Bedienerhandbuch

Kommunikationsprofil

für Nockensteuerungen der 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

Vorwort

Das vorliegende Bedienerhandbuch gibt Anwendern und OEM-Kunden alle Informationen, die für die Installation und Bedienung des in diesem Handbuch beschriebenen Produktes benötigt werden.

Alle Angaben in diesem Handbuch sind nach sorgfältiger Prüfung zusammengestellt worden, gelten jedoch nicht als Zusicherung von Produkteigenschaften. Dennoch kann keine Haftung für Fehler übernommen werden. Weiter hält sich die DEUTSCHMANN AUTOMATION vor, Änderungen an den beschriebenen Produkten vorzunehmen, um Zuverlässigkeit, Funktion oder Design zu verbessern.

DEUTSCHMANN AUTOMATION haftet ausschließlich in dem Umfang, der in den Verkaufs- und Lieferbedingungen festgelegt ist.

Version 7.0 vom 13. November 2008, Art.-No. V2064

P/C: B

1 Einführung

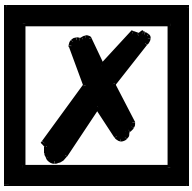
Um den Anforderungen des Marktes gerecht zu werden, wird von **Deutschmann Automation** verstärkt der Einsatz von Nockensteuerungen mit abgesetzter Bedien- und Anzeigeeinheit unterstützt.

Da applikationsspezifisch immer wieder unterschiedliche Kombinationen zwischen Nockensteuerungen und Terminals benötigt werden, war es notwendig, eine einheitliche Schnittstelle (Kommunikationsprofil) zu definieren, die von allen Terminals und Nockensteuerungen aus dem Lieferprogramm der DEUTSCHMANN AUTOMATION unterstützt wird.
Damit ist die Möglichkeit gegeben, daß sich jeder Anwender die für ihn am besten geeignete Kombination zusammenstellt.

Durch Offenlegung dieses Kommunikationsprofiles in dieser Spezifikation erhält der Anwender außerdem die Möglichkeit, mit DEUTSCHMANN-Nockensteuerungen zu kommunizieren, und somit vorhandene Informationen (Geberposition, Geschwindigkeit, ...) für seine eigenen Anwendung zu nutzen oder die Nockensteuerung über ein eigenes Terminal zu bedienen.

Basierend auf diesem Kommunikationsprofil hat der Anwender sogar die Möglichkeit, mit übergeordneten Bussystemen (Interbus-S, PROFIBUS, CAN ...) Daten auszutauschen.

1.1 Symbole



Besonders **wichtige Textpassagen** erkennen Sie am nebenstehendem Piktogramm.

Diese Hinweise sollten Sie **unbedingt beachten**, da ansonsten Fehlfunktionen oder Fehlbedienung die Folge sind.

1.2 Begriffliches - Definitionen

1.2.1 Abkürzungen

Abkürzung	Bedeutung
MT	Multi-Turn-Geber
IBS	Interbus-S
NS	Nockensteuerung
ERSTE_NOCKE	7F00 (Hex) = 32512
ERSTE_NOCKE (nur MT)	FF7F00 (Hex)
TZK	Totzeitkompensation
WZ	Winkel-Zeit-Nocke

1.2.2 Begriffserklärungen

Skaliert Bei Nockensteuerungen mit der Funktion 'Skalierbarer Geberwert', auch unter dem Begriff 'Getriebefaktor' bekannt, muß zwischen den realen und skalierten Werten sorgfältig unterschieden werden. Aus diesem Grund ist bei den entsprechenden Parameter immer angegeben, ob es sich um die realen oder skalierten Werte handelt.

1.2.3 Anregungen

Für Anregungen, Wünsche etc. sind wir stets dankbar und bemühen uns, diese zu berücksichtigen. Hilfreich ist es ebenfalls, wenn Sie uns auf Fehler aufmerksam machen.

2 Kommunikationsbeschreibung

2.1 Hardware

Das Kommunikationsprofil ist unabhängig von der tatsächlichen Hardware-Schnittstelle und beschreibt die Dienste, die gemäß dem Layer 7 des ISO/OSI-Modells von der Nockensteuerung zur Verfügung gestellt werden. Als Kommunikaitionsmedium kann eine der nachfolgend beschriebenen Schnittstellen verwendet werden, sofern sie von der eingesetzten Nockensteuerung unterstützt wird.

2.1.1 RS232-Schnittstelle

Die Übertragung auf der RS232-Schnittstelle erfolgt im Full-Duplex-Verfahren mit 9600 Baud, 8 Datenbit, 1 Start- und 1 Stopbit, sowie keinem Paritätsbit.
Es ist ausschließlich eine reine Punkt zu Punkt-Verbindung zwischen der Nockensteuerung und einem Teilnehmer möglich.

2.1.2 DICNET®-Bus-Schnittstelle

Bei DICNET® (DEUTSCHMANN-Industrie-Controller-Net) handelt es sich um einen Feldbus, der beim Physical-Layer gemäß dem ISO-OSI-Schichtenmodell der DIN 19245 Teil 1 entspricht; d. h. es wird mit einer RS485-Zweitdraht-Leitung eine Verbindung zwischen allen Teilnehmern im Netz hergestellt.
Die physikalische Anordnung ist somit ein Bussystem, an dem die Teilnehmer beliebig an- und abgeschaltet werden können.
Logisch handelt es sich um einen Token-Ring; d. h. es darf immer nur der Teilnehmer, der die Buszugriffsberechtigung (Token) besitzt auf dem Bus senden. Besitzt er keine Daten für einen anderen Teilnehmer, gibt er den Token an seinen Nachbarn, der in einer Konfigurationsphase ermittelt wurde, weiter.
Durch dieses Prinzip wird eine deterministische Buszykluszeit erreicht; d. h. die Zeit (worst-case) bis ein Datenpaket gesendet werden kann, ist genau berechenbar.
Beim Zu- oder Abschalten eines Teilnehmers erfolgt eine automatische Neukonfiguration.
Die Übertragungsbaudrate beträgt 312,5 kBaud bei einer Länge von 11 Bit/Byte. Es können maximal 127 Teilnehmer an einem Bus betrieben werden, wobei Datenpakete von maximal 14 Byte pro Zyclus geschickt werden.
Es erfolgt eine automatische Überprüfung der empfangenen Informationen und eine Fehlermeldung bei einem zweifachen Übertragungsfehler.
Die Verwendung dieser Schnittstelle setzt genaue Kenntnisse über den internen Aufbau des DEUTSCHMANN-Busses (DICNET) voraus, so daß hier nicht näher darauf eingegangen werden soll.

2.1.3 Feldbus-Schnittstelle

Es besteht die Möglichkeit, einer Umsetzung von RS 232 oder DICNET auf alle wichtigen Feldbusse. Dazu wird ein Gateway der Serie UNIGATE-RS verwendet, das einen Feldbus mit maximal 16 Nockensteuerungen verbindet.
Detaillierte Informationen zum Feldbusanschluß finden sich in Kap. 2.3.
Außerdem hat Deutschmann Automation Nockensteuerungen mit integriertem Feldbus im Programm.

2.2 Allgemeiner Kommunikationsaufbau

Bei der Kommunikation wird ein strenges Master-Slave-Verfahren eingehalten, wobei das Nokkensteuerung (NS) immer als Slave arbeitet, das nur auf Anforderung von dem angeschlossenen Kommunikationspartner (Terminal, PC, SPS etc.) Daten sendet.

2.3 Kommunikationsablauf

2.3.1 Protokoll-Rahmen für DICNET und RS232

Jeder Datenrecord der vom NS oder vom Terminal ausgetauscht wird, hat bei Verwendung der RS232- oder DICNET-Schnittstelle folgenden einheitlichen Aufbau:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Konstantes Schlüsselzeichen
2	Länge	Recordlänge Byte 3 (incl). - Byte N+4 (incl.)
3	Eigener Netz-ID	Eigene Adresse im DICNET (bei RS232 = 0)
4	Command	Kommando gemäß Befehlstabelle (s. u.)
5	Parameter 1	Parameter 1
...	...	...
N+4	Parameter N	Parameter n
N+5	Checksumme	Checksumme (XOR Byte 2 bis Byte N+4)

Die Länge wird ermittelt ab dem Byte "Netz-ID" bis zum "Parameter N" (jeweils inklusiv). Die Checksumme und die ersten beiden Bytes werden nicht mitgerechnet.
Die Checksumme ermittelt sich aus der Exclusiv-Oder-Verknüpfung (XOR) der Bytes "Länge" bis "Parameter N" (jeweils inklusiv).

Es sind maximal 9 Parameter möglich!

2.3.2 Protokoll-Rahmen für Interbus-S

Der Interbus-S verfügt gegenüber den oben beschrieben Schnittstellen über folgende Besonderheiten:

- Feste Datenlänge (konfigurierbar über WINGATE) bis 32 Byte (ab dem 5. Byte werden die Ausgangszustände des Gerätes ausgegeben)
- Ständiger zyklischer Datenaustausch
- Keine Netz-Adresse
- Eigenständige Datensicherung

Aus diesem Grund erfolgt bei Verwendung der Interbus-Schnittstelle die Kommunikation gemäß dem nachfolgend beschriebenen Verfahren.

2.3.2.1 Prozessdaten (Output) im Normalbetrieb

Die Nockensteuerung überträgt in jedem Interbus-S-Zyclus normalerweise folgende Daten:

1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	7.Byte	....
Position High (00..7F)	Position Low	Geschw. High (00..7F)	Geschw. Low	Output 1-8	Output 9-16	Output 17-24	Output ....

Dabei ist zu beachten, daß das höchstwertige Bit (MSB) im 1.Byte und im 3.Byte **immer** 0 ist; d. h. Position und Geschwindigkeit liegen immer im Bereich 0..32767.

2.3.2.2 Prozessdaten (Output) im Fehlerzustand

Erkennt die Nockensteuerung einen Fehler, ändert sich der Datensatz wie folgt:

1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	7.Byte	...
Position High (00..7F)	Position Low	FehlerNr High (80..FF)	FehlerNr Low	Output 1-8	Output 9-16	Output 17-24	Output ...

Es ist zu beachten, daß im Fehlerzustand das höchstwertige Bit (MSB) des 3.Byte **immer** 1 ist; d. h. die Fehlernummer hat einen Offset von 32768. Erkennt die Nockensteuerung beispielsweise einen Geberfehler (Fehlernummer 100) wird als 3. und 4.Byte der Wert 32868 übertragen. Dieser Datensatz (mit der Fehlernummer) wird solange zyklisch übertragen, bis der Fehler mit dem Kommando "SET_ERROR_QUIT" quittiert wurde.

2.3.2.3 Prozessdaten (Input)

Werden vom Master keine Daten außer den zyclischen Prozessdaten (s.o.) benötigt, sendet er einen Prozessdatensatz, bei dem im 1. Byte das MSB nicht gesetzt ist (00..6F).
Die restlichen Bytes werden nicht ausgewertet.

1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	...	n.Byte
00..6F	x	x	x	x	x	x	x

2.3.2.4 Parameteraustausch

Der Austausch von Parametern erfolgt dadurch, daß **anstatt** eines Prozessdatensatzes (s. o.) in einem Interbus-S-Zyclus ein Parametersatz übertragen wird.
Dabei ist zu beachten, daß bei einem Parametersatz das MSB im 1. Byte **immer** gesetzt ist; d.h: zum Befehlscode (01H..6FH) muß 128 dazuaddiert werden.
Eine Unterscheidung zwischen Parameter- und Prozessdatensatz kann somit durch das MSB des 1. Bytes erfolgen.

Da es sich um ein Master-Slave-System handelt, wird ein Parameteraustausch **immer** vom Master eingeleitet; d. h. er überträgt anstelle des Prozessdatensatzes (s. o.) das gewünschte Kommando aus der nachfolgend beschriebenen Befehlsliste.

