

Motion Control Schunk V 1.59

Software Handbuch



Impressum

Urheberrecht:

Diese Anleitung bleibt urheberrechtlich Eigentum der SCHUNK GmbH & Co. KG. Sie wird nur unseren Kunden und den Betreibern unserer Produkte mitgeliefert und ist Bestandteil des Produktes. Ohne unsere ausdrückliche Genehmigung dürfen diese Unterlagen weder vervielfältigt noch dritten Personen, insbesondere Wettbewerbsfirmen, zugänglich gemacht werden.

Technische Änderungen:

Änderungen im Sinne technischer Verbesserungen sind uns vorbehalten.

Dokumentennummer: 0389505

Auflage: V 1.00 | 16.01.2015 | de

© SCHUNK GmbH & Co. KG

Alle Rechte vorbehalten

Sehr geehrter Kunde,

wir gratulieren zu Ihrer Entscheidung für SCHUNK. Damit haben Sie sich für höchste Präzision, hervorragende Qualität und besten Service entschieden.

Sie erhöhen die Prozesssicherheit in Ihrer Fertigung und erzielen beste Bearbeitungsergebnisse – für die Zufriedenheit Ihrer Kunden.

SCHUNK-Produkte werden Sie begeistern.

Unsere ausführlichen Montage- und Betriebshinweise unterstützen Sie dabei.

Sie haben Fragen? Wir sind auch nach Ihrem Kauf jederzeit für Sie da.

Mit freundlichen Grüßen

Ihre SCHUNK GmbH & Co. KG

Spann- und Greiftechnik

Bahnhofstr. 106 – 134

D-74348 Lauffen/Neckar

Tel. +49-7133-103-0

Fax +49-7133-103-2399

info@de.schunk.com

www.schunk.com



Inhaltsverzeichnis

1 Allgemein	8
1.1 Warnhinweise.....	8
1.2 Elektrischer Anschluss	8
1.3 Anzeigenelemente.....	10
1.3.1 Werkseinstellungen	11
1.3.2 Booten	11
1.4 Protokoll	12
1.4.1 Daten Format	12
1.4.2 Datenrahmen	15
1.4.3 Besonderheiten bei Serieller Kommunikation.....	17
1.4.4 Besonderheiten bei CAN	18
1.4.5 Besonderheiten bei Profibus	19
1.4.6 Fragmentierung.....	21
1.5 Einheitensystem	23
1.5.1 Float.....	23
1.5.2 Integer	24
1.6 Benutzerverwaltung	25
1.6.1 Nutzer.....	25
1.6.2 Diagnose.....	25
1.6.3 Profi	26
1.6.4 Advanced.....	26
1.6.5 Root	26
1.7 Pseudoabsolutwertgeber	27
1.7.1 Voraussetzungen.....	27
1.7.2 Funktion	28
1.8 Stillstandskommutierung	29
1.8.1 Voraussetzungen.....	29
1.8.2 Funktion	29

