devices), which are operated near the corresponding conductor structure.
SUB-D plug	9-pole plug for connecting the fieldbus.
Topology	Geometric structure of a network or circuit arrangement.
UART	Abbreviation for “Universal Asynchronous Receiver/Transmitter”. A “UART” is a logic circuit used for converting an asynchronous serial data sequence into a bit-parallel data sequence or vice versa.
Unidirectional	Working in one direction.

Index

Num.	7-day time switch	
	easy600 (DPV1), read status	254
	easy600 (DPV1), read/write parameters	273
	easy700	130, 147
	easy700 (DPV1), read and write	336
	easy700 (DPV1), read status	296
	easy800/MFD (read)	215
A	Address range	30
	Analog comparators	
	easy600 (DPV1), read status	254
	easy600, read status	69
	easy600, write	73
	easy700 (DPV1), read and write	325
	easy700 (DPV1), read status	294
	easy700, read and write	132, 325
	easy700, read status	110
	easy800/MFD (read/write)	183
	Analog inputs	
	easy600 (DPV1), read status	249
	easy600, read status	81
	easy700, read status	115
	easy700/800/MFD (DPV1), read status	288
	easy800/MFD, read status	164
	Analog output	
	easy800/MFD (read/write)	170
	Analog value comparators	
	easy600 (DPV1), write parameters	271
	Arithmetic function block	
	easy800/MFD (read/write)	185
B	Bit array	106
	Block Compare, read/write (easy800/MFD)	187
	Block Transfer, read/write (easy800/MFD)	189
	Boolean operation, read/write (easy800/MFD) ...	191

Bus cable lengths	26
BUS LED	33
Bus terminating resistors	24

C	Comparators	
	easy800/MFD (read/write)	201
	Comparison values	
	easy600	73
	Control commands	
	easy600	59
	easy700	101
	easy800/MFD	153
	Counter	
	easy600 (DPV1), read status	254
	easy600 (DPV1), read/write parameters	268
	easy700 (DPV1), read and write	327
	easy700 (DPV1), read status	306
	easy700, read status	111
	easy800/MFD (read/write)	193
	Counter relay	
	easy600, read actual value	76
	easy600, read status	69
	easy600, write setpoint	78
	easy700	135
	Cycle time	34

D	Data	
	input	48
	Output	50
	Data exchange procedure	
	easy600	59
	easy700	101
	easy800/MFD	153
	Data function block, read/write (easy800/MFD)	206
	Debounce	48
	Diagnostics, local	
	easy800/MFD (read)	166

Digital inputs	
easy600 (DPV1), read status	249
easy600, read status	81
easy700, read status	113
easy700/800/MFD (DPV1), read status	286
easy800/MFD, read status	161
Digital outputs	
easy600 (DPV1), read status	251
easy600, read status	65
easy700, read status	124
easy700/800/MFD (DPV1), read status	290
E	Electrical isolation 25
Error codes, via EASY-LINK	
easy700	150
F	Frequency counter
easy800/MFD (read/write)	195
Function block data	
Read, easy700 (DPV1)	321
Selection, easy700 (DPV1)	319
Write, easy700 (DPV1)	323
Function blocks, overview	
easy600	72
easy600 (DPV1)	242
easy700	131
easy700 (DPV1)	283
easy800/MFD	182
G	GSD file 44
H	Hardware requirements 19
High-speed counters	
easy800/MFD (read/write)	197
Hour	
easy600	62

I	Image data	
	General information	64
	Overview easy600 (DPV1)	242
	Overview easy700	109
	Overview easy700 (DPV1)	282
	Overview of easy800/MFD	160
	Overview, easy600	64
	Incremental encoder counters	
	easy800/MFD (read/write)	199
	Input data, definition	47
	Inputs	
	easy800/MFD, read status	164
	Inputs of EASY-LINK	
	easy600 (DPV1), read status	259
	easy700, read status	125
	easy800/MFD, read status	173
	Inputs, local	
	easy800/MFD, read status	161
	Inputs, network stations	
	easy800/MFD, read status	163
	Invalid operating mode	150
	Invalid telegram	150
L	Local inputs	
	easy700, read status	113
	Local outputs	
	easy700, read status	124
	easy800/MFD (read/write)	170
	Local P buttons	
	easy800/MFD (read)	171
M	Marker relay	
	easy600, read status	65
	Markers	
	easy600 (DPV1), read status	251
	easy700 (DPV1), read status	310, 312
	easy700 (DPV1), write	314, 316
	easy800/MFD (read/write)	177