Der Aufbau dieses Parametersatzes ist wie folgt:

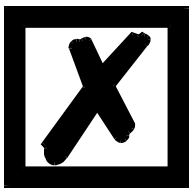
1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	...	10.Byte
Befehl 80..EF	1.Para	2.Para	3.Para	4.Para	5.Para	...	9.Para

Es ist zu beachten, daß immer soviel Parameter übertragen werden, wie die IBS-Länge minus 1 beträgt. Sind in dem entsprechenden Befehl weniger Parameter spezifiziert, werden als restliche Parameter 0 übertragen. Die Nockensteuerung wertet nun diese Anfrage des Masters aus, und antwortet seinerseits mit einem Parametersatz, der, wenn kein Fehler aufgetreten ist, wie folgt aufgebaut ist:

1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	...	10.Byte
Befehl 80..EF	1.Para	2.Para	3.Para	4.Para	5.Para	...	9. Para

Das 1.Byte ist dabei identisch mit dem 1.Byte der Anfrage.
Im Fehlerfall (Kapitel Fehlerbehandlung) wird im 1. Byte der Fehlercode (F0..FF) übertragen.

1.Byte	2.Byte	3.Byte	4.Byte	5.Byte	6.Byte	...	10.Byte
Befehl F0..FF	1.Para	2.Para	3.Para	4.Para	5.Para	...	9.Para



Es erfolgt immer eine Antwort auf eine Anfrage des Masters!

2.3.2.5 Zeitverhalten des Kommunikationsablaufes

Startet der Master eine Anfrage bei der Nockensteuerung, ist es möglich, daß bis zur Antwort einige Interbus-S-Zyklen vergehen. Um ein sauberes Zeitverhalten beim Parameteraustausch zu erreichen, muß der Master den Parameterdatensatz solange übertragen, bis er von der Nockensteuerung beantwortet wurde. Ebenso sendet die NS solange den unveränderten Datensatz bis es einen neuen Datensatz empfängt.

2.3.3 Protokoll-Rahmen für LONWORKS

Das Gateway arbeitet im Netz als LON Teilnehmer und schreibt ab der ersten Anfrage zyklisch die Antwort in die parametrierte SNVT.

Jede Anfrage hat folgendes Format:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
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	

Es werden im LONbus vom LON Teilnehmer an das Gateway die parametrierten SNVTs übertragen, jedoch sind nur die Parameter, die mit dem Längenbyte spezifiziert werden gültig. (Gültige Parameter = Länge - 2).
Der Antwortrecord des Gateways ist abhängig von den Konfigurationen der SNVTs. Die Gesamtlänge ergibt sich aus der Länge der einzelnen SNVTs.
Folgende Typen könnten beispielsweise einmal als Input und Output parametriert werden: Typ 36, 86 oder 96.

2.3.3.1 Antwort der Parameter und Prozessdaten

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge oder Errorcode	Byte 3 (incl) bis letzter Parameter (incl) oder Fehlernummer, wenn Wert > 127
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
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	

Hier sind ebenfalls wie bei der Anfrage nur die mit dem Längenbyte spezifizierten Parameter gültig.
Der Antwortrecord ist dann gültig, wenn die Bytes 1, 3 und 4 mit den entsprechenden Bytes im Anfragerecord identisch sind.
Tritt bei der Anfrage bei der Nockensteuerung ein Fehler auf, wird der entsprechende Fehler im Byte 2 (anstatt der Länge) übertragen. Der Fehlerwert ist immer größer als 127 und kann dem Kapitel „Fehlerbehandlung“ entnommen werden.

2.3.4 Protokoll-Rahmen für Profibus-DP

2.3.4.1 Kommunikation über Gateway

Das Gateway arbeitet im Netz als reiner Profibus-Slave und wertet zyklisch die Masteranfragen aus.

Jede Masteranfrage hat folgendes Format:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
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	

Es werden im Profibus vom Master an das Gateway (Slave) immer 13 Byte übertragen, jedoch sind nur die Parameter, die mit dem Längenbyte spezifiziert werden gültig. (Gültige Parameter = Länge - 2).
Der Antwortrecord des Gateways ist abhängig von der Konfiguration (s. GSD-Datei) 13 oder 17 Byte lang und sieht wie folgt aus.

2.3.4.2 Konfiguration Parameterdaten

Gültig, wenn über die GSD-Datei 13 Byte Daten zum PB-Master konfiguriert wurden:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
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	

2.3.4.2.1 Konfiguration von Parameter und Prozessdaten

Gültig, wenn 17 Byte zum PB-Master konfiguriert sind.

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge oder Errorcode	Byte 3 (incl) bis letzter Parameter (incl) oder Fehlernummer, wenn Wert > 127
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gem. Befehltabelle
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 bei 17 Byte-Konfiguration
15	Pos. L	
16	Speed H	
17	Speed L	

Hier sind ebenfalls wie bei der Masteranfrage nur die mit dem Längenbyte spezifizierten Parameter gültig.
Der Antwortrecord ist dann gültig, wenn die Bytes 1, 3 und 4 mit den entsprechenden Bytes im Anfragerecord identisch sind.
Tritt bei der Anfrage bei der Nockensteuerung ein Fehler auf, wird der entsprechende Fehler im Byte 2 (anstatt der Länge) übertragen. Der Fehlerwert ist immer größer als 127 und kann dem Kapitel „Fehlerbehandlung“ entnommen werden.

2.3.4.3 Direkte Kommunikation mit der Nockensteuerung

Je nach Projektierung durch die GSD-Datei ist einer der folgenden Protokoll-Rahmen möglich:

2.3.4.3.1 Nur Parameterdaten

Masteranfrage:

Byte-Nr.	Bezeichnung	Bedeutung
1	Auftragsnummer	Eindeutige Kennzeichnung für Anfrage
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	Auftragsart	0 = einmalig, 1 = zyklisch
4	Command	Kommando gemäß Befehltabelle
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	

Antwort Nockensteuerung:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
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	

2.3.4.3.2 Parameter und Prozessdaten (2 Byte Logik)

Masteranfrage:

Byte-Nr.	Bezeichnung	Bedeutung
1	Auftragsnummer	Eindeutige Kennzeichnung für Anfrage
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	Auftragsart	0 = einmalig, 1 = zyklisch
4	Command	Kommando gemäß Befehltabelle
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	Logikinput 1 - 8	Eingänge für Logikverknüpfung
15	Logikinput 9 - 16	Eingänge für Logikverknüpfung

Antwort Nockensteuerung:

Byte-Nr.	Bezeichnung	Bedeutung
1	Auftragsnummer	Eindeutige Kennzeichnung für Anfrage
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	Auftragsart	0 = einmalig, 1 = zyklisch
4	Command	Kommando gemäß Befehltabelle
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 als 32Bit-Logik-Wert
15	Position 23 - 16	
16	Position 15 - 8	
17	Position 7 - 0	
18	Speed H	Geschwindigkeit
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	Aktuelles Programm	
27	Error Nummer	

2.3.4.3.3 Parameter- und Prozessdaten (1 Byte Logik)

Masteranfrage:

Byte-Nr.	Bezeichnung	Bedeutung
1	Auftragsnummer	Eindeutige Kennzeichnung für Anfrage
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	Auftragsart	0 = einmalig, 1 = zyklisch
4	Command	Kommando gemäß Befehltabelle
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	Logikinput 1 - 8	Eingänge für Logikverknüpfung

Antwort Nockensteuerung:

Byte-Nr.	Bezeichnung	Bedeutung
1	Auftragsnummer	Eindeutige Kennzeichnung für Anfrage
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	Auftragsart	0 = einmalig, 1 = zyklisch
4	Command	Kommando gemäß Befehltabelle
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 als 32Bit-Logik-Wert
15	Position 23 - 16	
16	Position 15 - 8	
17	Position 7 - 0	
18	Speed H	Geschwindigkeit
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	Aktuelles Programm	
27	Error Nummer	

2.3.5 Protokoll-Rahmen für ARCNET

Das Gateway arbeitet im ARCNET als reiner Multi-Master und updated nach jeder Masteranfrage den entsprechenden Sendepuffer.

Jede Masteranfrage hat folgendes Format:

Byte-Nr.	Bezeichnung	Bedeutung
1	ID (0..255)	Ziel ID des gewünschten GW
2	Ctrl-K (OB Hex)	Schlüsselzeichen
3	Länge	Byte 3 (incl) bis letzter Parameter (incl)
4	ID (0..15)	Nockensteuerung-ID
5	Command	Kommando gemäß Befehltabelle
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	

Es werden im ARCNET immer 18 Byte übertragen, jedoch sind nur die Parameter, die mit dem Längenbyte spezifiziert werden gültig. (Gültige Parameter = Länge - 2).

Der Antwortrecord des Gateways ist ebenfalls 18 Byte lang und sieht wie folgt aus:

Byte-Nr.	Bezeichnung	Bedeutung
1	ID (1..255)	Quell ID des beantwortenden Gateways
2	Ctrl-K (0B Hex)	Schlüsselzeichen
3	Länge oder Errorcode	Byte 3 (incl) bis letzter Parameter (incl) oder Fehlernummer, wenn Wert > 127
4	ID (0..15)	Nockensteuerung-ID
5	Command	Kommando gemäß Befehltabelle
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	
15	NS Pos	High Byte
16	NS Pos	Low Byte
17	NS Speed	High Byte
18	NS Speed	Low Byte

Hier sind ebenfalls wie bei der Masteranfrage nur die mit dem Längenbyte spezifizierten Parameter gültig.
Der Antwortrecord ist dann gültig, wenn die Bytes 2, 4 und 5 mit den entsprechenden Bytes im Anfragerecord identisch sind.
Tritt bei der Anfrage bei der Nockensteuerung ein Fehler auf, wird der entsprechende Fehler im Byte 3 (anstatt der Länge) übertragen. Der Fehlerwert ist immer größer als 127 und kann dem Kapitel „Fehlerbehandlung“ entnommen werden.

2.3.6 Protokoll-Rahmen für DeviceNet

Das Gateway arbeitet im DeviceNet als „Group 2 Only Slave“. Es werden die Zugriffsverfahren „Polling“, „Bit-Strobe“ und „Change of state“ unterstützt, die nachfolgend beschrieben sind. Eine Parametrierung der Daten wird zur Zeit noch nicht unterstützt.

2.3.6.1 Polling

Beim Polling sendet der DeviceNet-Master ein 11-Byte langes Telegramm an das Gateway und erhält als Antwort eine Sequenz von 13 Byte. Über dieses Zugriffsverfahren ist es dem DeviceNet-Master möglich, jeden beliebigen Parameter der angeschlossenen Nockensteuerung zu lesen oder zu schreiben.
Die Bedeutung der Sende- und Empfangsbytes ist nachfolgend im Kapitel „Parameterdaten“ beschrieben.

2.3.6.2 Bit-Strobe

Beim Bit-Strobe-Zugriff wird vom Master das Kommando „Bit-Strobe“ ohne weitere Daten gesendet. Er erhält daraufhin vom Gateway 8 Byte als Antwort, die nachfolgend im Kapitel „Prozessdaten“ beschrieben sind.

2.3.6.3 Change of state

Bei diesem Verfahren sendet das Gateway selbständig die im Kapitel „Prozessdaten“ beschriebene Sequenz, sobald sich mindestens ein Bit der Prozessdaten geändert hat.

2.3.6.4 Parameterdaten

2.3.6.5 DeviceNet-Master zum Gateway

Es werden vom DeviceNet-Master folgende 11 Byte zum Gateway gesendet:

Byte	Bezeichnung	Bedeutung
1	ID (0..15)	Nockensteuerung-ID im DICNET
2	Command	Kommando gemäß Kapitel „Befehlstabelle“
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	

Die Parameter sind nachfolgend im Kapitel des jeweiligen Kommandos beschrieben. Werden weniger als 9 Parameter benötigt, werden die restlichen Parameter mit 0 aufgefüllt. Das Gateway schickt das Kommando und die Parameter an die Nockensteuerung, das mit Byte 1 (ID) selektiert wird.

2.3.6.6 Gateway zum DeviceNet-Master

Es werden vom Gateway an den DeviceNet-Master folgende 13 Byte als Antwort auf eine Anfrage (s. o) gesendet:

Byte	Bezeichnung	Bedeutung
1	ID (0..15)	Nockensteuerung-ID im DICNET
2	Command	Kommando gemäß Kapitel „Befehlstabelle“
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	
12	Errorcode (High)	Siehe Kapitel „Fehlermeldungen“
13	Errorcode (Low)	

Die Bedeutung der Parameter ist nachfolgend im Kapitel des jeweiligen Kommandos beschrieben. Werden weniger als 9 Parameter zurückgeliefert, werden die restlichen Parameter mit 0 aufgefüllt. Das Gateway antwortet mit der ID und dem Kommando aus dem Anfragetelegramm.

