Instruction manual

Communication profile

for cam controls by
Deutschmann Automation

Deutschmann Automation GmbH & Co. KG Carl-Zeiss-Str. 8 D-65520 Bad Camberg
Tel: +49-(0)6434/9433-0 Fax: +49-(0)6434/9433-40
e-mail: mail@deutschmann.de Internet: www.deutschmann.de

Foreword

This operating manual provides users and OEM customers with all the information necessary for the installation and operation of the product described in this manual.

All details contained in this manual have been checked carefully, however, they do not represent an assurance of product characteristics. No liability can be accepted for errors. DEUTSCHMANN AUTOMATION reserves the right to carry out alterations to the described products in order to improve the reliability, function or design thereof. DEUTSCHMANN AUTOMATION only accepts liability to the extent as described in the terms and conditions of sale and delivery.

Version 7.0, 7.5.08, Art.-No. V2064E

P/C: B

1 Introduction . . . 9
1.1 Symbols . . . 9
1.2 Definitions . . . 9
1.2.1 Abbreviations . . . 9
1.2.2 Concepts . . . 9
1.2.3 Suggestions . . . 9
2 Communication description . . . 10
2.1 Hardware . . . 10
2.1.1 RS232 interface . . . 10
2.1.2 DICNET® bus interface . . . 10
2.1.3 Fieldbus interface . . . 10
2.2 General communication structure . . . 10
2.3 Communication sequence . . . 11
2.3.1 Protocol frame for DICNET and RS232 . . . 11
2.3.2 Protocol frame for Interbus-S . . . 11
2.3.2.1 Process data (output) in normal operation . . . 11
2.3.2.2 Process data (output) in error state . . . 11
2.3.2.3 Process data (input) . . . 12
2.3.2.4 Parameter exchange . . . 12
2.3.2.5 Time response of the communication sequence . . . 12
2.3.3 Protocol frame for LonWorks . . . 13
2.3.3.1 Response of the parameters and the process data . . . 13
2.3.4 Protocol frame for Profibus-DP . . . 14
2.3.4.1 Communication via Gateway . . . 14
2.3.4.1.1 Configuration parameter data . . . 14
2.3.4.1.2 Configuration of parameters and process data . . . 15
2.3.4.2 Direct communication with the cam control . . . 15
2.3.4.2.1 Parameter data only . . . 15
2.3.4.2.2 Parameter and process data (2 byte logic) . . . 16
2.3.4.2.3 Parameter and process data (1 byte logic) . . . 17
2.3.5 Protocol frame for ARCNET . . . 18
2.3.6 Protocol frame for DeviceNet . . . 19
2.3.6.1 Polling . . . 19
2.3.6.2 Bit-Strobe . . . 19
2.3.6.3 Change of state . . . 19
2.3.6.4 Parameterdata . . . 20
2.3.6.5 DeviceNet-Master to the Gateway . . . 20
2.3.6.6 Gateway to DeviceNet-Master . . . 20
2.3.6.7 Error messages . . . 20
2.3.6.7.1 Example . . . 21
2.3.6.8 Processdata . . . 21
2.3.6.9 DeviceNet-Master to the Gateway . . . 21

2.3.6.10 Gateway to the DeviceNet-Master . . . 21
2.3.6.11 Processdata without error report . . . 22
2.3.6.12 Processdata with DeviceNet-error report . . . 22
2.3.6.13 Processdata with Gateway-error report . . . 22
2.3.6.13.1 Example . . . 22
2.3.6.14 Error acknowledgement . . . 22
2.3.7 Protocol-frame for Ethernet . . . 22
2.3.7.1 Data from the client to the server (Gateway) . . . 23
2.3.7.2 Data from the server (Gateway) to the client . . . 23
2.3.7.3 Example with the DEUTSCHMANN Ethernet starterkit . . . 23
2.3.8 Command table . . . 28
2.3.9 Parameter table . . . 29
2.3.9.1 PNR_ENCODER_TYP - encoder type . . . 30
2.3.9.2 PNR_RESOLUTION_PER_TURN . . . 30
2.3.9.3 PNR_ENCODER_INVERT . . . 30
2.3.9.4 PNR_LANGUAGE - language selection . . . 30
2.3.9.5 PNR_DEADTIME_TYP . . . 30
2.3.10 Bitnumbers of the parameter PNR_STATUS_FLAGS . . . 30
2.3.11 Bitnumbers of the parameter PNR_ACTIVE_STATUS . . . 31
2.4 Error handling . . . 31
2.5 Command descriptions . . . 31
2.5.1 Parameter received from PLS . . . 31
2.5.1.1 GET_OUTPUT . . . 31
2.5.1.2 GET_INPUT . . . 31
2.5.1.3 GET_NEXT_CAM . . . 32
2.5.1.4 GET_BACK_CAM . . . 32
2.5.1.5 GET_IDLETIME . . . 32
2.5.1.6 GET_POSITION . . . 33
2.5.1.7 GET_SPEED . . . 33
2.5.1.8 GET_STATUS . . . 33
2.5.1.9 GET_OUT_POS . . . 33
2.5.1.10 GET_DISPLAY . . . 34
2.5.1.11 GET_LOGIC . . . 34
2.5.1.12 GET_DATA_EXIST . . . 35
2.5.1.13 GET_GATEWAY_ID . . . 35
2.5.1.14 GET_PARAMETER . . . 35
2.5.1.15 GET_OUTPUT_NAME . . . 36
2.5.1.16 GET_GATEWAY_DATA . . . 36
2.5.1.17 GET_EEROM_BLOCK . . . 36
2.5.1.18 GET_L2000_DATA . . . 37
2.5.2 Send parameters to PLS . . . 37
2.5.2.1 SET_CAM_NEW . . . 37
2.5.2.2 SET_IDLETIME . . . 37

2.5.2.3 SET_ERROR_QUIT . . . 38
2.5.2.4 SET_LOGIC . . . 38
2.5.2.5 SET_CAM_MOVE . . . 39
2.5.2.6 SET_CAM_CHANGE_SHORT . . . 39
2.5.2.7 SET_GATEWAY_ID . . . 39
2.5.2.8 SET_CAM_CHANGE_MT (only MT) . . . 40
2.5.2.9 SET_PARAMETER . . . 40
2.5.2.10 SET_OUTPUT_NAME . . . 41
2.5.2.11 SET_EEROM_BLOCK . . . 41

3 Examples . . . 42

3.1 Transfer with DICNET or RS232 . . . 42
3.1.1 Reading the initial state of a LOCON 32 . . . 42
3.1.2 Transfer with InterBus-S . . . 42
3.1.2.1 Scanning the cam control type . . . 42
3.2 Sample-source-code for accessing the communication routine . . . 42

4 Appendix A . . . 51

4.1 Command tale (not for new applications) . . . 51
4.1.1 Explanations . . . 51
4.2 Description of commands not to be used any more . . . 51
4.2.1 Parameter received by the PLS . . . 51
4.2.1.1 GET_TYP . . . 51
4.2.1.2 GET_MAXPARA . . . 52
4.2.1.3 GET_CONFIG_PARA . . . 52
4.2.1.4 GET_SYSTEM_PARA . . . 52
4.2.1.5 GET_OUT_NAME . . . 53
4.2.1.6 GET_PROT_REC . . . 53
4.2.1.7 GET_GEAR_PARA . . . 53
4.2.1.8 GET_OUTPUT_MATTE (only pattern unit) . . . 54
4.2.2 Send parameters to PLS . . . 54
4.2.2.1 SET_ACT_PARA . . . 54
4.2.2.2 SET_CAM_CHANGE . . . 54
4.2.2.3 SET_CONFIG_PARA . . . 55
4.2.2.4 SET_SYSTEM_PARA . . . 55
4.2.2.4.1 SET_OUT_NAME . . . 55
4.2.2.5 SET_GEAR_PARA . . . 56
4.2.2.6 SET_OUTPUT_MATTE (only pattern unit) . . . 56

1 Introduction

To an increasing extent, DEUTSCHMANN AUTOMATION is supporting the use of cam controls with remote control and display unit in order to meet market demands.

Since different combinations of cam control and terminal are required in each individual case, depending on the particular application, it was necessary to define a uniform interface (communication profile) supported by all terminals and cam controls in the DEUTSCHMANN AUTOMATION range.
This enables any user to assemble the best combination for his purposes.

Since this communication profile is disclosed in these specifications, the user is also able to communicate with DEUTSCHMANN cam controls and, thus, use existing information (encoder position, speed,...) for his own application or operate the cam control via his own terminal.

On the basis of this communication profile, the user even has the option of exchanging data with higher-level bus systems (Interbus-S, PROFIBUS, CAN ...).

1.1 Symbols

Particularly **important text sections** can be seen from the adjacent pictograph.

You should **always** follow this information since, otherwise, this could result in malfunctions or operating errors.

1.2 Definitions

1.2.1 Abbreviations

Abbreviation	Significance
MT	Multiturn encoder
IBS	Interbus-S
PLS	Cam control
ERSTE_NOCKE	7F00 (hex)
ERSTE_NOCKE (only MT)	FF7F00 (hex)
ITC	Idle time compensation
ATC	Angle-time cam

1.2.2 Concepts

Scaled: With cam control with the function ’scalable encoder value’ also known by the expression ’transmission factor’ a thorough differentiation has to be made between the real and the scaled values. For that reason it is always specified at the corresponding parameter if the values are real or scaled.

1.2.3 Suggestions

We are always pleased to receive suggestions and wishes etc. and endeavour to allow for these. It is also helpful if you bring our attention to any errors.

2 Communication description

2.1 Hardware

The communication profile is independent of the actual hardware interface and defines the services made available by the cam control in accordance with layer 7 of the ISO/OSI model. One of the interfaces described below can be used as the communication medium provided it is supported by the cam control used.

2.1.1 RS232 interface

Transfer on the RS232 interface occurs in full-duplex mode at 9,600 baud, with 8 data bits, 1 start bit, 1 stop bit and no parity bit.
Only a pure point-to-point link is possible between the cam control and a user.

2.1.2 DICNET® bus interface

DICNET® (DEUTSCHMANN Industrial Controller Net) is a field bus whose physical layer corresponds to the ISO/OSI Layer Model of DIN 19245, Part 1, i. e. a link is established between all users in the network by means of one RS485 two-wire line.
Thus, the physical array is a bus system to which the users can be connected and disconnected as required.
Viewed logically, it is a Token Ring, i. e. only the user with the token (bus access authorization) may ever send on the bus. If the user has no data for another user, the user forwards the token to its neighbor which was determined in a configuration phase.
This principle achieves a deterministic bus cycle time, i. e. the time (worst-case) until a data packet can be sent can be computed precisely.
Automatic configuration occurs when a user is connected or disconnected.
The transfer rate is 312.5 kbaud with a length of 11 bits per byte. A maximum of 127 users may be operated on one bus, whereby data packets of maximum 14 bytes per cycle can be sent.
An automatic check is conducted on the information received and an error message is issued if a transfer error occurs twice.
Use of this interface prerequires precise knowledge of the internal structure of the DEUTSCHMANN bus (DICNET) so that we shall not discuss this in further detail at this point.

2.1.3 Fieldbus interface

A conversion of RS232 or DICNET to all significant fieldbuses is possible. For that purpose a Gateway of the series UNIGATE RS is used, that connects a fieldbus with a maximum of 16 cam controls.
Detailed information on the fieldbus connection can be found in chapter 2.3.
Moreover Deutschmann Automation is offering cam controls with integrated fieldbuses.

2.2 General communication structure

A strict Master-Slave hierarchy is used for communication, whereby the cam control always operates as the Slave which sends data only in response to a request from the connected communication partner (terminal or PC or PLC etc.).

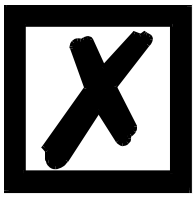
2.3 Communication sequence

2.3.1 Protocol frame for DICNET and RS232

Each data record exchanged by the PLS or terminal has the following standard structure if using the RS232 or DICNET interface:

Byte No.	Designation	Significance
1	Ctrl-K (0B hex)	Constant code character
2	Length	Record length byte 3 (incl.) - byte N+4 (incl.)
3	Own network ID	Own address in DICNET (= 0 with RS232)
4	Command	Command in accordance with command table (see below)
5	Parameter 1	Parameter 1
...	...	...
N+4	Parameter N	Parameter n
N+5	Checksum	Checksum (XOR byte 2 to byte N+4)

The length is determined as of byte "Network ID" through to "Parameter N" (inclusive in each case). The checksum and the first two bytes are not included.
The checksum is determined from the Exclusive-Or operation (XOR) on bytes "Length" to "Parameter N" (inclusive in each case).



A maximum of 9 parameters is possible!

2.3.2 Protocol frame for Interbus-S

The Interbus-S has the following special features compared to the interfaces described above:

- Fixed data length (can be configured via WINGATE) up to 32 bytes (from the 5th byte on the output states of the device are presented)
- Constant cyclic data exchange
- No network address
- Independent data security

For this reason, communication follows the procedure described below if using the Interbus interface.