2 Kommandos	30
2.1 Achsindex.....	30
2.2 Bewegung	32
2.2.1 CMD REFERENCE (0x92)	32
2.2.2 CMD REFERENCE HAND (0x97)	33
2.2.3 MOVE POS (0xB0).....	35
2.2.4 MOVE POS REL (0xB8).....	37
2.2.5 MOVE POS TIME (0xB1)	39
2.2.6 MOVE POS TIME REL (0xB9).....	41
2.2.7 MOVE POS LOOP (0xBA)	43
2.2.8 MOVE CUR (0xB3)	43
2.2.9 MOVE VEL (0xB5)	45
2.2.10 MOVE GRIP (0xB7)	46
2.2.11 SET TARGET VEL (0xA0)	47
2.2.12 SET TARGET ACC (0xA1)	48
2.2.13 SET TARGET JERK (0xA2)	48
2.2.14 SET TARGET CUR (0xA3)	49
2.2.15 SET TARGET TIME (0xA4)	50
2.2.16 SET TARGET POS (0xA6)	50
2.2.17 SET TARGET POS REL (0xA7).....	51
2.2.18 CMD STOP (0x91)	52
2.2.19 CMD FAST STOP (0x90)	53
2.3 Spontanmeldungen	54
2.3.1 CMD INFO (0x8A)	54
2.3.2 CMD MOVE BLOCKED (0x93)	55
2.3.3 CMD POS REACHED (0x94).....	56
2.3.4 CMD ERROR (0x88)	56
2.3.5 GET STATE (0x95)	57
2.3.6 GET STATE AXIS (0x98)	57
2.3.7 CMD TOGGLE IMPULSE MESSAGE (0xE7)	58
2.4 Einstellungen	59
2.4.1 SET CONFIG (0x81)	59
2.4.2 GET CONFIG (0x80)	62
2.4.3 SET CONFIG EXT (0x83)	65
2.4.4 GET CONFIG EXT (0x82).....	68
2.4.5 CMD_DEFAULT (0x8C).....	69
2.5 Befehle für Interne Programmierung.....	70
2.5.1 SET PHRASE (0xC0)	71
2.5.2 GET PHRASES (0xC2)	72
2.5.3 PRG EXE (0xCF)	72

2.5.4 EXE PHRASE (0xC1).....	73
2.5.5 EXE PHRASE0 (0xD0)	73
2.5.6 EXE PHRASE1 (0xD1)	74
2.5.7 EXE PHRASE2 (0xD2)	75
2.5.8 EXE PHRASE3 (0xD3)	75
2.5.9 EXE PHRASE4 (0xD4)	76
2.5.10 EXE PHRASE5 (0xD5)	77
2.5.11 EXE PHRASE6 (0xD6)	77
2.5.12 EXE PHRASE7 (0xD7)	78
2.5.13 EXE PHRASE8 (0xD8)	79
2.5.14 EXE PHRASE9 (0xD9)	79
2.5.15 EXE PHRASE10 (0xDA)	80
2.5.16 EXE PHRASE11 (0xDB)	80
2.5.17 EXE PHRASE12 (0xDC)	81
2.5.18 EXE PHRASE13 (0xDD)	81
2.5.19 EXE PHRASE14 (0xDE)	82
2.5.20 EXE PHRASE15 (0xDF).....	82
2.5.21 PRG GOTO (0xC3)	83
2.5.22 PRG WAIT (0xC4)	83
2.6 Sonstige	84
2.6.1 GET STATE (0x95)	84
2.6.2 GET STATE AXIS (0x98)	87
2.6.3 CMD REBOOT (0xE0)	90
2.6.4 CMD DIO (0xE1).....	91
2.6.5 CMD DIO AXIS(0xEB)	92
2.6.6 CMD GET DIO (0xE9)	93
2.6.7 CMD SET DIO (0xEA).....	95
2.6.8 FLASH MODE (0xE2)	97
2.6.9 CMD DISCONNECT (0xE6)	98
2.6.10 CHANGE USER (0xE3)	99
2.6.11 CHECK MC PC COMMUNICATION (0xE4)	100
2.6.12 CHECK PC MC COMMUNICATION (0xE5)	102
2.7 Fragmentierung	104
2.7.1 FRAG ACK (0x87)	104
2.7.2 FRAG START (0x84).....	104
2.7.3 FRAG MIDDLE (0x85)	104
2.7.4 FRAG END (0x86).....	105
2.8 Fehlermeldungen	105
2.8.1 Fehlerkommandos	106
2.8.2 Info- und Fehlercodes	109