2.3.6.7 Fehlermeldungen

Die Byte 12 und 13 der Parameterantwort repräsentieren einen Errorcode.
Ist der Errorcode kleiner als 8000H, handelt es sich um eine Fehlermeldung der DeviceNet-Kommunikation gemäß „DeviceNet Specification Release 2.0“.
Ist der Errorcode größer als 8000H (MSB gesetzt), handelt es sich um eine allgemeine Fehlermeldung des Gateways. Diese Fehler sind im Anhang im Kapitel „Fehlercodes“ beschrieben. Allerdings muß das MSB des Fehlercodes vorher ausmaskiert werden (8000H subtrahieren).

Errorcode = 0: Kein Fehler
Errorcode < 8000H: Fehler gemäß DeviceNet Specification
Errorcode > 8000H: Fehlercode = Errorocode - 8000H gemäß Anhang

Der Fehler muß gemäß Kapitel "Fehlerquittung", Seite 22 bestätigt werden!

2.3.6.7.1 Beispiel

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) von Programm 3 (Para1) und Ausgang 5 (Para2).

Antwort von Gateway an 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

Einschalttotzeit = Ausschalttotzeit = 9ms, Error Code = 0

2.3.6.8 Prozessdaten

Da das Gateway immer nur die Daten von einem der maximal 16 gleichzeitig angeschlossenen Nockensteuerungen dem DeviceNet-Master senden kann, muß dieser dem Gateway vorher mitteilen, welche Nockensteuerung selektiert werden soll. Das erfolgt über den Parameterdatensatz (Polling) mit dem Kommando „SET_GATEWAY_ID“. Diese Verbindung bleibt dann bis zum nächsten SET_GATEWAY_ID -Kommando erhalten.
Sollen beispielsweise vom Gateway die Prozessdaten der Nockensteuerung mit der ID 3 übertragen werden, muß der DeviceNet-Master folgende Sequenz senden:

DeviceNet-Master -> Gateway:

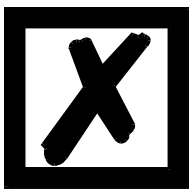
ID	Cmd	Para1	Para2	Para3	Para4	Para4	Para6	Para7	Para8	Para9
0	0x1C	3	0	0	0	0	0	0	0	0

Das 1. Byte (ID = 0) wird in diesem Fall vom Gateway nicht ausgewertet, da dieses Kommando lokal vom Gateway verarbeitet wird. Im 1. Byte kann bei diesem Kommando somit jeder beliebige Wert stehen.

Antwort von Gateway an DeviceNet-Master:

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

Liefert das Gateway als 1. Parameter 0 zurück, ist die selektierte Nockensteuerung im DICNET vorhanden, ansonsten steht im 1. Parameter eine 1.



Nach dem Einschalten des Gateways wird automatisch die Nockensteuerung mit dem gleichen ID selektiert, der als MAC-ID am DIP-Switch eingestellt ist, wobei nur die unteren 4 Bit ausgewertet werden!

2.3.6.9 DeviceNet-Master zum Gateway

Vom Master an das Gateway werden keine Prozessdaten übertragen.

2.3.6.10 Gateway zum DeviceNet-Master

Es werden vom Gateway an den DeviceNet-Master 8 Byte als Antwort auf eine Bit-Strobe-Anfrage oder eigenständig bei Änderung der Daten gesendet.
Dabei muß unterschieden werden zwischen

- Prozessdaten ohne Fehlermeldung
- Prozessdaten mit Fehlermeldung gemäß DeviceNet-Specification
- Prozessdaten mit Fehlermeldung des Gateways

2.3.6.11 Prozessdaten ohne Fehlermeldung

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

2.3.6.12 Prozessdaten mit DeviceNet-Fehlermeldung

Byte 1 und 2 enthalten den DeviceNet-Errorcode, Byte 3 = 0x80, Byte 4 = 0x09.
Die Beschreibung des DeviceNet-Errorcodes ist in der „DeviceNet-Specification“ enthalten.

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

Der Fehler muß gemäß Kapitel Abbildung 2.3.6.14, „Fehlerquittung,“ auf Seite 22 bestätigt werden!

2.3.6.13 Prozessdaten mit Gateway-Fehlermeldung

Byte 3 = 0x80, Byte 4 = Errorcode.
Die Beschreibung der Errorcode sind im Kapitel „Fehlercodes“ im Anhang enthalten.

1.Byte	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

Der Fehler muß gemäß Kapitel "Fehlerquittung", Seite 22 bestätigt werden!

2.3.6.13.1 Beispiel

Die vorher mit SET_GATEWAY_ID selektierte Nockensteuerung liefert folgend Parameter:

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

Position: 123, Geschwindigkeit 14, Ausgang 1 und 10 eingeschaltet.

2.3.6.14 Fehlerquittung

Alle Fehlermeldungen müssen über den Parameter-Kanal mit dem Kommando ’SET_ERROR_QUIT’ quittiert werden.

Dazu sendet der DeviceNet-Master folgendes Telegramm:

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

Da dieses Telegramm lokal vom Gateway ausgewertet wird, kann jeder beliebige ID im Byte 1 angegeben werden.

2.3.7 Protokoll-Rahmen für ETHERNET

Das Gateway arbeitet als TCP-IP-Server und schreibt eine Anfrage automatisch an den Ethernet Client zurück.

2.3.7.1 Daten vom Client zum Server (Gateway)

Jede Anfrage hat folgendes Format:

Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge	Byte 3 (incl) bis letzter Parameter (incl)
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
...		

Es werden vom Ethernet Client an das Gateway nur die Parameter, die mit dem Längenbyte spezifiziert werden, gültig. (Gültige Parameter = Länge - 2).
Der Antwortrecord des Gateways ist abhängig von dem Kommando.

2.3.7.2 Daten vom Server (Gateway) zum Client

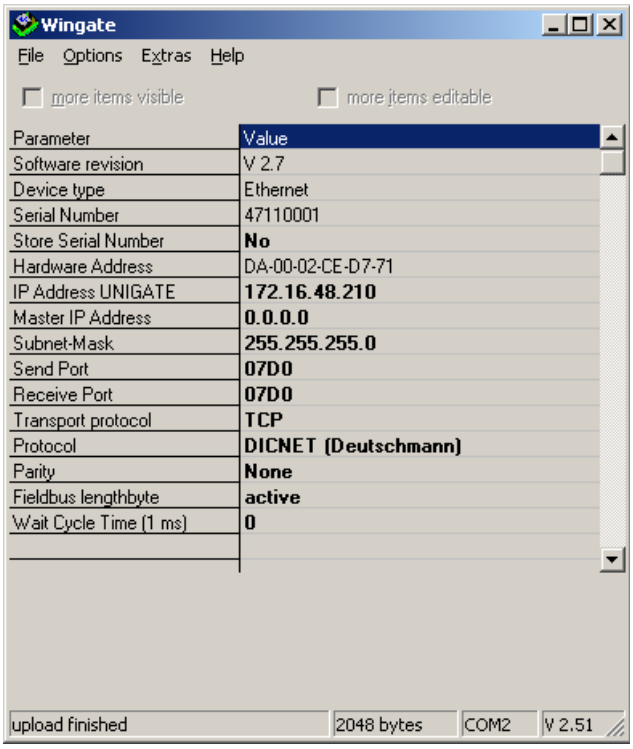
Byte-Nr.	Bezeichnung	Bedeutung
1	Ctrl-K (0B Hex)	Schlüsselzeichen
2	Länge oder Errorcode	Byte 3 (incl) bis letzter Parameter (incl) oder Fehlernummer, wenn Wert > 127
3	ID (0..15)	Nockensteuerung-ID
4	Command	Kommando gemäß Befehltabelle
5	Parameter 1	
6	Parameter 2	
7	Parameter 3	
8	Parameter 4	
9	Parameter 5	
...		

Hier sind ebenfalls wie bei der Anfrage nur die mit dem Längenbyte spezifizierten Parameter gültig.
Der Antwortrecord ist dann gültig, wenn die Bytes 1, 3 und 4 mit den entsprechenden Bytes im Anfragerecord identisch sind.
Tritt bei der Anfrage bei der Nockensteuerung ein Fehler auf, wird der entsprechende Fehler im Byte 2 (anstatt der Länge) übertragen. Der Fehlerwert ist immer größer als 127 und kann dem Kapitel "Fehlerbehandlung" entnommen werden.

2.3.7.3 Beispiel mit dem DEUTSCHMANN Ethernet-Starterkit

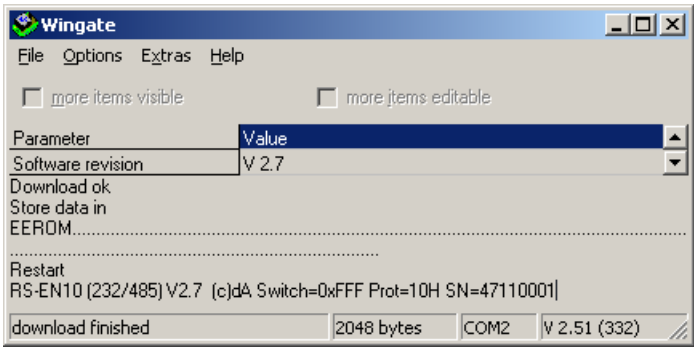
Konfiguration des UNIGATE Ethernet über WINGATE
Setzen Sie das UNIGATE in den Konfigmode:
S4 + S5 = "FF"
Interfaceschalter auf 232

Gerät starten.

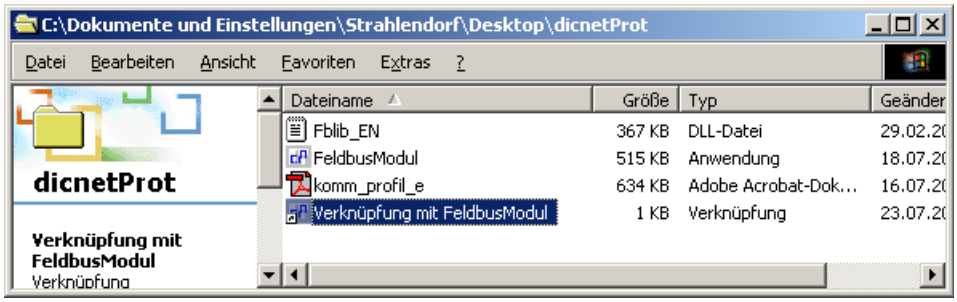


Konfiguration des UNIGATE Ethernet über WINGATE

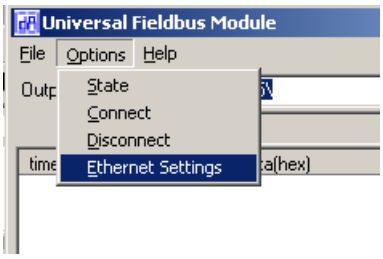
File - Download



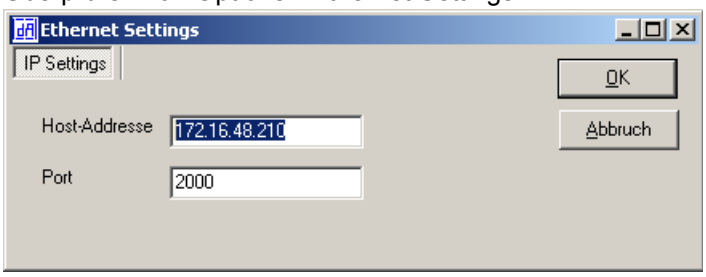
S4 + S5 auf "00" stellen
Interface Schalter auf 485
DICNET verbinden
UNIGATE starten (24V)