2.3.2.1 Process data (output) in normal operation

The cam control normally transfers the following data in each Interbus-S cycle:

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	....
Position High (00..7F)	Position Low	Speed High (00..7F)	Speed Low	Output 1-8	Output 9-16	Output 17-24	Output ....

In this case, please note that the most significant bit (MSB) in the 1st byte and in the 3rd byte is **always** 0, i. e. the position and speed are always in the range 0..32767.

2.3.2.2 Process data (output) in error state

The data record is changed as follows if the cam control detects an error.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	7th byte	....
Position High (00..7F)	Position Low	ErrorNo. High (80..FF)	ErrorNo. Low	Output 1-8	Output 9-16	Output 17-24	Output ....

Please note that, in error state, the most significant bit (MSB) of the 3rd byte is **always** 1, i. e. the error number has an offset of 32768. If, for example, the cam control detects an encoder error (error number 100), value 32838 is transferred as the 3rd and 4th byte.
This data record (with the error number) is transferred cyclically until the error has been reset with command "SET_ERROR_QUIT".

2.3.2.3 Process data (input)

If the Master requires no data apart from the cyclic process data (see above), it sends a process data record in which the MSB is not set in the 1st byte (00..6F).
The remaining bytes are not evaluated.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	....	n byte
00..6F	x	x	x	x	x	x	x

2.3.2.4 Parameter exchange

Parameters are exchanged by transferring a parameter set **instead of** a process data record (see above) in an Interbus-S cycle.
It must be noted that, in a parameter set, the MSB in the 1st byte is **always** set, i. e. 128 must be added to the command code (01H..6FH).
A distinction can thus be made between parameter set and process data record by the MSB of the 1st byte.

Since it is a Master-Slave system, a parameter exchange is **always** initiated by the Master, i. e. the Master transfers the required command from the command list described below instead of the process data record (see above).
The structure of this parameter set is as follows:

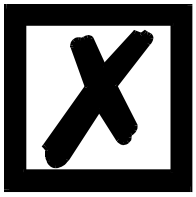
1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	....	10th byte
Command 80..EF	1st para	2nd para	3rd para	4th para	5th para	....	9th para

Please note that always that amount of parameters are transferred as the IBS-length minus 1. If less parameters are specified in the corresponding command, zeros are transferred as the remaining parameters.
The cam control now evaluates this request from the Master and responds, in turn, with a parameter set which has the following structure if no error has occurred:

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	....	10th byte
Command 80..EF	1st para	2nd para	3rd para	4th para	5th para	....	9th para

The 1st byte is identical to the 1st byte of the request.
In **the event of an error** (see chapter Error recovery), the error code (F0..FF) is transferred in the 1st byte.

1st byte	2nd byte	3rd byte	4th byte	5th byte	6th byte	....	10th byte
Command F0..FF	1st para	2nd para	3rd para	4th para	5th para	....	9th para



There is always a response to a request from the Master!

2.3.2.5 Time response of the communication sequence

If the Master initiates a request to the cam control, it may take a few Interbus-S-cycles before the response occurs.
In order to achieve an acceptable time response when exchanging parameters, the Master must

transfer the parameter data record until it has been answered by the cam control. Likewise, the PLS continues to send the unchanged data record until it receives a new data record.

2.3.3 Protocol frame for LonWorks

In the network the Gateway operates as Lon participant and it cyclically writes the answers into the parameterized SNVT from the first request on.

Each request has the following format:

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length	Byte 3 (incl) up to last parameter (incl)
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	

In the Lon-bus the Lon-participants transfers the parameterized SNVTs to the Gateway, however, only those parameters are valid that are specified with the length byte (valid parameters = length -2).
The answer record of the Gateway depends on the configurations of the SNVTs. The total length results from the length of the single SNVTs.
The following types could for instance be parameterized as input and output: Type 36, 86 or 96.

2.3.3.1 Response of the parameters and the process data

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length or error code	Byte 3 (incl) up to last parameter (incl) or error number, if value > 127
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Pos. H	
15	Pos. L	
16	Speed H	
17	Speed L	

Here only those parameters are valid that are specified with the length byte (as for the request).
The response record is valid if the bytes 1, 3 and 4 are identical with the corresponding bytes in the request record.
In case of an error during the request at the cam control, the corresponding error will be transferred in byte 2 (instead of the length). The error value always exceeds 127 and can be adopted from the chapter “error recovery”.

2.3.4 Protocol frame for Profibus-DP

2.3.4.1 Communication via Gateway

In the network the Gateway operates as mere Profibus-Slave and evaluates the master requests cyclically.

Each master request has the following format:

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length	Byte 3 (incl) up to last Parameter (incl)
3	ID (0..15)	Cam control ID
4	Command	Command according to Command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	

In the Profibus always 13 byte are transferred from the Master to the Gateway (Slave), nevertheless the only valid parameters are those specified with the length byte (valid Parameter = length - 2).
The Gateway's response record has a length of 13 or 17 bytes depending on the configuration (see GSD-file) and looks as follows:

2.3.4.1.1 Configuration parameter data

Valid, if 13 bytes data are configured to the Profibus-Master via the GSD-file.

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length	Byte 3 (incl.) up to last parameter (incl.)
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	

2.3.4.1.2 Configuration of parameters and process data

Valid, if 17 bytes ar configured to the Profibus-Master

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length or error code	Byte 3 (incl) up to last parameter (incl) or error number, if value > 127
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Pos. H	Optional at 17-byte configuration
15	Pos. L	
16	Speed H	
17	Speed L	

Here only those parameters are valid that are specified with the length byte (as for the master request).
The response record is valid if the bytes 1, 3 and 4 are identical with the corresponding bytes in the request record.
In case of an error during the request at the cam control, the corresponding error will be transferred in byte 2 (instead of the length). The error value always exceeds 127 and can be adopted from the chapter “error recovery”.

2.3.4.2 Direct communication with the cam control

Depending on the projecting through the GSD-file one of the following protocol-frames is possible:

2.3.4.2.1 Parameter data only

Master inquiry:

Byte No.	Marking	Significance
1	Order number	Clear designation for inquiry
2	Length	Byte 3 (incl.) up to last parameter (incl.)
3	Kind of order	0 = unique, 1 = cyclic
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	

Response from cam control:

Byte No.	Marking	Significance
1	Ctrl-K (0B Hex)	Code character
2	Length	Byte 3 (incl.) up to last parameter (incl.)
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	

2.3.4.2.2 Parameter and process data (2 byte logic)

Master inquiry:

	Marking	Significance
1	Order number	Clear designation for inquiry
2	Length	
3	Kind of order	0 = unique, 1 = cyclic
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Logic input 1 - 8	Inputs for logic link
15	Logic input 9 - 16	Inputs for logic link

Response from cam control:

Byte No.	Marking	Significance
1	Order number	Clear designation for inquiry
2	Length	Byte 3 (incl.) up to last parameter (incl.)
3	Kind of order	0 = unique, 1 = cyclic
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Position 31 - 24	Position as 32 bit-logic value
15	Position 23 - 16	
16	Position 15 - 8	
17	Position 7 - 0	
18	Speed H	Speed
19	Speed L	
20	Output 1 - 8	Output 1 = LSB
21	Output 9 - 16	
22	Output 17 - 23	
23	Output 24 - 31	
24	Output 32 - 40	
25	Output 41 - 48	
26	Current program	
27	Error number	

2.3.4.2.3 Parameter and process data (1 byte logic)

Master inquiry:

	Marking	Significance
1	Order number	Clear designation for inquiry
2	Length	
3	Kind of order	0 = unique, 1 = cyclic
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Logic input 1 - 8	Inputs for logic link

Response from cam control:

Byte No.	Marking	Significance
1	Order number	Clear designation for inquiry
2	Length	Byte 3 (incl.) up to last parameter (incl.)
3	Kind of order	0 = unique, 1 = cyclic
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
10	Parameter 6	
11	Parameter 7	
12	Parameter 8	
13	Parameter 9	
14	Position 31 - 24	Position as 32 bit-logic value
15	Position 23 - 16	
16	Position 15 - 8	
17	Position 7 - 0	
18	Speed H	Speed
19	Speed L	
20	Output 1 - 8	Output 1 = LSB
21	Output 9 - 16	
22	Output 17 - 23	
23	Output 24 - 31	
24	Output 32 - 40	
25	Output 41 - 48	
26	Current program	
27	Error number	

2.3.5 Protocol frame for ARCNET

In the ARCNET the Gateway operates as a mere multi-master and updates the corresponding transmitting buffer after each master request.

Each master request has the following format:

	Marking	Significance
1	ID (0..255)	Destination ID of the desired Gateway
2	Ctrl-K (OB Hex)	Code character
3	Length	Byte 3 (incl.) up to last parameter (incl.)
4	ID (0..15)	Cam control ID
5	Command	Command according to command table
6	Parameter 1	
7	Parameter 2	
8	Parameter 3	
9	Parameter 4	
10	Parameter 5	
11	Parameter 6	
12	Parameter 7	
13	Parameter 8	
14	Parameter 9	

In the ARCNET always 18 bytes are transferred, however, only those parameters are valid that are specified with the length byte (valid parameter = length - 2).

The response record of the Gateway has a length of 18 byte as well and looks as follows:

	Marking	Significance
1	ID (1..255)	Origin ID of the responding Gateway
2	Ctrl-K (0B Hex)	Key character
3	Length or error code	Byte 3 (incl) up to last parameter (incl) or error number, if value > 127
4	ID (0..15)	Cam control ID
5	Command	Command according command table
6	Parameter 1	
	Parameter 2	
	Parameter 3	
	Parameter 4	
	Parameter 5	
	Parameter 6	
	Parameter 7	
	Parameter 8	
	Parameter 9	
	Cam control pos	High byte
16	Cam control pos	Low byte
17	PLS speed	High byte
18	PLS speed	Low byte

In this connection as well as for the master request only those parameters specified with the length byte are valid.
The response record is valid if the bytes 2, 4 and 5 are identical with the corresponding bytes in the request record.
In case of an error during the request at the cam control, the corresponding error will be transferred in byte 3 (instead of the length). The error value always exceeds 127 and can be adopted from the chapter “error recovery”

2.3.6 Protocol frame for DeviceNet

In the DeviceNet the Gateway works as “Group 2 Only Slave”. The access procedures “Polling”, “Bit-Strobe” and “Change of state” are supported, which are described in the following. At present a parameterization of the data is not supported.

2.3.6.1 Polling

When it comes to Polling the DeviceNet-Master transfers an 11-byte-long telegram to the Gateway and receives sequence of 13 byte as an answer. Through the access procedure the DeviceNet-Master is in the position either to read or to write to any parameter of the connected cam controls.
The meaning of the transmitting- and receiving bytes is described in the chapter “Parameterdata”.

2.3.6.2 Bit-Strobe

When it comes to Bit-Strobe access the master transfers the command “Bit-Strobe” without any further data. Consequently he receives 8 byte from the Gateway as an answer, which are described in the chapter “Processdata”

2.3.6.3 Change of state

When it comes to this method the Gateway independently transfers the sequence described in the chapter “Processdata”, as soon as at least one bit of the processdata has been changed.

2.3.6.4 Parameterdata

2.3.6.5 DeviceNet-Master to the Gateway

The following 11 byte are transferred to the Gateway from the DeviceNet-Master:

Byte	Marking	Significance
1	ID (0..15)	Cam control-ID in DICNET
2	Command	Command referring to chapter “Command table“
3	Parameter 1	
4	Parameter 2	
5	Parameter 3	
6	Parameter 4	
7	Parameter 5	
8	Parameter 6	
9	Parameter 7	
10	Parameter 8	
11	Parameter 9	

Following the parameters are described in the chapter of the respective command. In case less than 9 parameters are required, then the remaining parameters are filled up with 0. The Gateway sends the command and the parameters to the cam control, which is selected with byte 1 (ID).

2.3.6.6 Gateway to DeviceNet-Master

The Gateway transfers the following 13 byte to the DeviceNet-Master as an answer to an inquiry (see above):

	Marking	
	ID (0..15)	Cam control-ID in DICNET
2	Command	Command referring to chapter „Command table“
3	Parameter 1	
	Parameter 2	
	Parameter 3	
	Parameter 4	
	Parameter 5	
	Parameter 6	
	Parameter 7	
	Parameter 8	
	Parameter 9	
	Errorcode (High)	See chapter “Error messages“
13	Errorcode (Low)	

Following the parameters are described in the chapter of the respective command. In case less than 9 parameters are sent back, then the remaining parameters are filled up with 0. The Gateway answers with the ID and the command from the inquiry telegram.