Master reset	
easy700	129
easy700 (DPV1), read status	300
Minute	
easy600	63
Module	
Inputs 1 byte	50
Outputs 1 byte	54
Outputs 3 bytes	50
module	
Inputs 3 bytes	48

N	Network data	
	easy800/MFD (read)	213, 229
	Numerical converters	
	easy800/MFD (read/write)	225
O	Operating buttons	
	easy600 (DPV1), read status	257
	Operating buttons (easy600)	68
	Operating hours counter	
	easy700	138
	easy700 (DPV1), read and write	329
	easy700 (DPV1), read status	308
	easy800/MFD (read/write)	227
	Operating mode, invalid	150
	Operating system version requirements	19
	Output data, definition	47
	Outputs of EASY-LINK	
	easy600 (DPV1), read status	261
	easy700, read status	125
	easy800/MFD, read status	173
	Outputs, local and network stations	
	easy800/MFD (read/write)	168

P	P buttons	
	easy600 (DPV1), read status	257
	easy600, read status	68
	easy700, read status	122
	easy700/800/MFD (DPV1), read status	292
	PID controllers	
	easy800/MFD (read/write)	208
	POW LED	33
	Power supply	22
	PROFIBUS-DP connection assignment	23
	Pulse width modulation	
	easy800/MFD (read/write)	231
R	Reaction times (basic unit)	34
	Read/write date	
	easy600	61
	easy600/700 (DPV1)	276
	easy700	104
	easy800	156
	Read/write function block data, procedure for	
	easy700 (DPV1)	318
	Read/write time	
	easy600	61
	easy600/700 (DPV1)	276
	easy700	104
	easy800	156
	Real-time clock	156
	Receive data, network stations	
	easy800/MFD	175
	easy800/MFD (read)	213
	Receiving, network data (easy800/MFD)	213
	Resetting, easy/MFD inputs/outputs	50
S	Send data, network stations	
	easy800/MFD	175
	easy800/MFD (read)	229
	Set cycle time	
	easy800/MFD (read/write)	234

Set operating mode	51
easy600/700 (DPV1)	247
Setting the station address	30
Setting, easy/MFD inputs/outputs	50
Signal smoothing filter	
easy800/MFD (read/write)	211
Status display	
easy outputs S1 to S8	49
easy/MFD inputs	52
Summer time	
easy600	61
easy600 (DPV1)	279
easy700	105
easy700 (DPV1)	338
easy800/MFD	157
Switching rule	106
Synchronize clock	
easy800/MFD (read)	233

T	Telegram, invalid	150
	Text display	
	Read (easy600)	65
	Text function block	
	easy700 (DPV1), read status	302
	easy700, read status	112
	Text markers	
	easy600 (DPV1), read status	251
	Text output function block	
	easy800/MFD (read/write)	203
	Threshold value comparator	
	easy700, read status	110
	Threshold value switch	
	easy700	132, 325
	easy700 (DPV1), read and write	325
	Time change	
	easy800	157
	Time switch	
	easy600 (read)	94
	easy600, read status	69
	Write (easy600)	98

Timing relay		
easy600 (DPV1), read status	254	
easy600 (DPV1), read/write parameters	263	
easy600, read actual value	84	
easy600, read status	69	
easy600, write setpoint	87	
easy700	140	
easy700 (DPV1), read and write	331	
easy800/MFD (read/write)	234	
Timing relays		
easy700 (DPV1), read status	304	
easy700, read status	127	
Toggle byte		
easy600	60	
easy700	102	
easy800/MFD	154	
Transfer rates	25	
Transmit data, network stations		
easy800/MFD)	175	
V	Value limitation	
	easy800/MFD (read/write)	239
	Value scaling	
	easy800/MFD (read/write)	221
	Version history, easy800	155
W	Weekday	
	easy600	62
	Winter time	
	easy600	61
	easy600 (DPV1)	279
	easy700	106
	easy700 (DPV1)	338
	easy800/MFD	157
	Write input data to EASY-LINK	
	easy600 (DPV1), read status	259

Y	Year time switch	
	easy700	144
	easy700 (DPV1), read and write	334
	easy700 (DPV1), read status	298
	easy700, read status	128
	easy800/MFD (read)	218