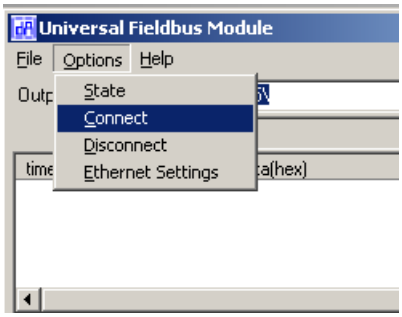
"Verknüpfung mit FeldbusModul" starten



Überprüfen von Options - Ethernet Settings



Die HostAdresse "172.16.48.210" muss mit der in WINGATE übereinstimmen.
Lassen Sie sich hierzu von Ihrem System-Administrator eine freie IP Adresse geben



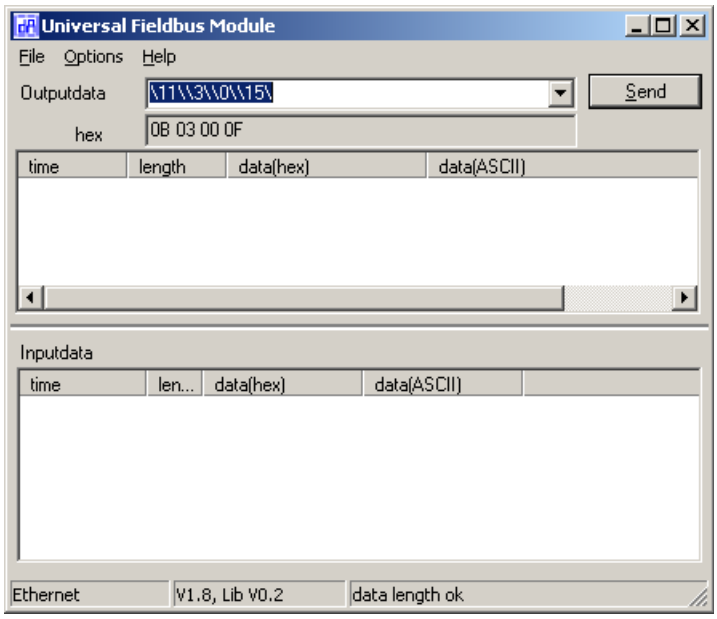
Client - Server - Verbindung herstellen mit Options - Connect

Bei fehlerhafter Verbindung bekommen Sie nach ca. 20 Sekunden folgende Meldung:

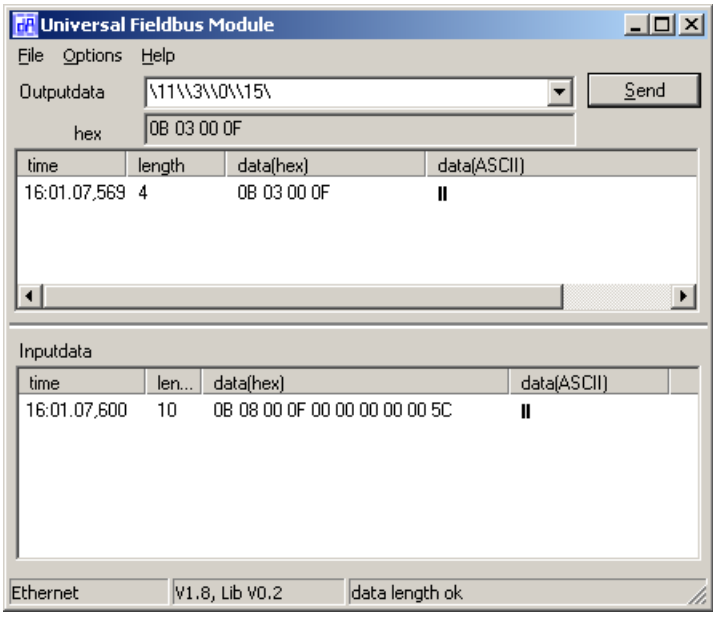
Die BUS-LED blinkt rot-grün im Wechsel

Ist die Verbindung in Ordnung wechselt die BUS-LED in statisch grün.

Geben Sie nun ein Kommando in die Eingabezeile ein:
\11\\3\\0\\15\

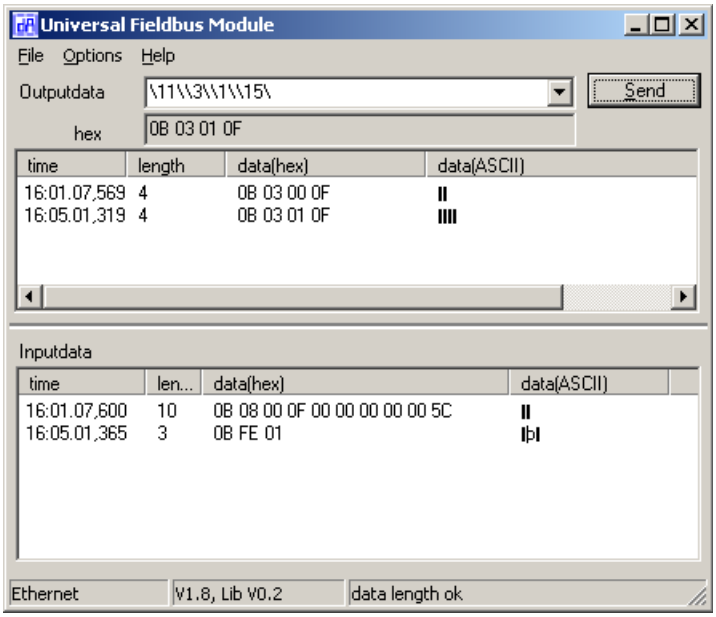


und klicken auf „Send“



Im Input Fenster bekommen Sie ein korrekt beantwortete Antwort für das Kommando "**GET_DISPLAY**" vom DICNET Teilnehmer mit der ID 0 (3. Byte im Sende und Empfangs-String) zurück.

Hier ein Beispiel für einen Teilnehmer mit ID 1 der nicht vorhanden ist:



2.3.8 Befehlstabelle

Befehlsname	Befehlswert	Bedeutung siehe Kapitel
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

Weitere Befehle, die in aktuellen Produkten nicht mehr unterstützt werden sind im Anhang beschrieben.

2.3.9 Parametertabelle

Diese Parametertabelle wird von den Befehlen GET_PARAMETER und SET_PARAMETER verwendet.

Befehlsname	Befehlswert	Bedeutung	Erläuterung
PNR_SOFT_REV	0x0001	see PNR_HARD_REV	
PNR_HARD_REV	0x0002	ASCII i. e.: '3"1"2"t' = V3.12t - gibt den Soft- bzw. Hardware Versionsstand zurück	
PNR_UNIT_NAME	0x0003	ASCII z. B: 'L'4"8" ' = L48	
PNR_UNIT_TYP	0x0004	Gerätetyp	
PNR_VNUMBER	0x0005	Artikelnummer	
PNR_SN	0x0006	Seriennummer	
PNR_OPTION X	0x0007	Option X	
PNR_ENCODER_TYP	0x0010	Gebertyp	Kapitel 2.3.9.1
PNR_RESOLUTION_PER_TURN	0x0011	Real-Auflösung pro Umdrehung	Kapitel 2.3.9.2
PNR_NUMBER_OF_TURNS	0x0012	Real-Anzahl Umdrehung	
PNR_SCALED_ENCODER_RES	0x0013	Virtueller Geberwert	
PNR_ENCODER_INVERT	0x0014	Drehrichtungsumkehr	Kapitel 2.3.9.3
PNR_SCALED_COUNT_RANGE	0x0017	Virtueller Zählbereich	
PNR_COUNT_RANGE	0x0018	Zählbereich bei Ink-Gebern	
PNR_COUNT_RESTORE_VALUE	0x0019	Bei X 16:= Bremspunkt	
PNR_TIMEBASE	0x001C	Zeitbasis bei Timer	
PNR_DEADTIME_BASE_US	0x001D	Zeiteinheit für TZK in µs (wenn nicht definiert -> 1000µs)	
PNR_NUMBER_OUTPUTS	0x0020	Anzahl Ausgänge	
PNR_NUMBER_LOCK_OUTPUTS	0x0021	Anzahl verriegelte Ausgänge	
PNR_NUMBER_DATA_RECORDS	0x0022	Anzahl Datensätze	
PNR_NUMBER_LOGIC_INPUTS	0x0023	Anzahl Logik Eingänge	
PNR_NUMBER_ANGLE_TIME	0x0024	Anzahl WZ-Ausgänge ab Ausgang 1	
PNR_NUMBER_OUTNAME_CHAR	0x0025	Ausgangsnamen	
PNR_NUMBER_PROGRAMS	0x0026	Anzahl Programme	
PNR_NUMBER_AXIS	0x0027	Anzahl Achsen	
PNR_NUMBER_ANALOGOUTPUT	0x0028	Anzahl Analog Ausgänge	
PNR_NUMBER_COUNTER_CAM	0x0029	Anzahl Zählnocken	
PNR_FIRST_OUTPUT_NR	0x002A	Zählung beginnt bei 1	
PNR_SPEED_SCALE	0x0030	Bezogen auf U/msec => 60000 = U/min 0...9999000 (Umdr./mSek)	
PNR_LANGUAGE	0x0031	Sprache	Kapitel 2.3.9.4
PNR_DEADTIME_TYP	0x0032	TZK-Typ	Kapitel 2.3.9.5
PNR_ZEROPOINT_OFFSET	0x0033	Presetwert bei Ink. / Abs.: Virtueller Wert	
PNR_ACTIV_PROGNR	0x0034	Aktives Programm	0..max Programm -1
PNR_ACTIV_AXIS	0x0035	Aktive Achse	1..max AchsNr.
PNR_CALC_SPEED_START	0x0036	TotStart skaliert	
PNR_CALC_SPEED_STOP	0x0037	TotStop skaliert	
PNR_DICNET_ID	0x0038	Tatsächl. Wert (NS= 80..95), RS232 = 232	
PNR_CLEAR_LENGTH	0x0039	Länge Clearimpuls	
PNR_BREAK_PARA	0x003A	(BremsA*0x10000)+BremsB	
PNR_OUTPUT_OFF_SPEED	0x003B	Geschwindigkeits-Schwellenwert unterhalb dem die Ausgänge abgeschaltet werden	
PNR_WZ_MAXTIME	0x003C	Zeit in ms	
PNR_WZ_TIMEBASE	0x003D	Zeit in µs	
PNR_V_LIMIT	0x003E	M13 = 1,wenn V_LIMIT überschritten	
PNR_DREHSCHALTER	0x003F	Schalterstellung lesen	
PNR_RESTART	0x004E	Warmstart mit Wert 0x1234	
PNR_CLEAR_EEROM	0x004F	Generallöschung:1: 0x1234 -> 2:~0x1234	
PNR_STATUS_FLAGS	0x0050		
PNR_PROC_OUT_MAPPING	0x0051	Mapping der Prozeßdaten im Feldbus	
PNR_PROC_IN_MAPPING	0x0052	Mapping der Prozeßdaten im Feldbus	
PNR_USED_EEROM_LEN	0x0053	Tatsächlich genutzte EEROM Länge	
PNR_S7_MODE	0x0054	1 = S7 keine Daten ins EEROM kopieren; 0xFF = kein PB	
PNR_RESET_EEROM	0x0055	Auf Werkseinstellung setzen 1:0x1234 -> 2:~0x1234	
PNR_CYCLETIME	0x0056	Zykluszeit lesen	
PNR_AKTIV_STATUS	0x0057	siehe Kapitel 2.3.11	
PNR_PROC_LOAD	0x0058	Prozessorauslastung	
PNR_ENABLE_OPTION	0x0059	Freischaltung von Optionen	
PNR_TEACH_IN_ZEROPOINT	0x005A	Teach-In Nullpunktverschiebung	
PNR_ENABLE_TESTMODE	0x005B	Mit 0x1234 -> Umschaltung in Testmode	
PNR_DATA_NOT_IN_EEROM	0x005C	Nocken, Totzeiten werden im RAM gespeichert (flüchtig)	
PNR_SCALED_NR_OF_TURNS	0x0015	Virtuelle Anzahl Umdrehungen	
PNR_ZEROPOINT_OFFSET_REAL	0x005D	Realer Offset-Wert	
PNR_DYN_ZEROPOINT_OFFSET	0x005E	Dynamischer Offset (nur IO8)	
PNR_INTERFACE	0x005F	0=RS232; 1=DICNET; ohne Busabschluss; 3= DICNET mit BA	
PNR_ERROR_QUIT	0x0060	Error Quit über Modbus 0 -> 1 (nur LOCON 100-MB)	