2.3.6.7 Error messages

The byte 12 and 13 of the parameter response represents an error code.
In case the error code does not exceed 8000H it is an error message of the DeviceNet-communication according to “DeviceNet Specification Release 2.0”.
In case the error code exceeds 8000H (MSB set) it is a general error message of the Gateway. These errors are described in the appendix in chapter “Errorcodes”.
Though the MSB of the errorcode has to be subtracted (subtract 8000H).

Errorcode = 0:	No error
Errorcode < 8000H:	Error according to DeviceNet specification
Errorcode > 8000H:	Errorcode = Errorcode - 8000H ref. to appendix

The error has to be acknowledged according to picture „Error acknowledgement“ on page 22!

2.3.6.7.1 Example

DeviceNet-Master -> Gateway:

ID	Cmd	Para1	Para2	Para3	Para4	Para4	Para6	Para7	Para8	Para9
0	5	3	5	0	0	0	0	0	0	0

GetTotzeit (Cmd = 5) from program 3 (Para1) and exit 5 (Para2).

Answer from Gateway to DeviceNet-Master:

ID	Cmd	P1	P2	P3	P4	P4	P6	P7	P8	P9	P10	P11
0	5	0	9	0	9	0	0	0	0	0	0	0

Switch on idle time = Switch off idle time = 9ms, Error Code = 0

2.3.6.8 Processdata

The Gateway is always in the position only to transfer the data of one of the maximally 16 connected cam controls to the DeviceNet-Master. The DeviceNet-Master has to inform the Gateway in advance which cam control is to be selected. This is carried out through the parameter data set (Polling) with the command „SET_GATEWAY_ID“. This connection is maintained to the next SET_GATEWAY_ID -command.
Supposing for instance the process data of the cam control with the ID3 are to be transferred, then the DeviceNet-Master has to send the following sequence:

DeviceNet-Master -> Gateway:

ID	Cmd	Para1	Para2	Para3	Para4	Para4	Para6	Para7	Para8	Para9
0	0x1C	3	0	0	0	0	0	0	0	0

In that case the 1. byte (ID = 0) is not evaluated by the Gateway, as this command is processed by the Gateway locally. Therefore in the 1. byte any value is possible.

Answer from Gateway to DeviceNet-Master:

ID	Cmd	P1	P2	P3	P4	P4	P6	P7	P8	P9	P10	P11
0	0x1C	0 or 1	0	0	0	0	0	0	0	0	0	0

In case the Gateway sends back 0 as the 1. parameter, the selected cam control is available in the DICNET. Otherwise there is a 1 in the 1. parameter.

After switching on the Gateway, the cam control is automatically selected with the same ID which is set at the DIP-Switch as MAC-ID. In doing so only the 4 lower bit are evaluated!

2.3.6.9 DeviceNet-Master to the Gateway

No process data are transferred from the Master to the Gateway.

2.3.6.10 Gateway to the DeviceNet-Master

Either the Gateway sends 8 byte as an answer to a Bit-Strobe-Inquiry to the DeviceNet-Master or they are sent independently when the data are changed.
On that occasion a differentiation has to be made between

- processdata without error report
- processdata with error report according to DeviceNet-Specification
- processdata with error report of the Gateways

2.3.6.11 Processdata without error report

	2. byte	3. byte	4. byte	5. byte	6. byte	7. byte	8. byte
Position High	Position Low	Speed High (00..7F)	Speed Low	Output 1-8	Output 9-16	Output 17-24	Output 25-32

2.3.6.12 Processdata with DeviceNet-error report

Byte 1 and 2 contain the DeviceNet-Errorcode, byte 3 = 0x80, byte 4 = 0x09.
The description of the DeviceNet-Errorcode is included in the „DeviceNet-Specification“.

1. byte	2. byte	3. byte	4. byte	5. byte	6. byte	7. byte	8. byte
DeviceNet Error High	DeviceNet Error Low	0x80	0x09	Output 1-8	Output 9-16	Output 17-24	Output 25-32

The error has to be acknowledged according to chapter „Error acknowledgement“ on page 22!

2.3.6.13 Processdata with Gateway-error report

Byte 3 = 0x80, byte 4 = Errorcode.
For the description of the errorcodes see appendix of chapter „Errorcodes“.

	2. byte	3. byte	4. byte	5. byte	6. byte	7. byte	8. byte
Position High	Position Low	0x80	Errorcode	Output 1-8	Output 9-16	Output 17-24	Output 25-32

The error has to be acknowledged according to picture „Error acknowledgement“ on page 22!

2.3.6.13.1 Example

The cam control which was previously selected with SET_GATEWAY_ID sends the following parameters:

ID	Cmd	Para1	Para2	Para3	Para4	Para4	Para6	Para7	Para8	Para9
0	123	0	14	1	2	0	0	0	0	0

Positon: 123, Speed 14, Exit 1 and 10 are switched on.

2.3.6.14 Error acknowledgement

All error reports have to be acknowledged through the parameter channel with the command ’SET_ERROR_QUIT’.

For that reason the DeviceNet-Master sends the following telegram:

ID	Cmd	Para1	Para2	Para3	Para4	Para4	Para6	Para7	Para8	Para9
0x00	0x17	0	0	0	0	0	0	0	0	0

As this telegram is locally evaluated from the Gateway, any ID can be stated in byte 1.

2.3.7 Protocol-frame for Ethernet

The Gateway operates as Ethernet-server and automatically writes back an inquiry to the Ethernet-client.

2.3.7.1 Data from the client to the server (Gateway)

Each inquiry features the following format:

Byte-no.	Designation	Meaning
1	Ctrl-K (0B Hex)	Code character
2	Length	Byte 3 (incl) up to last parameter (incl)
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
...		

Only those parameters are valid which are specified from the Ethernet client to the Gateway with the length byte (valid parameter = length -2).
The Gateway's response record depends on the command.

2.3.7.2 Data from the server (Gateway) to the client

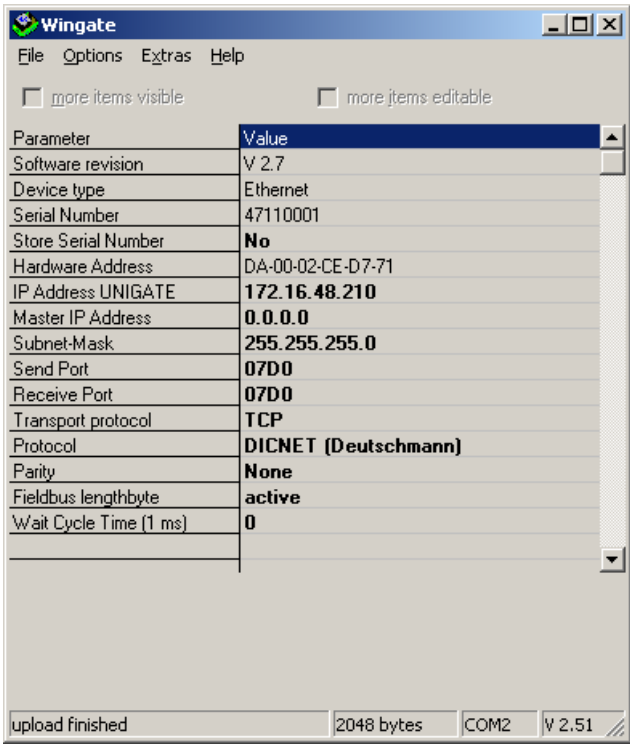
Byte-no.	Designation	Meaning
1	Ctrl-K (0B Hex)	Code character
2	Length or errorcode	Byte 3 (incl) up to last parameter (incl) or errornumber, if value > 127
3	ID (0..15)	Cam control ID
4	Command	Command according to command table
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
...		

Here as well same as for the inquiry only those parameters are valid that are specified with the length byte.
The response record is valid provided that the bytes 1, 3 and 4 are identical to the corresponding bytes in the inquiry record.
In case of an error during the inquiry at the cam control, the corresponding error will be transferred
in byte 2 (instead of the length). The error value always exceeds 127 and can be adopted from the chapter "error recovery".

2.3.7.3 Example with the DEUTSCHMANN Ethernet starterkit

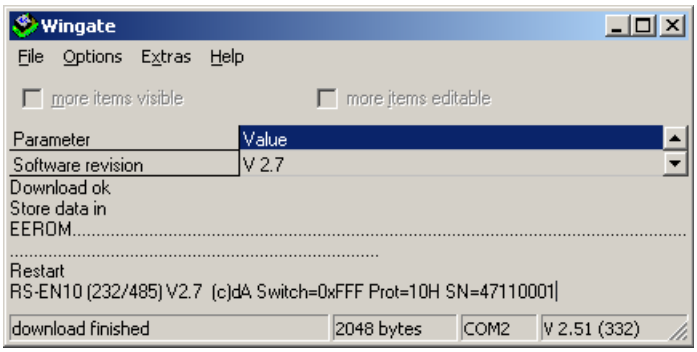
Configuration of the UNIGATE Ethernet via WINGATE
Setzen Sie das UNIGATE in den Konfigmode:
S4 + S5 = "FF"
Interfaceschalter auf 232

Start the device

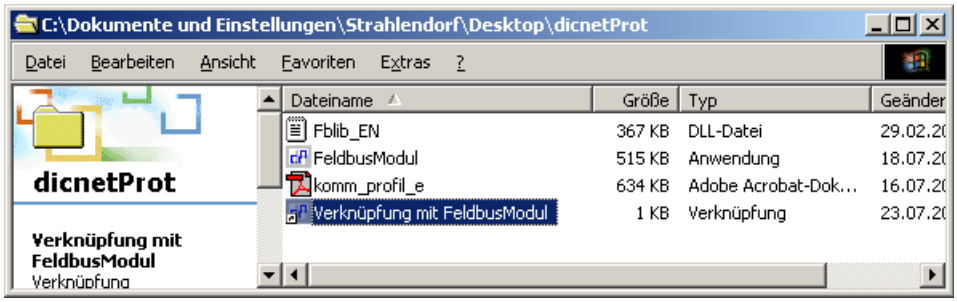


Configuration of UNIGATE Ethernet via WINGATE

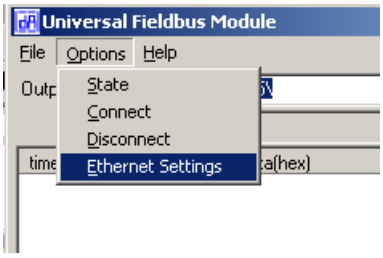
File - Download



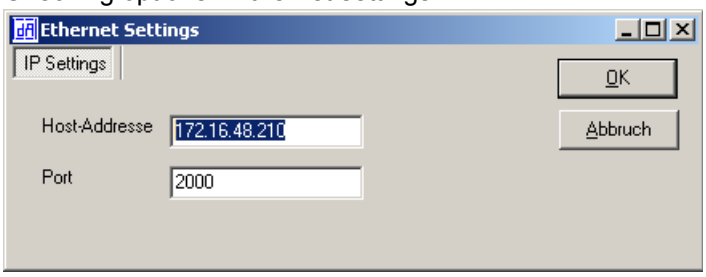
Set S4 + S5 to "00"
Interface switch to 485
Connect DICNET
Start UNIGATE (24V)



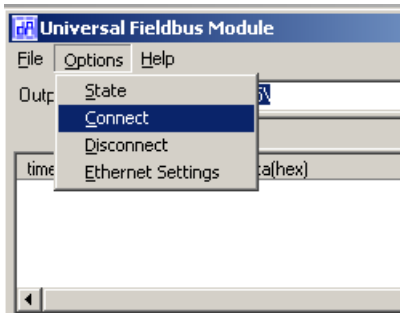
Start "Connection with Fieldbusmodule"



Checking options - Ethernet settings



The host address "172.16.48.210" has to be the same as the one in WINGATE.
For this ask your system administrator for a free IP address.



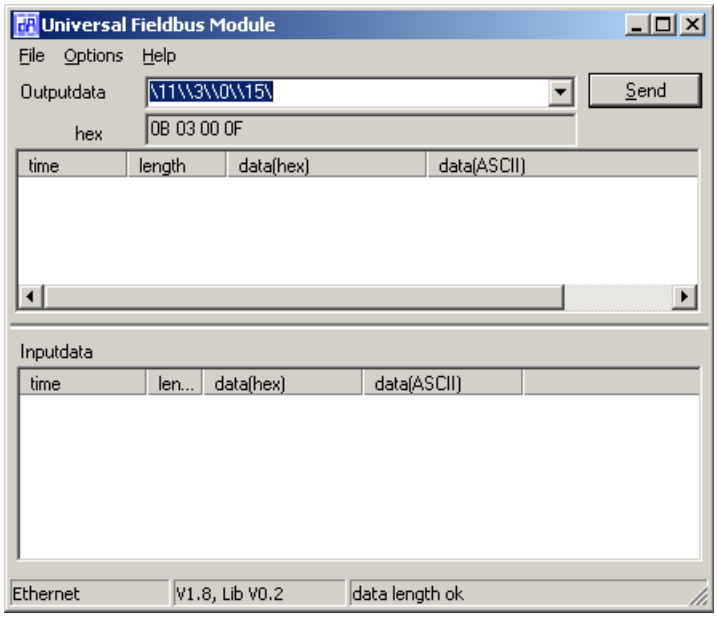
Establishing Client - Server - Connection with Options - Connect

BIn case of a faulty connection after approximately 20 seconds the following message comes up:

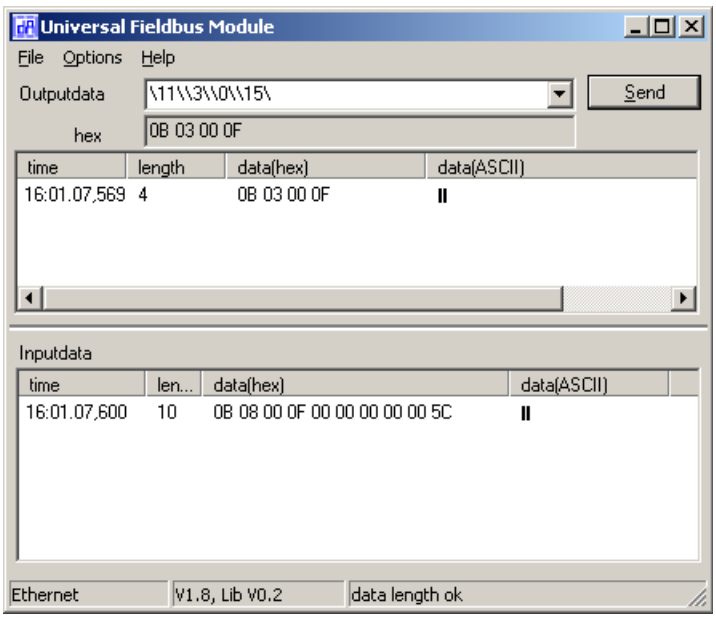
The BUS-LED flashes red - green in turns

In case the connection is correct then the BUS LED changes to statically green.

Now please enter a command into the input line:
\11\\3\\0\\15\

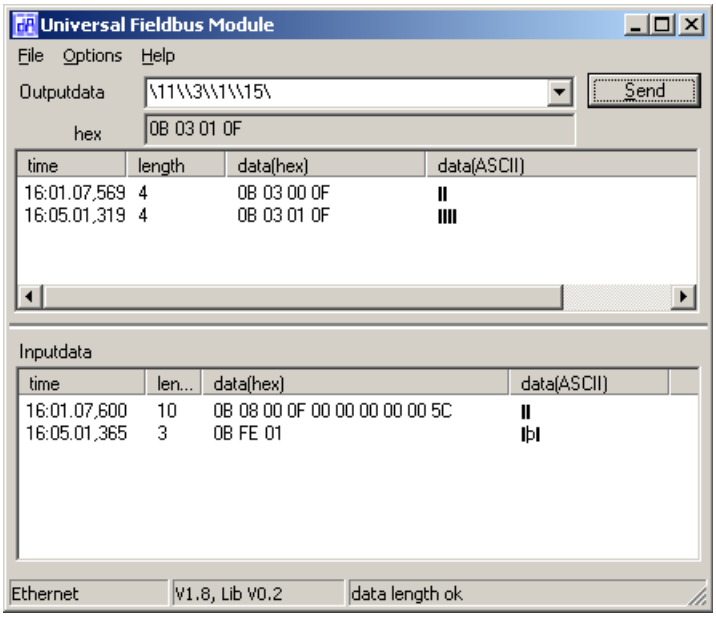


and click on "Send"



In the input window you will now get back a correct answer for the command "**GET_DISPLAY**" From the DICNET participant with the ID 0 (3. byte in the send and receive string).

Following please find an example for a participant with ID 1, which does not exist:



2.3.8 Command table

Command's name	Command's value	Meaning - see chapter
GET_OUTPUT	0x01	2.5.1.1
GET_NEXT_CAM	0x03	2.5.1.3
GET_BACK_CAM	0x04	2.5.1.4
GET_IDLETIME	0x05	2.5.1.5
GET_POSITION	0x08	2.5.1.6
GET_SPEED	0x09	2.5.1.7
GET_STATUS	0x0A	2.5.1.8
GET_OUT_POS	0x0E	2.5.1.9
GET_DISPLAY	0x0F	2.5.1.10
GET_LOGIC	0x41	2.5.1.11
GET_DATA_EXIST	0x43	2.5.1.12
GET_GATEWAY_ID	0x44	2.5.1.13
GET_PARAMETER	0x45	2.5.1.14
GET_OUTPUT_NAME	0x46	2.5.1.15
GET_GATEWAY_DATA	0x47	2.5.1.16
GET_EEROM_BLOCK	0x48	2.5.1.17
GET_L2000-DATA	0x49	2.5.1.18
GET-INPUT	0x4A	2.5.1.2
SET_CAM_NEW	0x10	2.5.2.1
SET_IDLETIME	0x12	2.5.2.2
SET_ERROR_QUIT	0x17	2.5.2.3
SET_LOGIC	0x18	2.5.2.4
SET_CAM_MOVE	0x1A	2.5.2.5
SET_CAM_CHANGE_SHORT	0x1B	2.5.2.6
SET_GATEWAY_ID	0x1C	2.5.2.7
SET_CAM_CHANGE_MT	0x20	2.5.2.8
SET_PARAMETER	0x21	2.5.2.9
SET_OUTPUT_NAME	0x22	2.5.2.10
SET_EEROM_BLOCK	0x23	2.5.2.11

Further commands which are not supported by our current products any more are described in the appendix.

2.3.9 Parameter table

This Parameter table is used by the commands GET_PARAMETER and SET_PARAMETER (see „Command descriptions“ on page 31

Command's name	Command's value	Meaning	Explanation
PNR_SOFT_REV	0x0001	s. PNR_HARD_REV	
PNR_HARD_REV	0x0002	ASCII z. B: '3" 1"2' 't' = V3.12t - gives back the soft-, or hardware version	
PNR_UNIT_NAME	0x0003	ASCII i. e. 'L' '4' '8 " ' = L48	
PNR_UNIT_TYP	0x0004	Device type	
PNR_VNUMBER	0x0005	Article number	
PNR_SN	0x0006	Serial number	
PNR_OPTION X	0x0007	Option X	
PNR_ENCODER_TYP	0x0010	Encoder type	chapter 2.3.9.1
PNR_RESOLUTION_PER_TURN	0x0011	Real-resolution per revolution	chapter 2.3.9.2
PNR_NUMBER_OF_TURNS	0x0012	Real-number revolution	
PNR_SCALED_ENCODER_RES	0x0013	Virtual encoder value	
PNR_ENCODER_INVERT	0x0014	Reversal of rotational direction	chapter 2.3.9.3
PNR_SCALED_COUNT_RANGE	0x0017	Virtual Count range	
PNR_COUNT_RANGE	0x0018	Counting area at incremental encoders	
PNR_COUNT_RESTORE_VALUE	0x0019	At X 16: = brake point	
PNR_TIMEBASE	0x001C	Time basis at Timer	
PNR_DEADTIME_BASE_US	0x001D	Time unit for idle time compensation in µs (if not defined -> 1000µs)	
PNR_NUMBER_OUTPUTS	0x0020	Number of outputs	
PNR_NUMBER_LOCK_OUTPUTS	0x0021	Number of locked outputs	
PNR_NUMBER_DATA_RECORDS	0x0022	Number of data records	
PNR_NUMBER_LOGIC_INPUTS	0x0023	Number of Logic inputs	
PNR_NUMBER_ANGLE_TIME	0x0024	Number of angle/time outputs from output 1	
PNR_NUMBER_OUTNAME_CHAR	0x0025	Output names	
PNR_NUMBER_PROGRAMS	0x0026	Number of programs	
PNR_NUMBER_AXIS	0x0027	Number of axes	
PNR_NUMBER_ANALOGOUTPUT	0x0028	Number of analog outputs	
PNR_NUMBER_COUNTERCAM	0x0029	Number of counter cams	
PNR_FIRST_OUTPUT_NR	0x002A	Counting starts at 1	
PNR_SPEED_SCALE	0x0030	With reference to rev./ms =>60000 = rev./min 0...9999000 (rev./ms)	
PNR_LANGUAGE	0x0031	Language	chapter 2.3.9.4
PNR_DEADTIME_TYP	0x0032	ITC-type	chapter 2.3.9.5
PNR_ZEROPOINT_OFFSET	0x0033	Preset value at inc. / abs.: virtual value	
PNR_ACTIV_PROGNR	0x0034	Active program	0..max program -1
PNR_ACTIV_AXIS	0x0035	Active axis	1..max AxisNo.
PNR_CALC_SPEED_START	0x0036	IdleStart scaled	
PNR_CALC_SPEED_STOP	0x0037	IdleStop scaled	
PNR_DICNET_ID	0x0038	Actual value (PLS = 80..95), RS232 = 232	
PNR_CLEAR_LENGTH	0x0039	Length clear pulse	
PNR_BREAK_PARA	0x003A	(BrakeA*0x10000) + BrakeB	
PNR_OUTPUT_OFF_SPEED	0x003B	Speed-threshold value below which the outputs are switched off	
PNR_WZ_MAXTIME	0x003C	Time in ms	
PNR_WZ_TIMEBASE	0x003D	Time in µs	
PNR_V_LIMIT	0x003E	M13 = 1, if V_LIMIT is exceeded	
PNR_DREHSCHALTER	0x003F	Read switch position	
PNR_RESTART	0x004E	Warmstart with value 0x1234	
PNR_CLEAR_EEROM	0x004F	General deletion: 1: 0x1234 -> 2:~0x1234	
PNR_STATUS_FLAGS	0x0050		
PNR_PROC_OUT_MAPPING	0x0051	Mapping of the process data in the Fieldbus	
PNR_PROC_IN_MAPPING	0x0052	Mapping of the process data in the Fieldbus	
PNR_USED_EEROM_LEN	0x0053	Actual used EEROM length	
PNR_S7_MODE	0x0054	1 = S7 do not copy data into the EEROM; 0xFF = no PB	
PNR_RESET_EEROM	0x0055	Set to set in factory 1:0x1234 -> 2:~0x1234	
PNR_CYCLETIME	0x0056	Read cycle time	
PNR_AKTIV_STATUS	0x0057	See chapter 2.3.11	
PNR_PROC_LOAD	0x0058	Processor utilization	
PNR_ENABLE_OPTION	0x0059	Release of options	
PNR_TEACH_IN_ZEROPOINT	0x0060	Teach-in zero offset	
PNR_ENABLE_TESTMODE	0x005B	With 0x1234 -> Switch to testmode	
PNR_DATA_NOT_IN_EEROM	0x005C	Cams, dead times are only stored in the RAM (volatile)	
PNR_SCALED_NR_OF_TURNS	0x0015	Virtual number of revolutions	
PNR_ZEROPOINT_OFFSET_REAL	0x005D	Real offset-value	
PNR_DYN_ZEROPOINT_OFFSET	0x005E	Dynamic offset (only IO8)	
PNR_INTERFACE	0x005F	0=RS232; 1=DICNET w/o bus termination; 3=DICNET with bus termination	
PNR_ERROR_QUIT	0x0060	Error quit through Modbus 0 -> 1 (only LOCON 100-MB)	

2.3.9.1 PNR_ENCODER_TYP - encoder type

1 = Absolute encoder Parallel Gray
2 = Incremental encoder
3 = Absolute encoder SSI Gray
5 = Timer
6 = Multiturn-SSI
7 = Incremental 24 bit

2.3.9.2 PNR_RESOLUTION_PER_TURN

Absolute Parallel Gray: 360, 512, 720, 1000, 1024, 2048, 3600, 4096
SSI Gray: 360, 1024, 4096
Incremental: 1024 (16), 4096 (17)

2.3.9.3 PNR_ENCODER_INVERT

0 = Regular
1 = Inverted

2.3.9.4 PNR_LANGUAGE - language selection

0	=	German	5	=	Flemish
1	=	English	6	=	Dutch
2	=	French	7	=	Swedish
3	=	Italian	8	=	Finnish
4	=	Spanish	9	=	Danish

2.3.9.5 PNR_DEADTIME_TYP

Path-dependent ITC (standard at Deutschmann cam controls)
0 = None
1 = Blockwise
2 = Bitwise
3 = Blockwise, separate switch-on and switch-off idle times
4 = Bitwise, separate switch-on and switch-off idle times

Time-dependent ITC (only ROTARNOCK 100, LOCON 100)
Value = Idle time type from above table + 10
-> Time-dependent bitwise idle time = 2+10 = 12

Direct ITC (only ROTARNOCK 100, LOCON 100)
Value = Idle time type from above table + 20
-> Direct bitwise ON/OFF idle time = 24

2.3.10 Bitnumbers of the parameter PNR_STATUS_FLAGS

Command's number	Command's value	Meaning
BITNR_ENCODER_INVERT	0x00	Reversal of rotational direction
BITNR_SOFT_RUNCONTROL	0x01	Floating fault signal switching contact
BITNR_UPDOWNLOAD_ENABLE	0x02	Release up-download
BITNR_UPDATE_DEP_TURN	0x03	Direction cam possible
BITNR_ENCODER_TEST	0x04	Encoder monitoring
BITNR_SPEED_OUT_HARD	0x05	Speed output 0..255 at maximal 8 outputs
BITNR_ANALOGOUT_EXIST	0x06	Analog output available
BITNR_ENCODER_SCALABLE	0x07	Encoder resolution can be adjusted
BITNR_BINAER_UP_DOWNLOAD	0x08	Identification of the data transfer method

2.3.11 Bitnumbers of the parameter PNR_ACTIVE_STATUS

MASK_STATUS_BITW_TZK	0x01	Type of idle time compensation
MASK_STATUS_BLOCKW_TZK	0x02	
MASK_STATUS_BITW_EA_TZK	0x04	
MASK_STATUS_BLOCKW_EA_TZK	0x08	
MASK_STATUS_RICHT_NOCK	0x10	Direction cams active
MASK_STATUS_WZ	0x20	Angle-time cams active
MASK_STATUS_LOGIK	0x40	Logic active

2.4 Error handling

The below listed error messages might be generated during the communication:

Error number	Meaning / error handling
249	Error in accessing the internal extension card
250	Output buffer overflow
251	Last command not complete yet
252	Unknown command
253	Checksum error or length error
254	Any other error during the communication (i. e. participant not online)

Instead of the network ID, the error code is transferred as 3rd byte in the response record provided that the net-ID is available. In case no error occurs, this byte contains a value between 0..128.
At Interbus-S the error code is transferred in the 1st byte instead of the command.
In all other cases the error number is transferred to the field “Error number“.