2.3.9.1 PNR_ENCODER_TYP - Gebertyp

1 = Absolutwertgeber Parallel Gray
2 = Inkrementalgeber
3 = Absolutwertgeber SSI Gray
5 = Timer
6 = Multiturn-SSI
7 = Inkremental 24-Bit

2.3.9.2 PNR_RESOLUTION_PER_TURN

Absolut parallel Gray: 360, 512, 720, 1000, 1024, 2048, 3600, 4096
SSI Gray: 360, 1024, 4096
Inkremental: 1024 (16), 4096 (17)

2.3.9.3 PNR_ENCODER_INVERT

0 = Normal
1 = Invertiert

2.3.9.4 PNR_LANGUAGE - Sprachauswahl

0 = Deutsch
1 = Englisch
2 = Französisch
3 = Italienisch
4 = Spanisch
5 = Flämisch
6 = Niederländisch
7 = Schwedisch
8 = Finnisch
9 = Dänisch

2.3.9.5 PNR_DEADTIME_TYP

Wegzeitabhängige TZK (Standard bei Deutschmann NS)
0 = Keine
1 = Blockweise
2 = Bitweise
3 = Blockweise, getrennte Ein- und Ausschalttotzeit
4 = Bitweise, getrennte Ein- und Ausschalttotzeit

Zeitabhängige TZK (nur ROTARNOCK 100, LOCON 100)
Wert = Totzeitart aus obiger Tabelle + 10
-> Zeitabhängige bitweise TZ = 2+10 = 12

Direkte TZK (nur ROTARNOCK 100, LOCON 100)
Wert = Totzeitart aus obiger Tabelle + 20
-> Direkte bitweise ON/OFF TZ = 24

2.3.10 Bitnummer des Parameters PNR_STATUS_FLAGS

Befehlsnummer	Befehlswert	Bedeutung
BITNR_ENCODER_INVERT	0x00	Drehrichtungsumkehr
BITNR_SOFT_RUNCONTROL	0x01	Potentialfreier Störmeldeumschaltkontakt
BITNR_UPDOWNLOAD_ENABLE	0x02	Up-Download freigegeben
BITNR_UPDATE_DEP_TURN	0x03	Richtungsnocke möglich
BITNR_ENCODER_TEST	0x04	Geberüberwachung
BITNR_SPEED_OUT_HARD	0x05	Geschw.Ausgang 0..255 an höchstens 8 Ausgängen
BITNR_ANALOGOUT_EXIST	0x06	Analogausgang vorhanden
BITNR_ENCODER_SCALABLE	0x07	Geberauflösung einstellbar
BITNR_BINAER_UP_DOWNLOAD	0x08	Erkennung Datenübertragungsverfahren

2.3.11 Bitnummern des Parameters PNR_AKTIV_STATUS

MASK_STATUS_BITW_TZK	0x01	Typ der Totzeitkompensation
MASK_STATUS_BLOCKW_TZK	0x02	
MASK_STATUS_BITW_EA_TZK	0x04	
MASK_STATUS_BLOCKW_EA_TZK	0x08	
MASK_STATUS_RICHT_NOCK	0x10	Richtungsnocken aktiv
MASK_STATUS_WZ	0x20	Winkel-Zeit-Nocken aktiv
MASK_STATUS_LOGIK	0x40	Logik aktiv

2.4 Fehlerbehandlung

Folgende Fehlermeldungen können bei der Kommunikation erzeugt werden:

Fehlernummer	Bedeutung/Fehlerbehandlung
249	Fehler bei Zugriff auf interne Erweiterungskarte
250	Überlauf Ausgangspuffer
251	Letzter Befehl noch nicht beendet
252	Unbekanntes Kommando
253	Checksum- oder Längen-Error
254	Sonstiger Fehler bei Kommunikation (z. B. Teilnehmer nicht Online)

Der Fehlercode wird anstelle der Netz-ID als 3. Byte im Antwort-Datenrecord übertragen, sofern die Netz-ID vorhanden ist. Tritt kein Fehler auf, beinhaltet dieses Byte einen Wert zwischen 0..128.
Bei Interbus-S wird der Fehlercode anstatt des Befehls im Byte 1 übertragen.
In allen anderen Fällen wird die Fehlernummer im Feld „Error Nummer“ übertragen.

2.5 Befehlsbeschreibungen

2.5.1 Parameter vom NS empfangen

2.5.1.1 GET_OUTPUT

Parameter zum NS: 1. „Offset Ausgangsblöcke“

Parameter vom NS: 1. Ausgangsbelegung 1..8 + („Offset Ausgangsblöcke“ * 8)
2. Ausgangsbelegung 9..16 + ...

Bemerkung: Es werden von der NS immer die Zustände aller Ausgänge übertragen. Handelt es sich beispielsweise um ein Geräte mit 32 Ausgängen, werden 4 Byte übertragen.

Wird „Offset Ausgangsblöcke“ nicht übertragen entspricht das einem Offset von 0.

2.5.1.2 GET_INPUT

Parameter zum NS: 1. „Offset Eingangsblöcke“

Parameter vom NS: 1. Eingangsbelegung 1..8 + („Offset Eingangsblöcke“ * 8)
2. Eingangsbelegung 9..16 + ...

Bemerkung: Es werden von der NS immer die Zustände aller Eingänge übertragen. Handelt es sich beispielsweise um ein Geräte mit 32 Eingängen, werden 4 Byte übertragen.

Wird „Offset Eingangsblöcke“ nicht übertragen entspricht das einem Offset von 0.

2.5.1.3 GET_NEXT_CAM

Parameter zum NS:

1. Programm-Nr (0..Anzahl Programm -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge) MSB = 1 -> Analogausgang
3. Letzter Einschaltpunkt (High-Byte)
4. Letzter Einschaltpunkt (Low-Byte)
5. Letzter Einschaltpunkt (Bit 16..23) **nur bei MT**
6. Flag ERSTE_NOCKE (1=Erste Nocke, sonst 0) **nur MT**

Parameter vom NS:

1. Einschaltpunkt (High-Byte)
2. Einschaltpunkt (Low-Byte)
3. Ausschaltpunkt (High-Byte)
4. Ausschaltpunkt (Low-Byte)
5. Nocken-Nummer (High-Byte)
6. Nocken-Nummer (Low-Byte)
7. Einschaltpunkt (Bit 16..23) **nur bei MT**
8. Ausschaltpunkt (Bit 16..23) **nur bei MT**

Bemerkung: Soll die erste Nocke gelesen werden, wird als "Letzter Einschaltpunkt" der Wert ERSTE_NOCKE übergeben.

Ist keine Nocke mit einem höheren Einschaltpunkt als "Letzter Einschaltpunkt" vorhanden, wird von der NS ERSTE_NOCKE als neuer Einschaltpunkt zurückgeliefert.

Der Parameter "Nocken-Nummer" wird benötigt, wenn diese gelesene Nocke abgeändert werden soll.

Weichen skalierte und reale Geberauflösung voneinander ab, werden skalierte Ein- und Ausschaltpunkte übertragen

Wenn der Ausschaltpunkt größer 0x8000 ist, dann entspricht das der Einschaltzeit minus 0x 8000 bei WZ (Zeitbasis berücksichtigen!)

2.5.1.4 GET_BACK_CAM

Parameter zum NS:

1. Programm-Nr (0..Anzahl Programm -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge) MSB = 1 -> Analog
3. Letzter Einschaltpunkt (High-Byte)
4. Letzter Einschaltpunkt (Low-Byte)
5. Letzter Einschaltpunkt (Bit 16..23) **nur bei MT**
6. Flag ERSTE_NOCKE (1=Erste Nocke, sonst 0) **nur MT**

Parameter vom NS:

1. Einschaltpunkt (High-Byte)
2. Einschaltpunkt (Low-Byte)
3. Ausschaltpunkt (High-Byte)
4. Ausschaltpunkt (Low-Byte)
5. Nocken-Nummer (High-Byte)
6. Nocken-Nummer (Low-Byte)
7. Einschaltpunkt (Bit 16..23) **nur bei MT**
8. Ausschaltpunkt (Bit 16..23) **nur bei MT**

Bemerkung: s. GET_NEXT_CAM

2.5.1.5 GET_IDLETIME

Parameter zum NS:
1. Programm-Nr (0..Anzahl Programm -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge)

Parameter vom NS:
1. Einschalttotzeit (High-Byte)
2. Einschalttotzeit (Low-Byte)
3. Ausschalttotzeit (High-Byte)
4. Ausschalttotzeit (Low-Byte)
5. Ausgangsupdate(0=Immer,1=Pos.,2=Neg.Dreh)
6. Nock Count (0...255)

Bemerkung: Die Ausschalttotzeit wird nur übertragen, wenn es sich um eine NS mit getrennter Ein-/Ausschalt-Totzeit handelt.

Der Parameter 5 wird nur übetragen, wenn es sich um eine NS mit drehrichtungsabhängigen Ausgängen (s. GET_CONFIG_PARA) handelt. In diesem Fall wird auch immer eine Ausschalttotzeit übertragen. Die Einheit der Totzeiten ist über den Parameter PNR_DEADTIME_BASE_US auslesbar.

Der Parameter 6 wird nur bei aktiven Zählnocken übertragen und beinhaltet die Anzahl Zyklen, in denen der Ausgang disabled ist.

2.5.1.6 GET_POSITION

Parameter zum NS: Keine

Parameter vom NS:
1. Aktuelle Geberposition (High)
2. Aktuelle Geberposition (Low)
3. Aktuelle Geberposition (Bit 16..23) **nur MT**

Bemerkung: Weichen skalierte und reale Geberauflösung voneinander ab, wird die skalierte Geberposition übertragen.

2.5.1.7 GET_SPEED

Parameter zum NS: Keine

Parameter vom NS:
1. Aktuelle Gebergeschwindigkeit (High)
2. Aktuelle Gebergeschwindigkeit (Low)
3. Ink/10 ms (High)
4. Ink/10 ms (Low)

Bemerkung: Die Parameter 1 und 2 werden über 2 Sekunden, die Parameter 3 und 4 über 10 ms gemittelt.

2.5.1.8 GET_STATUS

Parameter zum NS: Keine

Parameter vom NS:
1. Aktuelle Error-Nummer (0 = OK)
2. Aktuelles Programm, das abgearbeitet wird
3. Aktive Achse
4. Statusbyte
   Bit 0 = Programm Enable
   Bit 1 = Passwortabfrage enabled
   Bit 2-7 = undefiniert

2.5.1.9 GET_OUT_POS

Parameter zum NS:
1. Skalierte Geberposition (High)
2. Skalierte Geberposition (Low)
3. Skalierte Geberposition (Bit 16..23) (nur MT)
4. Ausgangsblock-Offset (wenn vorhanden) s. GET_OUTPUT

Parameter vom NS:
1. Ausgangsbelegung 1..8 + (Ausgangsblock -Offset * 8)
2. Ausgangsbelegung 9..16 + ...

Bemerkung: Es werden von der NS immer die Zustände aller Ausgänge übertragen. Handelt es sich beispielsweise um Geräte mit 32 Ausgängen, werden 4 Byte übertragen.

Dieser Befehl unterscheidet sich von GET_OUTPUT lediglich dadurch, daß die Ausgangsbelegung einer beliebigen Position angefragt werden kann, und nicht nur die der aktuellen Geberposition.

Die Totzeitkompensation und Logik werden hier nicht berücksichtigt.