2.5 Command descriptions

2.5.1 Parameter received from PLS

2.5.1.1 GET_OUTPUT

Parameters to the PLS: 1. „Offset output blocks“

Parameters from the PLS: 1. Output assignment 1..8 („Offset output block“ * 8)
2. Output assignment 9..16 + ...

Note: The PLS always transfers the states of all outputs. If, for instance, the device has 32 outputs, 4 bytes are transferred.

If „Offset output blocks“ is not transferred, this equals an Offset of 0.

2.5.1.2 GET_INPUT

Parameters to the PLS: 1. „Offset input blocks“

Parameters from the PLS: 1. Input assignment 1..8 („Offset input block“ * 8)
2. Input assignment 9..16 + ...

Note: The PLS always transfers the states of all inputs. If, for instance, the device has 32 inputs, 4 bytes are transferred.

If „Offset input blocks“ is not transferred, this equals an Offset of 0.

2.5.1.3 GET_NEXT_CAM

Parameters to the PLS:
1. Program No. (0..number of program -1)
2. Output No. (1..number of outputs) MSB = 1 -> analog output
3. Last switch-on point (High byte)
4. Last switch-on point (Low byte)
5. Last switch-on point (bits 16..23) **only in the case of MT**
6. Flag ERSTE_NOCKE (1 = first cam, otherwise 0) **only MT**

Parameters from the PLS:
1. Switch-on point (High byte)
2. Switch-on point (Low byte)
3. Switch-off point (High byte)
4. Switch-off point (Low byte)
5. Cam number (High byte)
6. Cam number (Low byte)
7. Switch-on point (bits 16..23)**only in the case of MT**
8. Switch-off point (bits 16..23) **only in the case of MT**

Note: If the first cam is to be read, the value ERSTE_NOCKE is transferred as the "Last switch-on point".

If no cam is present with a higher switch-on point than "Last switch-on point", the PLS returns LEERE_NOCKE as the new switch-on point.

Parameter "Cam number" is required if this read cam is to be changed.

If scaled and real encoder resolution differ, the scaled switch-on and switch-off points are transferred.

Switch-off point > 0x8000 -> this equals the switch-on time at angle-time -0x8000! (Time basis is to be considered!)

2.5.1.4 GET_BACK_CAM

Parameters to the PLS:
1. Program No. (0..number of program -1)
2. Output No. (1..number of outputs) MSB = 1 -> analog
3. Last switch-on point (High byte)
4. Last switch-on point (Low byte)
5. Last switch-on point (bits 16..23) **only in the case of MT**
6. Flag ERSTE_NOCKE (1 = first cam, otherwise 0) **only MT**

Parameters from the PLS:
1. Switch-on point (High byte)
2. Switch-on point (Low byte)
3. Switch-off point (High byte)
4. Switch-off point (Low byte)
5. Cam number (High byte)
6. Cam number (Low byte)
7. Switch-on point (bits 16..23) **only in the case of MT**
8. Switch-off point (bits 16..23) **only in the case of MT**

Note: See GET_NEXT_CAM

2.5.1.5 GET_IDLETIME

Parameters to the PLS:
1. Program No. (0..number of program -1)
2. Output No. (1..number of outputs)

Parameters from the PLS:
1. Switch-on idle time (High byte)
2. Switch-on idle time (Low byte)
3. Switch-off idle time (High byte)
4. Switch-off idle time (Low byte)
5. Output update (0 = always, 1 = pos., 2 = neg. rot.)
6. Cam count (0..255)

Note: The switch-off idle time is transferred only if the PLS features separate switch-on/ switch-off idle time.

Parameter 5 is transferred only if the PLS features direction-dependent outputs (see GET_CONFIG_PARA). In this case, a switch-off idle time is always transferred. The unit of the idle times can be read out through the parameter PNR_DEADTIME_US.

Parameter 6 is only transferred at active counter cams and it contains the number of cycles in which the output is disabled.

2.5.1.6 GET_POSITION

Parameters to the PLS: None

Parameters from the PLS:
1. Current encoder position (High)
2. Current encoder position (Low)
3. Current encoder position (bits 16..23) **only MT**

Note: If the scaled and real encoder resolution differ, the scaled encoder position is transferred.

2.5.1.7 GET_SPEED

Parameters to the PLS: None

Parameters from the PLS:
1. Current encoder speed (High)
2. Current encoder speed (Low)
3. Inc/10 ms (High)
4. Inc/10 ms (Low)

Note: An average value is generated from the parameters 1 and 2 within 2 seconds and from the parameters 3 + 4 within 10 ms.

2.5.1.8 GET_STATUS

Parameters to the PLS: None

Parameters from the PLS:
1. Current error number (0 = ok)
2. Current program being run
3. Active axis
4. Status byte
   Bit 0 = Program Enable
   Bit 1 = Password prompt enabled
   Bit 2-7 = undefined

2.5.1.9 GET_OUT_POS

Parameters to the PLS:
1. Scaled encoder position (High)
2. Scaled encoder position (Low)
3. Scaled encoder position (bits 16..23) (only MT)
4. Output block offset (if available) see GET_OUTPUT

Parameters from the PLS: 1. Output assignment 1..8 + (Output block-offset * 8)
2. Output assignment 9..16 + ...

Note: The PLS always transfers the states of all outputs. If, for instance, the devices feature 32 outputs, 4 bytes are transferred.

This command differs from GET_OUTPUT only by virtue of the fact that the output assignment of any position can be scanned and not only the current encoder position.

The idle time compensation and logic are not taken into consideration.

2.5.1.10 GET_DISPLAY

Parameters to the PLS: None

Parameters from the PLS:
1. Current error number (0 = ok)
2. Current program being run
3. Current encoder speed (High)
4. Current encoder speed (Low)
5. Current scaled encoder position (High)
6. Current scaled encoder position (Low)
7. Current scaled encoder position (bits 16..23) **only MT**

2.5.1.11 GET_LOGIC

Parameters to the PLS:
1. Program No. (0.. number of programs -1)
2. Output No. (1.. number of outputs)
3. Output type (0 = output, 1 = flag)

Parameters from the PLS:

1. Output	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	|			+	+	+	+	+	Output No.. (1-31)
	|	+	-	-	-	-	-	-	0 = Output, 1 = Flag
	+	-	-	-	-	-	-	-	0 = Standard 1 = Inverted
2. Input 0	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	|	|		+	+	+	+	+	Input No. 0-31
	+	+	-	-	-	-	-	-	Edge trigger (see below)
1.Output1	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
3. Input 1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	|	|	|	+	+	+	+	+	Input No.. (0-31)
	+	+	+	-	-	-	-	-	Logic operation (see below)
4. Input 2									**(as for input 1)**
5. Input 3									**(as for input 1)**
6. Input type	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	|	|	|	|	|	|	+	+	Input 0 (see below)
	|	|	|	|	+	+	-	-	Input 1 (see below)
	|	|	+	+	-	-	-	-	Input 2 (see below)
	+	+	-	-	-	-	-	-	Input 3 (see below)
7.Switch-off time									

Note:

Edge trigger:	00	=	Leading edge
	01	=	Trailing edge
	10	=	Reserved
	11	=	Reserved
Logic operations:	000	=	None
	001	=	Or
	010	=	And
	011	=	Nor
	100	=	Nand
	rest	=	Reserve
Input type:	00	=	Cam output
	01	=	Hardware input
	10	=	Flag
	11	=	Shift register

2.5.1.12 GET_DATA_EXIST

Parameters to the PLS:
1. Program No. (0..number of program -1)
2. Output No. (1..number of outputs or 0 x FF)

Parameters from the PLS:
1. Cam present, if > 0
2. Idle times present, if > 0

Note: A check is conducted in order to establish whether data records with cams or idle times are present in the selected program and output (if output = 0xFF), the entire program is checked.

2.5.1.13 GET_GATEWAY_ID

Parameters to the PLS: None

Parameters from the PLS: ID of the selected PLS (0..15)

Note: The ID indicates to which cam control the Gateway is to be logically linked. All following commands are then exchanged with this ID.

This command is required in connection with Fieldbus Gateways only.

2.5.1.14 GET_PARAMETER

Parameters to the PLS:
1. Parameter number (High byte)
2. Parameter number (Low byte)
3. Parameter type (0 = current value, 1 = min. value, 2 = max. value)

Parameter from the PLS:
1. Parameter number (High byte)
2. Parameter number (Low byte)
3. Status (see below)
4. Value (31..24) MSB
5. Value (23..16)
6. Value (15..8)
7. Value (7..0) LSB

Note:

Status:
00(Hex) = Current value Read-Write
01(Hex) = Current value Read-Only
10(Hex) = Min. value Read-Write
11(Hex) = Min. value Read-Only
20(Hex) = Max. value Read-Write
21(Hex) = Max. value Read-Only
FF(Hex) = Value does not exist

Note: With this command it is possible to select all current parameters of the PLS as well as the minimum and maximum value, which can be adopted by these parameters (see „Parameter table“ on page 29) provided that they are existent.

2.5.1.15 GET_OUTPUT_NAME

Parameter to the PLS:
1. Output No. (1..number of outputs)
2. Pointer on output name (Ptr[0..nameLen-1])

Parameter from the PLS:
1. Output name [Ptr] (see above)
2. Output name [Ptr+1]
3. Output name [Ptr+2]
4. Output name [Ptr+3]
5. Output name [Ptr+4]
6. Output name [Ptr+5]
7. Output name [Ptr+6]
8. Output name [Ptr+7]

2.5.1.16 GET_GATEWAY_DATA

Parameter to the PLS:
1. Output block-offset (see GET_OUTPUT)

Parameter from the PLS:
1. Speed MSB current encoder speed
2. Speed LSB
3. Pos \ current scaled encoder position
4. Pos | 24 bit
5. Pos LSB /
6. Out 1 - 8 + ((output block-offset) * 8)
7. Out 9 - 16 + ...
8. Out 17 - 23 + ...
9. Out 24 - 32 + ((output block-offset) * 8)

2.5.1.17 GET_EEROM_BLOCK

Parameter to the PLS:
1. Para: BlockNo (0-255)
2. Para: BlockLen (1..249) - recommended: 128

Parameter from the PLS:
1. Para: EEROM [BlockNr * BlockLen]
2. Para: EEROM [(BlockNr * BlockLen) + 1]
:
BlockLen.Para: EEROM [.. + (BlockLen - 1)]

2.5.1.18 GET_L2000_DATA

Parameter to the PLS:

```
1. State of PLS ------------  X  X  X  X  X  X  X  X
                                                |  |
                                                |  +----> Prog Enable
                                                +--------> Output Enable
```

Parameter from the PLS:

```
1. Error number
2. State of output 1...8
3. Error value
4. Status --------------------  X  X  X  X  X  X  X  X
5. Active program                                   |
                                                    + ----> Not used
```

2.5.2 Send parameters to PLS

2.5.2.1 SET_CAM_NEW

Parameters to the PLS:

1. Program No. (0..number of programs - 1)
2. Output No. (1..number of outputs) MSB = 1 -> analog output!
3. Switch-on point (High byte)
4. Switch-on point (Low byte)
5. Switch-off point (High byte)
6. Switch-off point (Low byte)
7. Switch-on point (bits 16..23) **only MT**
8. Switch-off point (bits 16..23) **only MT**

Parameters from the PLS:

1. Acknowledgement (see Note)

Note: For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS during programming, this can be queried by GET_STATUS.