2.5.1.10 GET_DISPLAY

Parameter zum NS: Keine

Parameter vom NS:
1. Aktuelle Error-Nummer (0=Ok)
2. Aktuelles Programm, das abgearbeitet wird
3. Aktuelle Gebergeschwindigkeit (High)
4. Aktuelle Gebergeschwindigkeit (Low)
5. Aktuelle skalierte Geberposition (High)
6. Aktuelle skalierte Geberposition (Low)
7. Aktuelle skalierte Geberposition (Bit 16..23) **nur MT**

2.5.1.11 GET_LOGIC

Parameter zum NS:
1. Programm-Nummer (0..Anzahl Programme -1)
2. Ausgangs-Nummer (1..Anzahl Ausgänge)
3. Ausgang-Type (0=Output, 1=Merker)

Parameter vom NS:

1. Ausgang	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I			+	+	+	+	+	Ausgangs-Nr. (1-31)
	I	+	-	-	-	-	-	-	0 = Outp, 1 = Merker
	+	-	-	-	-	-	-	-	0 = Norm, 1 = Invertiert
2. Eingang0	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I	I		+	+	+	+	+	Eingang Nr. 0-31
	+	+	-	-	-	-	-	-	Flankentrigger (s. u.)
3. Eingang1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	+	+	+	+	+	Eing-Nr. (0-31)
	+	+	+	-	-	-	-	-	Verknüpfung (s. u.)
4. Eingang2									**(wie Eingang 1)**
5. Eingang3									**(wie Eingang 1)**
6. Eingangtype	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	I	I	I	+	+	Eing 0 (s. u.)
	I	I	I	I	+	+	-	-	Eing 1 (s. u.)
	I	I	+	+	-	-	-	-	Eing 2 (s. u.)
	+	+	-	-	-	-	-	-	Eing 3 (s. u.)
7. Ausschaltzeit									

Bemerkung:

Flankentrigger:	00	=	Steigende Flanke
	01	=	Fallende Flanke
	10	=	Reserviert
	11	=	Reserviert
Verknüpfungen:	000	=	Keine
	001	=	Oder
	010	=	Und
	011	=	Oder Nicht
	100	=	Und Nicht
	rest	=	Reserve
EingangType:	00	=	Nocken-Ausgang
	01	=	Hardware-Eingang
	10	=	Merker
	11	=	Schieberegisterbit

2.5.1.12 GET_DATA_EXIST

Parameter zum NS:
1. Programm-Nr (0..Anzahl Programme -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge, oder 0xFF)

Parameter vom NS:
1. Nocken vorhanden, wenn > 0
2. Totzeiten vorhanden, wenn > 0

Bemerkung: Es wird geprüft, ob im selektierten Programm und Ausgang Datensätze mit Nocken oder Totzeiten vorhanden sind. (Ist Ausgang = 0xFF) wird das gesamte Programm geprüft.

2.5.1.13 GET_GATEWAY_ID

Parameter zum NS: Keine

Parameter vom NS: ID der selektierten NS (0..15)

Bemerkung: Der ID gibt an, mit welcher Nockensteuerung das Gateway logisch verbunden werden soll. Alle folgenden Kommandos werden daraufhin mit diesem ID ausgetauscht.

Dieses Kommando wird nur in Verbindung mit Feldbus-Gateways benötigt!

2.5.1.14 GET_PARAMETER

Parameter zum NS:
1. Parameter-Nummer (High-Byte)
2. Parameter-Nummer (Low-Byte)
3. Parameter-Typ (0=Akt.Wert, 1=Min.Wert, 2=Max.Wert)

Parameter vom NS:
1. Parameter-Nummer (High-Byte)
2. Parameter-Nummer (Low-Byte)
3. Status (s.u.)
4. Wert (31..24) MSB
5. Wert (23..16)
6. Wert (15..8)
7. Wert (7..0) LSB

Bemerkung:

Status
00(Hex) = Akt.Wert Read-Write
01(Hex) = Akt.Wert Read-Only
10(Hex) = Min.Wert Read-Write
11(Hex) = Min.Wert Read-Only
20(Hex) = Max.Wert Read-Write
21(Hex) = Max.Wert Read-Only
FF(Hex) = Wert existiert nicht

Bemerkung: Mit diesem Befehl lassen sich sowohl alle aktuellen Parameter der NS auslesen, als auch die minimalen und maximalen Wert, die diese Parameter annehmen können (siehe „Parametertabelle“ auf Seite 29), sofern diese vorhanden sind.

2.5.1.15 GET_OUTPUT_NAME

Parameter zum NS:
1. Ausgangs-Nr (1..AnzahlAusgaenge)
2. Pointer auf Ausangsnamen (0..NameLen-1)

Parameter vom NS:
1. Ausgangsname [Ptr] (s. o)
2. Ausgangsname [Ptr+1]
3. Ausgangsname [Ptr+2]
4. Ausgangsname [Ptr+3]
5. Ausgangsname [Ptr+4]
6. Ausgangsname [Ptr+5]
7. Ausgangsname [Ptr+6]
8. Ausgangsname [Ptr+7]

2.5.1.16 GET_GATEWAY_DATA

Parameter zum NS:
1. Ausgangsblock-Offset (s. GET_OUTPUT)

Parameter vom NS:
1. Speed MSB — Aktuelle Gebergeschwindigkeit
2. Speed LSB
3. Pos \ — Aktuelle skalierte Geberposition
4. Pos | 24 Bit
5. Pos LSB /
6. Out 1 - 8 + ((Ausgangsblock-Offset) * 8)
7. Out 9 -16 + ...
8. Out 17 - 23 +...
9. Out 24 - 32 + ((Ausgangsblock-Offset) * 8)

2.5.1.17 GET_EEROM_BLOCK

Parameter zum NS:
1. Para: BlockNr (0-255)
2. Para: BlockLen (1..249) - Empfehlung: 128

Parameter vom NS:
1. Para: EEROM [BlockNr * BlockLen]
2. Para: EEROM [(BlockNr * BlockLen) +1]
:
BlockLen.Para: EEROM [.. + (BlockLen -1)]

2.5.1.18 GET_L2000_DATA

Parameter zum NS:

1. NS Zustand ---------- X X X X X X X X
   | |
   | +----> Prog Enable
   +--------> Output Enable

Parameter vom NS:

1. Fehlernummer
2. Ausgangszustand 1...8
3. Fehlerwert
4. Status ---------- X X X X X X X X
5. Akt. Programm |
   + ----> Not used

2.5.2 Parameter zur NS senden

2.5.2.1 SET_CAM_NEW

Parameter zum NS:

1. Programm-Nr (0..Anzahl Programme -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge) MSB = 1 -> Analogausgang!
3. Einschaltpunkt (High-Byte)
4. Einschaltpunkt (Low-Byte)
5. Ausschaltpunkt (High-Byte)
6. Ausschaltpunkt (Low-Byte)
7. Einschaltpunkt (Bit 16..23) **nur MT**
8. Auschaltpunkt (Bit 16..23) **nur MT**

Parameter vom NS:

1. Quittung (s. Bemerkung)

Bemerkung: Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.

Weicht die reale von der skalierten Geberauflösung ab, werden die skalierten Ein- und Ausschaltwerte übertragen.

Wenn der Ausschaltpunkt größer 0x8000 ist, dann entspricht das der Einschaltzeit minus 0x8000 bei WZ (Zeitbasis berücksichtigen!)

2.5.2.2 SET_IDLETIME

Parameter zum NS:

1. Programm-Nr (0..Anzahl Programme -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge)
3. Einschalttotzeit (High-Byte)
4. Einschalttotzeit (Low-Byte)
5. Ausschalttotzeit (High-Byte)
6. Ausschalttotzeit (Low-Byte)
7. Ausgangsupdate(0=Immer,1=Pos.,2=Neg.Dreh)
8. Nock Count (0...255)

Parameter vom NS:

1. Quittung (0=OK, sonst Fehlernummer)

Bemerkung: Die Ausschalttotzeit wird nur benötigt bei NS mit getrennten Ein- Ausschalt-Totzeiten, andernfalls kann dieser Parameter ersatzlos entfallen. Ist bei einem solchen Geräte die Ein- oder Ausschaltzeit gleich 0, wird der Datensatz gelöscht.*)

Die Zeiteinheit wird von der NS festgelegt (1 ms/0,1 ms). Siehe Parameter PNR_DEADTIME_BASE_US.

Der Parameter 7 wird nur benötigt, wenn es sich um eine NS mit drehrichtungsabhängigen Ausgängen handelt. In diesem Fall muß auch immer die Ausschalttotzeit übertragen werden.

2.5.2.3 SET_ERROR_QUIT

Parameter zum NS: Keine

Parameter vom NS: 1. Quittung (0=OK, sonst Fehlernummer)

Bei Verbindung über RS232 oder Dicnet wird dieses Kommando nicht benötigt, da ein Fehler mit einem einzelnen CR (0DH) bestätigt wird.

2.5.2.4 SET_LOGIC

Parameter zum NS:

1. Programm-Nr.									**(0..Max)**
2. Ausgang	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I			+	+	+	+	+	Ausgangs-Nr. (1-16)
	I	+	-	-	-	-	-	-	0 = Outp, 1 = Merker
	+	-	-	-	-	-	-	-	0 = Norm, 1 = Invertiert
3. Eingang0	**X**	**X**	**0**	**X**	**X**	**X**	**X**	**X**	
	I	I		+	+	+	+	+	Eingang Nr. 0-31
	+	+	-	-	-	-	-	-	Flankentrigger (s.u.)
4. Eingang1	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	+	+	+	+	+	Eing-Nr. (0-31)
	+	+	+	-	-	-	-	-	Verknüpfung (s. u.)
5. Eingang2									**(wie Eingang 1)**
6. Eingang3									**(wie Eingang 1)**
7. Eingangtype	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	
	I	I	I	I	I	I	+	+	Eing 0 (s. u.)
	I	I	I	I	+	+	-	-	Eing 1 (s u.)
	I	I	+	+	-	-	-	-	Eing 2 (s. u.)
	+	+	-	-	-	-	-	-	Eing 3 (s. u.)
8. Ausschaltzeit									

Parameter vom NS: 1. Quittung (s. Bemerkung)

Bemerkung: Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.

Soll der Logikrecord gelöscht werden, muß sowohl für den Eingang 0 als auch die Ausschaltzeit der Wert 0 x FF übergeben werden.

Flankentrigger:
- 00 = Steigende Flanke
- 01 = Fallende Flanke
- 10 = Reserviert
- 11 = Reserviert

Verknüpfungen:	000	=	Keine
	001	=	Oder
	010	=	Und
	011	=	Oder Nicht
	100	=	Und Nicht
	rest	=	Reserve
EingangType:	00	=	Nocken-Ausgang
	01	=	Hardware-Eingang
	10	=	Merker
	11	=	Schieberegisterbit

2.5.2.5 SET_CAM_MOVE

Parameter zum NS:
1. Programm-Nr (0..Anzahl Programme -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge) MSB = 1 -> Analogausgang
3. Anzahl Inkremente (High-Byte)
4. Anzahl Inkremente (Low-Byte)
5. Anzahl Inkremente (Bit 16..23) **nur MT**

Parameter vom NS:
1. Quittung (s. Bemerkung)

Bemerkung: Programmtechnisch bedingt ist die Quittung der NS immer 0.

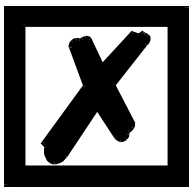
2.5.2.6 SET_CAM_CHANGE_SHORT

Parameter zum NS:
1. Einschaltpunkt (High-Byte)
2. Einschaltpunkt (Low-Byte)
3. Ausschaltpunkt (High-Byte)
4. Ausschaltpunkt (Low-Byte)
5. Nocken-Nummer (High-Byte)
6. Nocken-Nummer (Low-Byte)

Parameter vom NS:
1. Quittung (s. Bemerkung)

Bemerkung: Die Nocken-Nummer muß identisch sein mit der Nummer, die von der NS übertragen wurde bei der Anfrage GET_NEXT_NOCKE oder GET_BACK_NOCKE. Diese Nocken-Nummer definiert auch das Programm und den Ausgang.

LÖSCHEN: Ein = Aus! (Bei WZ: Ein = Aus $\geq$ 0x8000)



Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.
Weicht die reale von der skalierten Geberauflösung ab, werden die skalierten Ein- und Ausschaltwerte übertragen.

2.5.2.7 SET_GATEWAY_ID

Parameter zum NS:
1. Neuer ID (0..15)

Parameter vom NS:
1. Quittung (0=Ok, 1=ID nicht im Netz)

Bemerkung: Der ID legt fest, mit welcher Nockensteuerung das Gateway logisch verbunden werden soll. Alle folgenden Kommandos werden daraufhin mit diesem ID ausgetauscht.