If the real and the scaled encoder resolution differ, the scaled switch-on and switch-off values are transferred.

Switch-off point > 0x8000 -> this equals the switch-on time at angle-time -0x8000! (Time basis is to be considered!)

2.5.2.2 SET_IDLETIME

Parameters to the PLS:

1. Program No. (0..number of programs - 1)
2. Output No. (1..number of outputs)
3. Switch-on idle time (High byte)
4. Switch-on idle time (Low byte)
5. Switch-off point idle time (High byte)
6. Switch-off idle time (Low byte)
7. Output update (0 = always, 1 = pos., 2 = neg. rotation)
8. Cam count (0..255)

Parameters from the PLS:

1. Acknowledgement (0 = ok, otherwise error number)

Note: The switch-off idle time is required for PLS with separate switch-on and switch-off idle times only, otherwise this parameter can be dropped without being replaced. If in such a device the switch-on and switch-off times equal 0, then the data record will be deleted.

The time unit is set by the cam control (1 ms/0.1 ms). See parameter PNR_DEADTIME_BASE_US.

Parameter 7 is only required, if it is a PLS with outputs that depend on the rotational direction. In this case also the idle time compensation has always to be transferred.

2.5.2.3 SET_ERROR_QUIT

Parameters to the PLS: None

Parameters from the PLS: 1. Acknowledgement (0 = ok, otherwise error number)

This command is not required while connecting via RS232 or DICNET, since an error is acknowledged with a single CR (0DH).

2.5.2.4 SET_LOGIC

Parameters to the PLS:

1. Program No.									**(0..max)**
2. Output	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I			+	+	+	+	+	Output No.. (1-16)
	I	+	-	-	-	-	-	-	0 = Output, 1 = Flag
	+	-	-	-	-	-	-	-	0 = Standard, 1 = Inverted
3. Input 0	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I	I		+	+	+	+	+	Input No. 0-31
	+	+	-	-	-	-	-	-	Edge trigger (see below)
4. Input 1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	+	+	+	+	+	Input No.. (0-31)
	+	+	+	-	-	-	-	-	Logic operation (see below)
5. Input 2									**(as for input 1)**
6. Input 3									**(as for input 1)**
7. Input type	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	I	I	I	+	+	Input 0 (see below)
	I	I	I	I	+	+	-	-	Input 1 (see below)
	I	I	+	+	-	-	-	-	Input 2 (see below)
	+	+	-	-	-	-	-	-	Input 3 (see below)
8. Switch-off time									

Parameters from the PLS: 1. Acknowledgement (see Note)

Note: For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS during programming, this can be queried by GET_STATUS.

If the logic record is to be deleted, the value 0 for the input as well as for 0 x FF for the switch-off time has to be transferred.

Edge trigger	00	=	Leading edge
	01	=	Trailing edge
	10	=	Reserved
	11	=	Reserved
Logic operations:	000	=	None
	001	=	Or
	010	=	And
	011	=	Nor
	100	=	Nand
	rest	=	Reserve

Input type:	00	=	Cam output
	01	=	Hardware input
	10	=	Flag
	11	=	Shift register

2.5.2.5 SET_CAM_MOVE

Parameters to the PLS:
1. Program No. (0..number of programs - 1)
2. Output No. (1..number of outputs) MSB = 1 => analog output
3. Number of increments (High byte)
4. Number of increments (Low byte)
5. Number of increments (bits 16..23) **only MT**

Parameters from the PLS:
1. Acknowledgement (see Note)

Note: Conditional on the programming the acknowledgement of the PLS is always 0.

2.5.2.6 SET_CAM_CHANGE_SHORT

Parameters to the PLS:
1. Switch-on point (High byte)
2. Switch-on point (Low byte)
3. Switch-off point (High byte)
4. Switch-off point (Low byte)
5. Cam number (High byte)
6. Cam number (Low byte)

Parameters from the PLS:
1. Acknowledgement (see Note)

Note: The cam number must be identical to the number transferred by the PLS with request GET_NEXT_NOCKE or GET_BACK_NOCKE. This cam number also defines the program and the output.

DELETE: On = Off! (At angle-time: On = Off ≥ 0 x 8000)

For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS during programming, this can be queried by GET_STATUS.
If the real encoder resolution differs from the scaled encoder resolution, the scaled switch-on and switch-off values are transferred.

2.5.2.7 SET_GATEWAY_ID

Parameters to the PLS:
1. New ID (0..15)

Parameters from the PLS:
1. Acknowledgement (0 = ok, 1 = ID not in the network)

Note: The ID defines to which cam controls the Gateway is to be connected logically. All following commands are subsequently exchanged with this ID.

For LOCON 2000: Allocation of ID for internal communication

2.5.2.8 SET_CAM_CHANGE_MT (only MT)

Parameters to the PLS:
1. Switch-on point (High byte)
2. Switch-on point (Middle byte)
3. Switch-on point (Low byte)
4. Switch-off point (High byte)
5. Switch-off point (Middle byte)
6. Switch-off point (Low byte)
7. Cam number (High byte)
8. Cam number (Low byte)

Parameters from the PLS:
1. Acknowledgement (see Note)

Note: The cam number must be identical to the number transferred by the PLS with request GET_NEXT_NOCKE or GET_BACK_NOCKE.

The switch-on and switch-off points transferred also relate to the program and the output of the data record with the cam number transferred.

For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS, this can be queried by GET_STATUS.

If the real encoder resolution differs from the scaled encoder resolution, the scaled switch-on and switch-off values are transferred.

2.5.2.9 SET_PARAMETER

Parameters to the PLS:
1. Parameter No. (High byte)
2. Parameter No. (Low byte)
3. Value (31..24) MSB
4. Value (23..16)
5. Value (15..8)
6. Value (7..0) LSB

Parameters from the PLS:
1. Parameter No.(High byte)
2. Parameter No. (Low byte)
3. Status (see below)

Note:

Status:

00(Hex)	=	Ok, Value changed
01(Hex)	=	Value Read-Only, not changed
02(Hex)	=	No software release
03(Hex)	=	EEROM Write Error
04(Hex)	=	Range error
05(Hex)	=	PLS not ready for memory
06(Hex)	=	Repeat command
07(Hex)	=	External module error
FF(Hex)	=	Value does not exist

Note: With this command all current parameters can be changed (see „Parameter table“ on page 29).

2.5.2.10 SET_OUTPUT_NAME

Parameters to the PLS:
1. Output No. (1.. number of outputs)
2. Pointer on output name (Ptr[0..nameLen-1])
3. Output name [Ptr] (see above)
4. Output name [Ptr+1]
5. Output name [Ptr+2]
6. Output name [Ptr+3]
7. Output name [Ptr+4]
8. Output name [Ptr+5]

Parameters from the PLS:
1. Acknowledgement (0 = ok, otherwise error message)

2.5.2.11 SET_EEROM_BLOCK

Parameters to the PLS:
1. Parameter: Block_No. (0..255)
2. Parameter: Block Len (1..249) -> recommended: 128
3. Parameter:
   0 = Store in RAM:
   1 = COPY RAM -> EEROM (see below)
   2 = As 1, but after all actions - auto restart
   3 = General deletion - state of delivery (see below)
4. Parameter: EEROM [BlockNo * BlockLen]
5. Parameter: EEROM [(BlockNo * BlockLen)+1]

   :

BlockLen + 3. Para: EEROM[(BlockNo * Blocklen) + (BlockLen -1)]

Parameters from the PLS:
1. Parameter: Acknowledgement (0 = ok, otherwise error)

Note: When copying from RAM into EEROM (3. parameter = 1) always the complete address area 0 .. ((BlockNo + 1) * BlockLen) -1 is transferred. The remaining EEROM is replenished with 0 x FF.

For opening a general deletion (reset in the state of delivery) the 1. parameter has to contain = 0 x 55, the 2. parameter = 0 x AA and the 3. parameter = 3.

3 Examples

3.1 Transfer with DICNET or RS232

3.1.1 Reading the initial state of a LOCON 32

Transmission to cam control:

0BH	Constant code characters
02H	Length (network ID + command)
30H	Network ID (always 0 with RS232)
01H	Command for GET_OUTPUT
33H	Checksum (length XOR network ID XOR command)

Response from cam control:

0BH	Constant code characters
06H	Length (network ID + command) + parameters
50H	Network ID (no significance)
01H	Command for GET_OUTPUT (echo)
12H	Output 1..8
23H	Output 9..16
34H	Output 17..24
45H	Output 18..32
C7H	Checksum (length XOR ... XOR output 18..32)

3.1.2 Transfer with InterBus-S

3.1.2.1 Scanning the cam control type

Transmission to cam control:

86H	Command number with MSB set
00H	
00H	No parameters ==> e. g.: pad with 00
00H	
00H	
00H	
00H	
00H	

Response from cam control:

86H	Retry of command code with MSB set
02H	Type number (2 = LOCON 2)
33H	Software Version V3.
30H	0
31H	1 = V3.01
00H	Remainder padded with 00H
00H	
00H	

3.2 Sample-source-code for accessing the communication routine

In the following we show the source-code for accessing the communication routines PLS by means of the sample file 8051.

```
 #include <stdio.h>   /* define I/O functions */

// Defines
#define TRUE                    1
#define FALSE                   0

#define TRANSMITTRUE
#define RECEIVE                 FALSE

#define CTRL_K                  0x0B

#define MAX_PARA_LENGTH10            /* Max. length of para.-block incl. Cmd */

#define WAIT_NSW                0xFE /* Dummy-Commands for GetNSWPara */
#define TIMEOUT_NSW             0xFF

#define GET_OUTPUT              0x01
#define GET_NEXT_NOCKE          0x03
#define GET_BACK_NOCKE          0x04
#define GET_TOTZEIT             0x05
#define GET_TYP                 0x06
#define GET_MAXPARA             0x07
#define GET_POSITION            0x08
#define GET_GESCHW              0x09
#define GET_STATUS              0x0A /* Command-identification for serial Comm. */
#define GET_KONFIG_PARA         0x0B
#define GET_SYSTEM_PARA         0x0C
#define GET_AUSG_NAME           0x0D
#define GET_AUSG_POS            0x0E
#define GET_ANZEIGE             0x0F
#define GET_OUTPUT_MATTE        0x31
#define GET_LOGIC               0x41
#define GET_GEAR_PARA           0x42
#define GET_DATA_EXIST          0x43
#define GET_GATEWAY_ID          0x44
#define GET_PARAMETER           0x45 /* General parameter according to PNR_xxx */
#define GET_OUTPUT_NAME         0x46
#define GET_GATEWAY_DATA        0x47
#define GET_EEROM_BLOCK         0x48
#define GET_L2000_DATA          0x49

#define SET_NOCKE_NEU           0x10
#define SET_NOCKE_AENDERN       0x11
#define SET_TOTZEIT             0x12
#define SET_AKT_PARA            0x13
#define SET_KONFIG_PARA         0x14
#define SET_SYSTEM_PARA         0x15
#define SET_AUSG_NAME           0x16
#define SET_ERROR_QUIT          0x17
#define SET_LOGIC               0x18
#define SET_GEAR_PARA           0x19
#define SET_CAM_MOVE            0x1A
#define SET_CAM_CHANGE_SHORT    0x1B
#define SET_GATEWAY_ID          0x1C
#define SET_NOCKE_AENDERN_MT    0x20
#define SET_PARAMETER           0x21 /* General parameter according to PNR_xxx */
#define SET_OUTPUT_NAME         0x22
#define SET_EEROM_BLOCK         0x23
```

```
#define FIRST_SET_CMD              SET_NOCKE_NEU
#define LAST_SET_CMD               SET_EEROM_BLOCK /* General cam control parameters
                                                     (32-bit-values) */
#define PNR_SOFT_REV               0x0001 /* see Hard-rev */
#define PNR_HARD_REV               0x0002 /* ASCII e.g.:'3''1''2''t'= V3.12t */
#define PNR_UNIT_NAME              0x0003 /* ASCII e.g.:'L''4''8'' '= L48 */
#define PNR_UNIT_TYP               0x0004
#define PNR_VNUMBER                0x0005
#define PNR_SN                     0x0006
#define PNR_OPTION_X               0x0007

#define PNR_ENCODER_TYP            0x0010
#define PNR_RESOLUTION_PER_TURN    0x0011 /* Real */
#define PNR_NUMBER_OF_TURNS        0x0012 /* Real */
#define PNR_SCALED_ENCODER_RES     0x0013 /* Fictitious encoder value */
#define PNR_ENCODER_INVERT         0x0014
#define PNR_COUNT_RANGE            0x0018
#define PNR_COUNT_RESTORE_VALUE    0x0019
#define PNR_TIMEBASE               0x001C
#define PNR_DEADTIME_BASE_US       0x001D /* Time basis for ITC µs */

#define PNR_NUMBER_OUTPUTS         0x0020
#define PNR_NUMBER_LOCK_OUTPUTS    0x0021
#define PNR_NUMBER_DATA_RECORDS    0x0022
#define PNR_NUMBER_LOGIC_INPUTS    0x0023
#define PNR_NUMBER_ANGLE_TIME      0x0024
#define PNR_NUMBER_OUTNAME_CHAR    0x0025
#define PNR_NUMBER_PROGRAMS        0x0026
#define PNR_NUMBER_AXIS            0x0027
#define PNR_NUMBER_ANALOGOUTPUT    0x0028

#define PNR_SPEED_SCALE            0x0030 /* Refering to rev./msec => 60000 = rev./
                                             min */
#define PNR_LANGUAGE               0x0031
#definePNR_DEADTIME_TYP            0x0032
#define PNR_ZEROPOINT_OFFSET       0x0033 /* Scaled */
#define PNR_ACTIV_PROGNR           0x0034
#define PNR_ACTIV_AXIS             0x0035
#define PNR_CALC_SPEED_START       0x0036 /* TotStart scaled */
#define PNR_CALC_SPEED_STOP        0x0037 /* TotStop scaled */
#define PNR_DICNET_ID              0x0038 /* Actual value(cam control =80..95) */
#define PNR_CLEAR_LENGTH           0x0039
#define PNR_BREAK_PARA             0x003A /* (BrakeA * 0x10000) + BrakeB */
#define PNR_OUTPUT_OFF_SPEED       0x003B
#define PNR_WZ_MAXTIME             0x003C
#define PNR_WZ_TIMEBASE            0x003D
#define PNR_V_LIMIT                0x003E /* M13=1, if V_LIMIT

#define PNR_RESTART                0x004E /* Warm start with value 0x1234 */
#define PNR_CLEAR_EEROM            0x004F /* General deletion with 0x1234 */
#define PNR_STATUS_FLAGS           0x0050
#define PNR_PROC_OUT_MAPPING       0x0051 /* Mapping process data in the Fieldbus */
#define PNR_PROC_IN_MAPPING        0x0052 /* Mapping process data in the Fieldbus */
#define PNR_USED_EEROM_LEN         0x0053 /* Actual used EEROM-length */
#define PNR_S7_MODE                0x0054 /* 1=S7=>no data copy into the EEROM */
#define PNR_SCALED_NR_OF_TURNS     0x0015 /* Virtual number of revolutions */
#define PNR_ENABLE_TESTMODE 1234   0x005B /* = 0x1234 allowed shift in testmode
                                             (only L1, L2, L16 & L17)*/
```

```
#define PNR_ZEROPOINT_OFFSET_REAL  0x005D /* Real offset-value */
#define PNR_DYN_ZEROPOINT_OFFSET_  0x005E /* Dynamic offset (only IO8) */
#define PNR_INTERFACE              0x005F /* 0=RS232; 1=DICNET w/o bus termination;
                                             2=DICNET with bus termination (only
                                             L100 & L200) */
                                        /* Bit number of the STATUS_FLAGS */
#define BITNR_ENCODER_INVERT       0x00
#define BITNR_SOFT_RUNCONTROL      0x01
#define BITNR_UPDOWNLOAD_ENBALE    0x02
#define BITNR_UPDATE_DEP_TURN      0x03 /* Rotational direction-dependent output
                                            update m */
#define BITNR_ENCODER_TEST         0x04 /* Encoder */
#define BITNR_SPEED_OUT_HARD       0x05 /* Speed output 0..255 to hardware */
#define BITNR_ANALOGOUT_EXIST      0x06
#define BITNR_ENCODER_SCALABLE     0x07

#define PNR_AKT_WERT_OFFSET        0x00
#define PNR_MIN_WERT_OFFSET        0x10
#define PNR_MAX_WERT_OFFSET        0x20

#define PARA_AKTUELL               0x00
#define PARA_MIN                   0x01
#define PARA_MAX                   0x02

#define PNR_STATUS_WRITE_OK        0x00
#define PNR_STATUS_READ_WRITE      0x00
#define PNR_STATUS_READ_ONLY       0x01
#define PNR_STATUS_PROG_DISABLE    0x02
#define PNR_STATUS_EEROM_ERROR     0x03
#define PNR_STATUS_RANGE_ERROR     0x04
#define PNR_STATUS_NOT_READY       0x05
#define PNR_STATUS_REPEAT_CMD      0x06
#define PNR_PARA_NOT_EXIST         0xFF
#define CMD_RX_CHECKSUM_ERROR      224
#define CMD_RX_LEN_ERROR           225

#define LAST_COMMANDO_AKTIV        251 /* Internal error from cam control */
#define CMD_UNKNOWN_ERROR          252
#define CMD_CHECKSUM_ERROR         253 /* Checksum-error detected by cam control */
#define CMD_LENGTH_ERROR           CMD_CHECKSUM_ERROR
#define CMD_NSW_IO_ERROR           254


// Globale Variablen
unsigned char OwnID = 0;// Nur bei DICNET notwendig: Eigene Netz-ID
unsigned char ParaSendBlock[10];// Puffer für Sende-Parameter
unsigned char ParaEmpfBlock[10];//  Puffer  für  Empfangs-Parameter  in
„NSWTimeout()“


extern void SerialOut(unsigned char SendID /* Nur für DICNET*/ unsigned
char *SendBuffer, unsigned char SendLen);

extern bit NSWTimeout(unsigned int Timeout_ms);//   Wird   "Timeout_ms"
nach NSWTimeoutCounter=0 TRUE extern bit RxDataReceived();// Wird TRUE,
wenn mind. 1 Zeichen im Empfangspuffer liegt
```

```
unsigned char GetParameter(bit OnlySend, unsigned char Cmd, unsigned
char SendLen, unsigned char *EmpfLen, unsigned char SendID) /
*===========*/
/*
  Ist das Bit ONLY_SEND gesetzt, wird das Commando CMD an das selekti-
erte NSW
   geschickt zusammen mit dem "ParaSendBlock" der L‰nge "SendLen" und
danach zum
  aufrufenden Programm zur

   Ist ONLY_SEND FALSE, wurde bereits eine Anforderung an das NSW
geschickt.
  Es werden nun empfangene Daten ausgewertet oder, wenn keine vorhanden,
  das Commando WAIT zur

  Wird eine Timeout-Zeit
*/
{
unsigned char idata NSWOutputBuffer[16];
unsigned char i, Checksum, NSWPtr, Len;
char c;

  if (OnlySend == TRANSMIT)
    {
    for (c=0; c<10; c++)
      ParaEmpfBlock[c] = 0;  /* Pufferreste l
    NSWPtr = 0;
    ParaSendBlock[0] = Cmd;
    NSWOutputBuffer[NSWPtr++] = CTRL_K; /* ^K als Commando-Kennung */
    NSWOutputBuffer[NSWPtr++] = SendLen + 2; /*  Aufbau:  ^K,  Len,  ID,
Cmd, Para1 ... */
    NSWOutputBuffer[NSWPtr++] = OwnID;
    NSWOutputBuffer[NSWPtr++] = Cmd;
    Checksum = Cmd ^ OwnID ^ (SendLen + 2);
    if (SendLen > 0)
      {
      for (i=1; i<=SendLen; i++)
        {
        Checksum = Checksum ^ ParaSendBlock[i];
        NSWOutputBuffer[NSWPtr++] = ParaSendBlock[i];
        }
      }
    NSWOutputBuffer[NSWPtr++] = Checksum;
    NSWTimeoutCounter = 0;/* Timeout-Counter reset */
    SerialOut(SendID, NSWOutputBuffer, NSWPtr);
    return(0);
    }
  else         /* OnlySend == RECEIVE */
    {
    *EmpfLen = 0;         /* Default */
    if (RxDataReceived())
      {
```

```
      c = getchar();
      if (c != CTRL_K)     /* Anderer Datenrecord */
        return(WAIT_NSW);
      else
        {                              /* Antwort Record */
        Checksum = Len = (unsigned char)(getchar());/* Länge lesen */
        *EmpfLen = Len - 2;  /* Tatsächliche Anzahl der Parameter */
        if (Len > MAX_PARA_LENGTH + 2)
          return(CMD_RX_LEN_ERROR);
        i = (unsigned char)(getchar());        /*  Ohne  Fehler  ID-Echo
*/

        if (i > 127)
          return(i);              /* Sonst, Fehlerr

        Checksum = Checksum ^ i;
        Len--;                           /* und Länge korrigieren */
        for (i=0; i<Len; i++)
          {
          ParaEmpfBlock[i] = (unsigned char)(getchar());
          Checksum = Checksum ^ ParaEmpfBlock[i];
          }
        if (Checksum != (unsigned char)(getchar()))/* Checksumme testen
*/
          return(CMD_RX_CHECKSUM_ERROR);
        else
          return(ParaEmpfBlock[0]);     /* Commando return */
        }
      }
    else       /* Keine Daten vorhanden */
      {
      if (NSWTimeout(3000))    /*  3 Sek. Timeout */
        return(TIMEOUT_NSW);
      else
        return(WAIT_NSW);
      }
    }
}


bit GetDataOk(unsigned char Cmd, unsigned char SendLen,
  unsigned char *EmpfLen, unsigned char *CmdError, unsigned char NSWID)
/*=========*/
{
unsigned char Retcode;

  Retcode = GetParameter(TRANSMIT, Cmd, SendLen, EmpfLen, NSWID);

  do
    {
      switch  (Retcode  =  GetParameter(RECEIVE,  Cmd,  SendLen,  EmpfLen,
NSWID))
```

```
      {
  case LAST_COMMANDO_AKTIV:
  Retcode = GetParameter(TRANSMIT, Cmd, SendLen, EmpfLen, NSWID);/*
Erneut senden */
        Retcode = WAIT_NSW;
        break;

      case CMD_CHECKSUM_ERROR:
        *CmdError = CMD_CHECKSUM_ERROR;
        break;

      case CMD_UNKNOWN_ERROR:
        *CmdError = CMD_UNKNOWN_ERROR;
        break;

      case CMD_RX_CHECKSUM_ERROR:
      case CMD_RX_LEN_ERROR:
        *CmdError = CMD_CHECKSUM_ERROR;
        break;

      case WAIT_NSW:      /* Warten auf Empfang von NSW */
        break;

      case TIMEOUT_NSW:   /* ID im Netz nicht vorhanden */
        *CmdError = CMD_NSW_IO_ERROR;
        break;
      default:
        if (Retcode != Cmd)
          *CmdError = CMD_NSW_IO_ERROR;
        else
          {
          *CmdError = 0;
          if ((Cmd >= FIRST_SET_CMD) && (Cmd <= LAST_SET_CMD))
            {
            if (ParaEmpfBlock[1] == LAST_COMMANDO_AKTIV)  /*  In  alten
Ger%ten Fehlercode im  1.Para bei SET-Commands */
              {
               Retcode = GetParameter(TRANSMIT, Cmd, SendLen, EmpfLen,
NSWID); /* Erneut senden */
              Retcode = WAIT_NSW;
              break;
              }
            if ( (ParaEmpfBlock[1] == CMD_UNKNOWN_ERROR) ||
                 (ParaEmpfBlock[1] == CMD_CHECKSUM_ERROR) )
              {
              *CmdError = CMD_NSW_IO_ERROR;
              Cmd = 255;
              }
            }
          }
        break;
      }
```

```
    }
  while (Retcode == WAIT_NSW);

  return(Retcode == Cmd);
}


bit GetOutputOk(unsigned char *Output)
  /*============*/
{
unsigned char ErrorCode, InLen, i;

  if (GetDataOk(GET_OUTPUT, 0, &InLen, &ErrorCode, 0))
    {
    for (i=0; i<InLen; i++)
      *(Output+i) = ParaEmpfBlock[i+1];
    return(TRUE);
    }
  return(FALSE);
}


bit SetIdleTimeOk(unsigned char Prog, Ausg, unsigned int OnTime, Off-
Time, unsigned char Drehrichtung)
   /*===========*/
{
unsigned char ErrorCode, InLen;

  ParaSendBlock[1] = Prog;
  ParaSendBlock[2] = Ausg;
  ParaSendBlock[3] = OnTime / 256;
  ParaSendBlock[4] = OnTime % 256;
  ParaSendBlock[5] = OffTime / 256;
  ParaSendBlock[6] = OffTime % 256;
  ParaSendBlock[7] = Drehrichtung;

  if (GetDataOk(SET_TOTZEIT, 7, &InLen, &ErrorCode, 0))
    {
    if (ParaEmpfBlock[1] == 0)
      return(TRUE);
    }

  return(FALSE);
}


bit GetIdleTimeOk(unsigned char Prog, Ausg, unsigned int *OnTime,
unsigned int *OffTime)
  /*=============*/
{
unsigned char ErrorCode, InLen;
```

```
  ParaSendBlock[1] = Prog;
  ParaSendBlock[2] = Ausg;

  if (GetDataOk(GET_TOTZEIT, 2, &InLen, &ErrorCode, 0))
    {
         *OnTime    =  ((unsigned  int)(ParaEmpfBlock[1])  *  256)  +
ParaEmpfBlock[2];
    if (InLen == 2)
      *OffTime = *OnTime;
    else
            *OffTime  =  ((unsigned  int)(ParaEmpfBlock[3])  *  256)  +
ParaEmpfBlock[4];
    return(TRUE);
    }
  return(FALSE);
}
```