Bei LOCON 2000: ID-Vergabe für interne Kommunikation

2.5.2.8 SET_CAM_CHANGE_MT (nur Multiturn-Geräte)

Parameter zum NS:
1. Einschaltpunkt (High-Byte)
2. Einschaltpunkt (Middle-Byte)
3. Einschaltpunkt (Low-Byte)
4. Ausschaltpunkt (High-Byte)
5. Ausschaltpunkt (Middle-Byte)
6. Ausschaltpunkt (Low-Byte)
7. Nocken-Nummer (High-Byte)
8. Nocken-Nummer (Low-Byte)

Parameter vom NS:
1. Quittung (s. Bemerkung)

Bemerkung: Die Nocken-Nummer muß identisch sein mit der Nummer die von der NS übertragen wurde bei der Anfrage GET_NEXT_NOCKE oder GET_BACK_NOCKE.

Die gesendeten Ein- und Ausschaltpunkte beziehen sich ebenfalls auf das Programm und den Ausgang des Datensatzes mit der gesendeten Nocken-Nummer.

Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.

Weicht die reale von der skalierten Geberauflösung ab, werden die skalierten Ein- und Ausschaltwerte übertragen. Nocke löschen: Ein = Aus !

2.5.2.9 SET_PARAMETER

Parameter zum NS:
1. Parameter-Nummer (High-Byte)
2. Parameter-Nummer (Low-Byte)
3. Wert (31..24) MSB
4. Wert (23..16)
5. Wert (15..8)
6. Wert (7..0) LSB

Parameter vom NS:
1. Parameter-Nummer (High-Byte)
2. Parameter-Nummer (Low-Byte)
3. Status (s. u.)

Bemerkung:

Status
- 00(Hex) = Ok, Wert geändert
- 01(Hex) = Wert Read-Only, nicht geändert
- 02(Hex) = Keine Programmierfreigabe
- 03(Hex) = EEROM Write Error
- 04(Hex) = Range Error
- 05(Hex) = NS nicht speicherbereit
- 06(Hex) = Befehl wiederholen
- 07(Hex) = Ext. Modulerror
- FF(Hex) = Wert existiert nicht

Bemerkung: Mit diesem Befehl lassen sich alle aktuellen Parameter (siehe „Parametertabelle“ auf Seite 29) verändern.

2.5.2.10 SET_OUTPUT_NAME

Parameter zum NS:
1. Ausgangs-Nr (1..AnzahlAusgaenge)
2. Pointer auf Ausangsnamen (0..NameLen-1)
3. Ausgangsname [Ptr] (s.o)
4. Ausgangsname [Ptr+1]
5. Ausgangsname [Ptr+2]
6. Ausgangsname [Ptr+3]
7. Ausgangsname [Ptr+4]
8. Ausgangsname [Ptr+5]

Parameter zum NS:
1. Quittung (0=Ok, sonst Fehlermeldung)

2.5.2.11 SET_EEROM_BLOCK

Parameter zum NS:
1. Parameter: Block_Nr. (0 ...255)
2. Parameter: Block Len (1...249) -> Empfehlung: 128
3. Parameter:
   0 = Store in RAM:
   1 = COPY RAM -> EEROM (s.u)
   2 = wie 1, aber nach allen Aktionen - Auto Restart
   3 = Generallöschung - Auslieferungszustand (s.u.)
4. Parameter: EEROM [BlockNr * BlockLen]
5. Parameter: EEROM [(BlockNr * BlockLen)+1]
   :
   BlockLen+ 3.Para: EEROM[(BlockNr * Blocklen) + (BlockLen -1)]

Parameter vom NS:
1. Parameter: Quittung (0 = Ok, sonst error)

Bemerkung: Beim Kopieren vom RAM ins EEROM (3. Parameter = 1) wird immer der gesamte Adressbereich 0 .. ((BlockNr + 1) * BlockLen) -1 übertragen. Das restliche EEROM wird mit 0xFF aufgefüllt.

Zur Einleitung einer Generallöschung (Zurücksetzen im Auslieferungszustand) müssen der 1. Parameter = 0 x 55, der 2. Parameter = 0 x AA und der 3. Parameter = 3 enthalten.

3 Beispiele

3.1 Übertragung mit DICNET, RS232 oder Feldbus

3.1.1 Ausgangszustand eines LOCON 32 lesen

Sendung an NS:

0BH	Konstante Schlüsselzeichen
02H	Länge (Netz-ID + Command)
30H	Eigener Netz-ID (Bei RS232 immer 0)
01H	Command für GET_OUTPUT
33H	Checksumme (Länge XOR Netz-ID XOR Command)

Antwort von NS:

0BH	Konstante Schlüsselzeichen
06H	Länge (Netz-ID + Command) + Parameter
50H	Eigener Netz-ID (Bei RS232 immer 0)
01H	Command für GET_OUTPUT (Echo)
12H	Ausgang 1..8
23H	Ausgang 9..16
34H	Ausgang 17..24
45H	Ausgang 18..32
C7H	Checksumme (Länge XOR ... XOR Ausgang 18..32)

3.1.2 Übertragung mit Interbus-S

3.1.2.1 Nockensteuerung-Typ erfragen

Sendung an NS:

86H	Befehlsnummer mit gesetztem MSB
00H	
00H	Keine Parameter ==> z. B.: mit 00 auffüllen
00H	
00H	
00H	
00H	
00H	

Antwort von NS:

86H	Wiederholung des Befehlscode mit gesetztem MSB
02H	Typnummer (2 = LOCON2)
33H	Software-Version V 3.
30H	0
31H	1 = V3.01
00H	Rest mit 00H aufgefüllt
00H	
00H	

3.2 Muster-Source-Code für Zugriff auf Kommunikationsroutinen

Nachfolgend zeigen wir anhand der Beispieldatei File 8051 den Source-Code für Zugriff auf Kommunikationsroutinen NS.

```
#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. Länge des Para.Blockes inc. Cmd */

#define WAIT_NSW                0xFE /* Dummy-Kommandos für 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 /* Commando-Kennung für ser. Komm. */
#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 /* Allgem. Parameter gem. 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 /* Allgem.Parameter gem. 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 /* Allgemeine NSW-Parameter (32-
                                                   Bit-Werte) */
```

```
#define PNR_SOFT_REV             0x0001 /* s.HardRev */
#define PNR_HARD_REV             0x0002 / *ASCIIz.B:'3''1''2''t'= V3.12t */
#define PNR_UNIT_NAME            0x0003 /* ASCIIz.B:'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 /* Fiktiver Geberwert */
#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 /* Zeitbasis für TZK in 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 /* bezogen auf U/msec => 60000 = U/min */
#define PNR_LANGUAGE             0x0031
#definePNR_DEADTIME_TYP          0x0032
#define PNR_ZEROPOINT_OFFSET     0x0033 /* Skaliert */
#define PNR_ACTIV_PROGNR         0x0034
#define PNR_ACTIV_AXIS           0x0035
#define PNR_CALC_SPEED_START     0x0036 /* TotStart skaliert */
#define PNR_CALC_SPEED_STOP      0x0037 /* TotStop  skaliert */
#define PNR_DICNET_ID            0x0038 /* Tatsächl.Wert(NSW =80..95) */
#definePNR_CLEAR_LENGTH          0x0039
#define PNR_BREAK_PARA           0x003A /* (BremsA * 0x10000) + BremsB */
#define PNR_OUTPUT_OFF_SPEED     0x003B
#define PNR_WZ_MAXTIME           0x003C
#define PNR_WZ_TIMEBASE          0x003D
#define PNR_V_LIMIT              0x003E /* M13=1, wenn V_LIMIT */

#define PNR_RESTART              0x004E /* Warmstart mit Wert 0x1234 */
#define PNR_CLEAR_EEROM          0x004F /* Generallöschung mit 0x1234 */
#define PNR_STATUS_FLAGS         0x0050
#define PNR_PROC_OUT_MAPPING     0x0051 /* Mapping der Prozessdaten im Feldbus */
#define PNR_PROC_IN_MAPPING      0x0052 /* Mapping der Prozessdaten im Feldbus */
#define PNR_USED_EEROM_LEN       0x0053 /* Tatsächlich genutzte EEROM-Laenge */
#define PNR_S7_MODE              0x0054 /* 1=S7=>Kein Daten ins EEROM kopieren */
#define PNR_SCALED_NR_OF_TURNS   0x0015 /* Virtuelle Anzahl Umdrehung */
#define PNR_ENABLE_TESTMODE 1234 0x005B /* = 0x1234 erlaubt Umschaltung in Test-
                                           mode (nur L1, L2, L16 & L17) */
#define PNR_ZEROPOINT_OFFSET_REAL  0x005D /* Realer Offset-Wert */
#define PNR_DYN_ZEROPOINT_OFFSET_  0x005E /* Dynamischer Offset (nur IO8) */
#define PNR_INTERFACE              0x005F /* 0=RS232; 1=DICNET ohne Busabschluss;
                                             2=DICNET mit BA (nur L100 & L200) */
```

```
                                        /* Bitnummers des STATUS_FLAGS */
#define BITNR_ENCODER_INVERT       0x00
#define BITNR_SOFT_RUNCONTROL      0x01
#define BITNR_UPDOWNLOAD_ENBALE    0x02
#define BITNR_UPDATE_DEP_TURN      0x03 /* Drehrichtungsabh.Ausgangsupdate m */
#define BITNR_ENCODER_TEST         0x04 /* Geber */
#define BITNR_SPEED_OUT_HARD       0x05 /* Geschw.Ausgang 0..255 an 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 /* Interner Error von NSW */
#define CMD_UNKNOWN_ERROR          252
#define CMD_CHECKSUM_ERROR         253 /* Checksum-Error von NSW erkannt */
#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 selek-
tierte 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, unsi-
gned 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) + ParaEmpf-
Block[2];
    if (InLen == 2)
      *OffTime = *OnTime;
    else
       *OffTime = ((unsigned int)(ParaEmpfBlock[3]) * 256) + ParaEmpf-
Block[4];
    return(TRUE);
    }
  return(FALSE);
}
```

4 Anhang A

4.1 Befehlstabelle (nicht für neue Anwendungen)

Befehlsname	Befehlswert	Befehlsbedeutung
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 (nur bei Mattenschaltwerken)
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	4.2.2.5
SET_GEAR_PARA	0x19	4.2.2.6
SET_OUTPUT_MATTE	0x32	4.2.2.7 (nur bei Mattenschaltwerken)

4.1.1 Erläuterungen

Um eine höhere Flexibilität für zukünftige Anwendungen zu erreichen wurden Mitte 1997 die Befehle SET_PARAMETER und GET_PARAMETER sowie GET_OUTPUT_NAME und SET_OUTPUT_NAME ergänzt. Mit Hilfe dieser Befehle und der Parametertabelle (siehe Kapitel Parametertabelle) können alle unten aufgeführten Befehle ersetzt werden. Dadurch ergibt sich ein Schnitt bei den nachfolgend aufgeführten Software-Revisionen:

Geräteausführung	Neue Befehle (s. o.) ab Revisionsstand
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 alle Varianten	V3.0
ROTARNOCK 1, ROTARNOCK 2	V4.1
Multiturn-ROTARNOCK	V4.1

Die nachfolgend aufgeführten Befehle sind in allen Versionen der oben angeführten Nockensteuerungen vorhanden, sollten aber in Zusammenarbeit mit Geräten die den neuen Befehlssatz unterstützen, nicht mehr verwendet werden.

4.2 Beschreibung nicht mehr zu verwendender Befehle

4.2.1 Parameter vom NS empfangen

4.2.1.1 GET_TYP

Parameter zum NS: Keine

Parameter vom NS:
1. Typnummer (Codiert von 0..255)
2. Software-Revision 1. Zeichen
3. Software-Revision 2. Zeichen
4. Software-Revision 3. Zeichen

4.2.1.2 GET_MAXPARA

Parameter zum NS: Keine

Parameter vom NS:
1. Anzahl Programme
2. Maximale Ausgangsnummer
3. Maximaler realer Geberwert (High) --- 0 = 65536 !
4. Maximaler realer Geberwert (Low) /
5. Maximale Totzeit (High) in TZ-Einheiten
6. Maximale Totzeit (Low) in TZ-Einheiten
7. Anzahl Achsen
8. Log2 der max. Anzahl Umdrehungen (nur MT)

Bemerkung: Bei einem inkrementalen Geber wird anstelle des max. Geberwertes der maximale Zählbereich übertragen, bei einem Multi-Turn-Geber (MT) die Anzahl Inkremente/Umdrehung.