4 Appendix A

4.1 Command tale (not for new applications)

Command's name	Command's value	Command's meaning
GET_TYP	0x06	4.2.1.1
GET_MAXPARA	0x07	4.2.1.2
GET_CONFIG_PARA	0x0B	4.2.1.3
GET_SYSTEM_PARA	0x0C	4.2.1.4
GET_OUT_NAME	0x0D	4.2.1.5
GET_PROT_REC	0x40	4.2.1.6
GET_GEAR_PARA	0x42	4.2.1.7
GET_OUTPUT_MATTE	0x31	4.2.1.8 (pattern units only)
SET_ACT_PARA	0x13	4.2.2.1
SET_CAM_CHANGE	0x11	4.2.2.2
SET_CONFIG_PARA	0x14	4.2.2.3
SET_SYSTEM_PARA	0x15	4.2.2.4
SET_OUT_NAME	0x16	4.2.2.4.1
SET_GEAR_PARA	0x19	4.2.2.5
SET_OUTPUT_MATTE	0x32	4.2.2.6 (pattern units only)

4.1.1 Explanations

Halfway through the year of 1997 the commands SET_PARAMETER and GET_PARAMETER moreover GET_OUTPUT_NAME and SET_OUTPUT_NAME were completed in order to achieve a higher amount of flexibility for applications in the future. By using these commands and the parameter table (see chapter parameter table) all commands mentioned below can be replaced. That way a cut results for the below mentioned software revisions:

Equipment layout	New commands (see above) from revision
LOCON 1, LOCON 2	V4.1
LOCON 7	V4.1
LOCON 9	V4.1
LOCON 16, LOCON 17	V4.1
LOCON 24, LOCON 48, LOCON 64	V2.0
LOCON 32 all variants	V3.0
ROTARNOCK 1, ROTARNOCK 2	V4.1
Multiturn-ROTARNOCK	V4.1

The following commands are available in all versions of the above mentioned cam controls, yet they should not be used any more in cooperation with devices, that support the new command record.

4.2 Description of commands not to be used any more

4.2.1 Parameter received by the PLS

4.2.1.1 GET_TYP

Parameters to the PLS: None

Parameters from the PLS:
1. Type number (coded from 0..255)
2. Software revision, 1st character
3. Software revision, 2nd character
4. Software revision, 3rd character

4.2.1.2 GET_MAXPARA

Parameters to the PLS: None

Parameters from the PLS:
1. Number of programs
2. Maximum output number
3. Maximum real encoder value (High) --- 0 = 65536 !
4. Maximum real encoder value (Low) /
5. Maximum idle time (High) in idle time-units
6. Maximum idle time (Low) in idle time-units
7. Number of axes
8. Log2 of the max. number of revolutions (only MT)

Note: In the case of an incremental encoder, the maximum counting area is transferred in place of the max. encoder value. In the case of a Multiturn encoder (MT), the number of increments per revolution is transferred.

4.2.1.3 GET_CONFIG_PARA

Parameters to the PLS: None

Parameters from the PLS:
1. Network ID (no significance in the case of RS232 or IBS)
2. Encoder type (1 = abs, 2 = inc, 3 = SSI, 4 = DSI, 5 = Timer)
3. Real encoder resolution/timebase (High) --0=65536
4. Real encoder resolution/timebase (Low) /
5. ITC (.1 = block, .2 = bit, .3 = bl-I/O, .4 = bit-I/O)
6. Number of locked outputs
7. Log2 of the number of revolutions (only in the case of MT)
8. Special configuration

X	X	X	X	X	X	X	X		
I	I	I	I	I	I	I	+	1	= Direction-dependent outputs
I	I	I	I	I	+	+	-	0-3	= Number of logic blocks, 8 outputs each
I	I	I	I	I	+	+	-	0-3	= Number of logic blocks, 8 outputs each
I	I	I	I	+	-	-	-	1	= Gear function present
I	I	I	+	-	-	-	-	1	= Analog output present
+	+	+	-	-	-	-	-	0	(Reserve)

Note: Parameters 7 and 8 are not transferred on all PLSs!

The output type of the analog encoder is transferred in place of the encoder type in the case of an analog encoder.

4.2.1.4 GET_SYSTEM_PARA

Parameters to the PLS: None

Parameters from the PLS:
1. Analog output factor (High)
2. Analog output factor (Low)
3. Fictitious zero offset/counting range (H) 0 = 65536
4. Fictitious zero offset/counting range (Low)

5a. Zero offset/counting range (bits 16..23) **only MT**
5b. Direction of rotation (0 = pos., 1 = neg.) **otherwise**
6a. Language (0 = Ger, 1 = Eng, 2 = Fre, 3 = Ital, 4 = Spa)
6b. See Note

7. Speed indication factor (High) -> 0... 9999 (rev./s.)
8. Speed indication factor (Low) -> 0... 9999 (rev./s.)

Note: On MT, the 6th parameter contains the information on language and direction of rotation according to the following formula: Parameter 6 = (rotational direction * 16) + language.

4.2.1.5 GET_OUT_NAME

Parameters to the PLS:
1. (BlockNo * 64) + Output No. (see Note)

Parameters from the PLS:
1. Character 1
2. Character 2

...

8. Character 8

Note: The output name of each output is dissected into max. 4 blocks of 8 characters each (max. length of the name = 32 characters) in order to achieve the fastest possible transmission of the output names. This means that each of the 4 blocks can be scanned by each of the maximum 64 outputs via the 1st parameter. Here the following counting areas apply: BlockNo. 0..3, Output No. 0..63 (corresponds to output 1-64).

4.2.1.6 GET_PROT_REC

Parameters to the PLS:
1. Protocol number (High)
2. Protocol number (Low)
3. Block number (0..2)
4. Action (0 = None, 1 = Delete entries, 2 = Set time)
5. System time (MSB)
6. System time (.....)
7. System time (.....)
8. System time (LSB)

Parameters from the PLS:
1. Status* (High) (0 in the case of block number)
2. Status* (Middle)
3. Status* (Low)
4. Type (0 = data record,1 = byte,2 = integer)
5. Day
6. Month
7. Hour
8. Minute

Parameters from the PLS: 1.-8. Old data record (1 in the case of block number)

Parameters from the PLS: 1.-8. New data record (2 in the case of block number)

Note: The system time is taken from the cam control if the “Action” byte is set to 2. It is transferred as a long integer in unit "10 ms".

If a 1 is transferred in the "Action" byte, all stored protocol data is deleted.

The block number of the request determines which block of the data record is transferred by the cam control. *The status consists of:

(number of protocols * 4096) + (number of the last protocol)

4.2.1.7 GET_GEAR_PARA

Parameters to the PLS: None

Parameters from the PLS:
1. Fictitious encoder resolution (High)
2. Fictitious encoder resolution (Low)
3. Decimal-point position
4. Designation for unit of measurement, 1st character
5. Designation for unit of measurement, 2nd character
6. Designation for unit of measurement, 3rd character
7. Maximum possible fictitious encoder resolution (High)
8. Maximum possible fictitious encoder resolution (Low)

4.2.1.8 GET_OUTPUT_MATTE (only pattern unit)

Parameters to the PLS:
1. Program number (0..number of programs -1)
2. Fictitious encoder position (High)
3. Fictitious encoder position (Low)

Parameters from the PLS:
1. Last position (High)
2. Last position (Low)
3. Output assignment 1..8
4. Output assignment 9..16 (if present)

...

Note: The PLS always transfers the statuses of all outputs. If, for instance, the devices feature 32 outputs, 4 bytes are transferred.

4.2.2 Send parameters to PLS

4.2.2.1 SET_ACT_PARA

Parameters to the PLS:
1. Set current program
2. Set active axis
3. Status
   Bit 0 = Clear All
   Bits 1-7 = Reserved (currently = 0)

Parameters from the PLS:
1. Acknowledgement (0 = ok, otherwise error number)

If bit 0 (LSB) of the status byte is set to 1, the PLS deletes all user data!

4.2.2.2 SET_CAM_CHANGE

Parameters to the PLS:
1. Program No. (0..number of programs -1)
2. Output No. (1..number of outputs)
3. Switch-on point (High byte)
4. Switch-on point (Low byte)
5. Switch-off point (High byte)
6. Switch-off point (Low byte)
7. Cam number (High byte)
8. Cam number (Low byte)

Parameters from the PLS:
1. Acknowledgement (see Note)

Note: The cam number must be identical to the number transferred by the PLS with request GET_NEXT_NOCKE or GET_BACK_NOCKE.

For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS during programming, this can be queried by GET_STATUS.

If the real encoder resolution differs from the fictitious encoder resolution, the fictitious switch-on and switch-off values are transferred.

4.2.2.3 SET_CONFIG_PARA

Parameters to the PLS:
1. Axis No. (0..15)
2. Encoder type (1 = abs, 2 = inc, 3 = SSI, 4 = DSI, 5 = Timer)
3. Real encoder resolution (High)
4. Real encoder resolution (Low)
5. Idle time compensation. (1 = block, 2 = bit, 3 = block On-Off)
6. Number of locked outputs
7. Log2 of the max. number of revolutions (only MT)

Parameters from the PLS:
1. Acknowledgement (0 = ok, otherwise error number)

Note: On a Multiturn encoder, the number of increments/revolution is transferred in place of the max. encoder value.

0 is always transferred for axis No. in the case of an RS232 link.

4.2.2.4 SET_SYSTEM_PARA

Parameters to the PLS:
1. Analog output factor (High) speed at 10V (only LOCON 32)
2. Analog output factor (Low) speed at 10V (only LOCON 32)
3. Fictitious zero offset/counting area (High)
4. Fictitious zero offset/counting area (Low)

5a. Zero offset/counting area (bits 16..23) **only MT**
5b. Direction of rotation (0 = pos., 1 = neg.) **otherwise**
6a. Language (0 = Ger, 1 = Eng, 2 = Fre, 3 = Ital, 4 = Spa)
6b. See Note

7. Speed indication factor (High) -> 0... 9999 (rev./s.)
8. Speed indication factor (Low) -> 0... 9999 (rev./s.)

Parameters from the PLS:
1. Acknowledgement (0 = ok, otherwise error number)

Note: If an absolute encoder has been configured with SET_CONFIG_PARA, parameters 3 and 4 are interpreted as the zero offset. Otherwise, this defines the counting area of the incremental PLS (default 8192).

On MT, the 6th parameter contains the information for language and direction of rotation in accordance with the following formula:

Parameter 6 = (direction of rotation * 16) + language.

4.2.2.4.1 SET_OUT_NAME

Parameters to the PLS:
1. (BlockNo * 64) + Output No.. (see Note)
2. Character 1
3. Character 2

...

9. Character 8

Parameters from the PLS:
1. Acknowledgement (0 = ok, otherwise error number)

Note: The output name of each output is dissected into max. 4 blocks of 8 characters each (max. length of the name = 32 characters) in order to achieve the fastest possible transmission of the output names.

This means that each of the 4 blocks can be transferred in targeted manner by each of the maximum 64 outputs.

The following number ranges apply:
BlockNo 0..3
Output No.0..63 (corresponds to output 1-64)

4.2.2.5 SET_GEAR_PARA

Parameters to the PLS:
1. Fictitious encoder resolution (High)
2. Fictitious encoder resolution (Low)
3. Decimal-point position
4. Designation for unit of measurement, 1st character
5. Designation for unit of measurement, 2nd character
6. Designation for unit of measurement, 3rd character

Parameters from the PLS:
1. Acknowledgement (see Note)

4.2.2.6 SET_OUTPUT_MATTE (only pattern unit)

Parameters to the PLS:
1. Program number (0..max)
2. Fictitious encoder position (High)
3. Fictitious encoder position (Low)
4. Output assignment 1..8
5. Output assignment 9..16
6. Output assignment 17..23
7. Output assignment 24..32
8. Command:
   0 = Change outputs
   1 = Insert position
   2 = Delete position
   3 = Rebuild cam array + output enable
   (Prog + Pos -> Curr.Prog + Curr.Pos)
   4 = Disable output

Parameters from the PLS:
1. Acknowledgement (see Note)

Note: For programming reasons, the acknowledgement of the PLS is always 0. If an error occurs in the PLS during programming, this can be queried by GET_STATUS.