4.2.1.3 GET_CONFIG_PARA

Parameter zum NS: Keine

Parameter vom NS:
1. Netz-ID (bei RS232 oder IBS ohne Bedeutung)
2. Geberart (1=Abs,2=Ink,3=SSI,4=DSI,5=Timer)
3. Reale Geberauflösung/Zeitbasis (High) --0=65536
4. Reale Geberauflösung/Zeitbasis (Low) /
5. TZK (.1=Block,.2=Bit,.3=Bl-E/A,.4=Bit-E/A)
6. Anzahl gesperrter Ausgänge
7. Log2 der Anzahl Umdrehungen (nur bei MT)
8. Sonderkonfiguration

X	X	X	X	X	X	X	X		
I	I	I	I	I	I	I	+	1	= Drehrichtungsabhängige Ausgänge
I	I	I	I	I	+	+	-	0-3	= Anz. Logikbl. à 8 Ausgänge
I	I	I	I	I	+	+	-	0-3	= Anz. Logikbl. à 8 Ausgänge
I	I	I	I	+	-	-	-	1	= Getriebefunktion vorhanden
I	I	I	+	-	-	-	-	1	= Analogausgang vorhanden
+	+	+	-	-	-	-	-	0	(Reserve)

Bemerkung: Die Parameter 7 und 8 werden nicht bei allen NS übertragen!

Bei einem Analoggeber wird anstatt der Geberart der Ausgangstyp des Analoggebers übertragen.

4.2.1.4 GET_SYSTEM_PARA

Parameter zum NS: Keine

Parameter vom NS:
1. Analog-Ausgangsfaktor (High) Geschw. bei 10V (nur L32)
2. Analog-Ausgangsfaktor (Low) Geschw. bei 10V (nur L32)
3. Fikt.NullpunktverschiebungZählbereich (H) 0=65536
4. Fikt.Nullpunktverschiebung/Zählbereich (Low)

5a. Nullpunktverschiebung/Zählbereich (Bit 16..23) **nur MT**
5b. Drehrichtung (0=pos., 1=neg.) **sonst**
6a. Sprache (0=Dt., 1=Eng., 2=Frz, 3=It., 4=Sp.)
6b. Siehe Bemerkung

7. Geschwindigkeitsanzeigefaktor (High) -> 0... 9999 (Umdr./s.)
8. Geschwindigkeitsanzeigefaktor (Low) -> 0... 9999 (Umdr./s.)

Bemerkung: Bei MT enthält der 6. Parameter die Information für Sprache und Drehrichtung gemäß folgender Formel: Parameter 6 = (Drehrichtung * 16) + Sprache.

4.2.1.5 GET_OUT_NAME

Parameter zum NS: 1. (BlockNr * 64) + Ausgangs-Nr. (s. Bemerkung)

Parameter vom NS:
1. Zeichen 1
2. Zeichen 2
...
8. Zeichen 8

Bemerkung: Um eine möglichst schnelle Übertragung der Ausgangsnamen zu erreichen, wird der Ausgangsname jedes Ausganges in max. 4 Blöcke je 8 Zeichen zerlegt (max. Länge des Namens = 32 Zeichen). Es können somit über den 1. Parameter gezielt jeder der 4 Blöcke von jedem der maximal 64 Ausgänge abgefragt werden.Dabei gelten folgende Zahlenbereiche: BlockNr 0..3, Ausgangs-Nr0..63 (entspricht Ausgang 1-64)

4.2.1.6 GET_PROT_REC

Parameter zum NS:
1. Protokoll-Nummer (High)
2. Protokoll-Nummer (Low)
3. Blocknummer (0..2)
4. Aktion (0=Keine, 1=Einträge löschen, 2=Set Zeit)
5. System-Zeit (MSB)
6. System-Zeit (.....)
7. System-Zeit (.....)
8. System-Zeit (LSB)

Parameter vom NS:
1. Status* (High) (bei Blocknummer=0)
2. Status* (Middle)
3. Status* (Low)
4. Type (0=Datensatz,1=Byte,2=Integer)
5. Tag
6. Monat
7. Stunde
8. Minute

Parameter vom NS: 1.- 8. Alter Datensatz (bei Blocknummer=1)

Parameter vom NS: 1.- 8. Neuer Datensatz (bei Blocknummer=2)

Bemerkung: Die System-Zeit wird von der Nockensteuerung übernommen, wenn das Byte "Aktion" auf 2 gesetzt ist. Sie wird als Long-Integer in der Einheit "10ms" übergeben.

Wird im Byte "Aktion" eine 1 übertragen, werden alle gespeicherten Protokoll-Daten gelöscht.

Die Blocknummer der Anfrage bestimmt, welcher Block des Datensatzes von der Nockensteuerung übertragen wird.*Der Status setzt sich zusammen aus:
(Anzahl Protokolle * 4096) + (Nummer des letzten Protokolls)

4.2.1.7 GET_GEAR_PARA

Parameter zum NS: Keine

Parameter vom NS:
1. Fiktive Geberauflösung (High)
2. Fiktive Geberauflösung (Low)
3. Dezimalpunkt-Position
4. Bezeichnung für Maßeinheit 1.Char
5. Bezeichnung für Maßeinheit 2.Char
6. Bezeichnung für Maßeinheit 3.Char
7. Max. mögliche fiktive Geberauflösung (High)
8. Max. mögliche fiktive Geberauflösung (Low)

4.2.1.8 GET_OUTPUT_MATTE (nur Mattenschaltwerk)

Parameter zum NS:
1. Programm-Nummer (0..Anzahl Programme -1)
2. Fiktive Geberposition (High)
3. Fiktive Geberposition (Low)

Parameter vom NS:
1. Letzte Position (High)
2. Letzte Position (Low)
3. Ausgangsbelegung 1..8
4. Ausgangsbelegung 9..16 (wenn vorhanden)
...

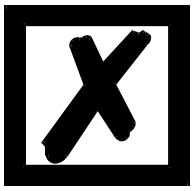
Bemerkung: Es werden von der NS immer die Zustände aller Ausgänge übertragen. Handelt es sich beispielsweise um ein Geräte mit 32 Ausgängen, werden 4 Byte übertragen.

4.2.2 Parameter zur NS senden

4.2.2.1 SET_ACT_PARA

Parameter zum NS:
1. Aktuelles Programm setzen
2. Aktive Achse setzen
3. Status
Bit 0 = Clear All
Bit 1-7 = reserviert (z.Zt = 0)

Parameter vom NS:
1. Quittung (0 = OK, sonst Fehlernummer)



Wird das Bit 0 (LSB) des Statusbytes auf 1 gesetzt, werden von der NS sämtliche Benutzerdaten gelöscht!

4.2.2.2 SET_CAM_CHANGE

Parameter zum NS:
1. Programm-Nr (0..Anzahl Programme -1)
2. Ausgangs-Nr (1..Anzahl Ausgänge)
3. Einschaltpunkt (High-Byte)
4. Einschaltpunkt (Low-Byte)
5. Ausschaltpunkt (High-Byte)
6. Ausschaltpunkt (Low-Byte)
7. Nocken-Nummer (High-Byte)
8. Nocken-Nummer (Low-Byte)

Parameter vom NS:
1. Quittung (s. Bemerkung)

Bemerkung: Die Nocken-Nummer muß identisch sein mit der Nummer die von der NS übertragen wurde bei der Anfrage GET_NEXT_NOCKE oder GET_BACK_NOCKE.

Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.

Weicht die reale von der fiktiven Geberauflösung ab, werden die fiktiven Ein- und Ausschaltwerte übertragen.

4.2.2.3 SET_CONFIG_PARA

Parameter zum NS:
1. Achs-Nr (0..15)
2. Geberart (1 = Abs, 2 = Ink, 3 = SSI, 4 = DSI, 5 = Timer)
3. Reale Geberauflösung (High)
4. Reale Geberauflösung (Low)
5. Totzeitkompensation (1=Block, 2=Bit, 3=Block-Ein-Aus)
6. Anzahl gesperrter Ausgänge
7. Log2 der max. Anzahl Umdrehungen (nur MT)

Parameter vom NS:
1. Quittung (0 = OK, sonst Fehlernummer)

Bemerkung: Bei einem Multi-Turn-Geber wird anstelle des max. Geberwertes die Anzahl der Inkremente/Umdrehung übertragen.

Bei einer RS232-Verbindung wird bei Achs-Nr immer 0 übertragen.

4.2.2.4 SET_SYSTEM_PARA

Parameter zum NS:
1. Analog-Ausgangsfaktor (High) Geschw. bei 10V (nur L32)
2. Analog-Ausgangsfaktor (Low) Geschw. bei 10V (nur L32)
3. Fikt. Nullpunktverschiebung/Zählbereich (High)
4. Fikt. Nullpunktverschiebung/Zählbereich (Low)

5a. Nullpunktverschiebung/Zählbereich (Bit 16..23) **nur MT**
5b. Drehrichtung (0=pos., 1=neg.) **sonst**
6a. Sprache (0 = Dt., 1 = Eng., 2 = Frz., 3 = Ital., 4 = Spn.)
6b. Siehe Bemerkung

7. Geschwindigkeitsanzeigefaktor (High) -> 0... 9999 (Umdr./s.)
8. Geschwindigkeitsanzeigefaktor (Low) -> 0... 9999 (Umdr./s.)

Parameter vom NS:
1. Quittung (0 = OK, sonst Fehlernummer)

Bemerkung: Wurde mit SET_KONFIG_PARA ein Absolutwertgeber konfiguriert, werden die Parameter 3 und 4 als Nullpunktverschiebung interpretiert, andernfalls wird damit der Zählbereich der inkrementalen NS festgelegt (Default 8192).

Bei MT enthält der 6. Parameter die Information für Sprache und Drehrichtung gemäß folgender Formel:

Parameter 6 = (Drehrichtung * 16) + Sprache.

4.2.2.5 SET_OUT_NAME

Parameter zum NS:
1. (BlockNr * 64) + Ausgangs-Nr. (s. Bemerkung)
2. Zeichen 1
3. Zeichen 2
...
9. Zeichen 8

Parameter vom NS:
1. Quittung (0 = OK, sonst Fehlernummer)

Bemerkung: Um eine möglichst schnelle Übertragung der Ausgangsnamen zu erreichen, wird der Ausgangsname jedes Ausganges in max. 4 Blöcke je 8 Zeichen zerlegt (max. Länge des Namens = 32 Zeichen),

Es können somit gezielt jeder der 4 Blöcke von jedem der maximal 64 Ausgänge übertragen werden.

Dabei gelten folgende Zahlenbereiche:
BlockNr 0..3
Ausgangs-Nr 0..63 (entspricht Ausgang 1-64)

4.2.2.6 SET_GEAR_PARA

Parameter zum NS:
1. Fiktive Geberauflösung (High)
2. Fiktive Geberauflösung (Low)
3. Dezimalpunkt-Position
4. Bezeichnung für Maßeinheit 1.Char
5. Bezeichnung für Maßeinheit 2.Char
6. Bezeichnung für Maßeinheit 3.Char

Parameter vom NS:
1. Quittung (s. Bemerkung)

4.2.2.7 SET_OUTPUT_MATTE (nur Mattenschaltwerk)

Parameter zum NS:
1. Programm-Nummer (0..Max)
2. Fikt. Geberposition (High)
3. Fikt. Geberposition (Low)
4. Ausgangsbelegung 1..8
5. Ausgangsbelegung 9..16
6. Ausgangsbelegung 17..23
7. Ausgangsbelegung 24..32
8. Kommando:
   0 = Ausgänge ändern
   1 = Position einfügen
   2 = Position löschen
   3 = Nockenfeld neu aufbauen + Ausg. enable
   (Prog+Pos -> Akt.Prog + Akt.Pos)
   4 = Ausgänge disablen

Parameter vom NS:
1. Quittung (s. Bemerkung)

Bemerkung: Programmtechnisch bedingt ist die Quittung der NS immer 0. Tritt beim Programmieren ein Fehler in der NS auf, kann dieser über GET_STATUS erfragt werden.