3 Konfigurationsparameter	118
3.1 Allgemein	118
3.2 EEPROM.....	118
3.2.1 Motor	119
3.2.2 Getriebe	124
3.2.3 Referenzierung.....	125
3.2.4 Regler	133
3.2.5 Gerät.....	136
3.2.6 Positionierung	146
3.2.7 Bremse	150
4 Motion Tool Schunk	152
4.1 Voraussetzungen	152
4.2 Erste Schritte	152
4.3 Hauptfenster.....	155
4.3.1 Symbolleiste	156
4.3.2 Menü	157
4.3.3 Ausgabefenster	159
4.3.4 Grundeinstellungen.....	160
4.3.5 Tools	162
4.4 Modulfenster.....	164
4.4.1 Modulzustände	166
4.4.2 Schalterelemente	167
4.4.3 Modulparameter.....	168
4.4.4 Menü	171
4.4.5 Manuelle Referenzierung.....	177
4.4.6 Inbetriebnahme Assistent.....	177
4.5 Tipps	178
4.5.1 Unterstützte Sprachen	178
4.5.2 Treiber Vector CAN	178
4.5.3 Treiber Peak CAN	178
4.5.4 Treiber Softing CAN.....	178
4.5.5 Schnittstelle ESD CAN.....	179
4.5.6 Schnittstelle Siemens Profibus.....	179
4.5.7 Modulstatus automatisch anzeigen lassen.....	179
4.5.8 Kommunikationsschnittstelle beim Start öffnen	179
4.5.9 Datendurchsatz mit CAN.....	180
4.5.10 Konfigurierte Module unter Profibus	180
4.5.11 Öftere Timeouts bei der Seriellen Kommunikation.....	181
4.5.12 Einzelne EEPROM-Parameter ändern	181
4.5.13 Hauptfenster nicht maximieren.....	182
4.5.14 Kommunikationsart „Auto“ beim Modul.....	182
4.5.15 Keine automatische Initialisierung.....	182

5	Bemerkungen.....	183
5.1	Module	183
5.2	Protokoll	183
5.3	Serielle Kommunikation	183
5.4	CAN	184
5.5	Profibus.....	184
6	Anhang.....	185
6.1	Ausgesuchte Beispiele	185
6.1.1	Serielle Kommunikation	185
6.1.2	CAN.....	188
6.1.3	Profibus	192
6.2	CRC16 Berechnung für Serielle Kommunikation.....	196
6.3	Kommando	197
6.4	Info- und Fehlercodes.....	200
6.5	Unterstützte Hardware	202
7	Versionshinweise	203
8	Max. Distanz Schalter	Fehler! Textmarke nicht definiert.

1 Allgemein

1.1 Warnhinweise

Zur Verdeutlichung von Gefahren werden in den Warnhinweisen folgende Signalworte und Symbole verwendet.

	GEFAHR
	Gefahren für Personen. Nichtbeachtung führt sicher zu irreversiblen Verletzungen bis hin zum Tod.
	WARNUNG
	Gefahren für Personen. Nichtbeachtung kann zu irreversiblen Verletzungen bis hin zum Tod führen.
	VORSICHT
	Gefahren für Personen. Nichtbeachtung kann zu leichten Verletzungen führen.
	ACHTUNG
	Sachschaden Informationen zur Vermeidung von Sachschäden.

1.2 Elektrischer Anschluss

Das Modul verfügt über getrennte Eingangsklemmen für die Motorspannung und die Logikspannung (24V DC). Es wird empfohlen, beide Spannungen getrennt anzuschließen. Damit ist gewährleistet, dass bei einer Überlastung der Motorspannung die Logik weiter arbeitet, und der Zustand des Moduls immer bekannt ist. Bei Modulen mit einer Motorspannung > 24V DC ist eine getrennte Spannungsversorgung ohnehin unumgänglich, da die Logikspannung immer im Bereich zwischen 18V DC und 32V DC liegen muss.

	ACHTUNG
Dauerhafte Beschädigung der Elektronik möglich!	
<ul style="list-style-type: none"> Bei getrennter Versorgung ist darauf zu achten, dass ein Potentialausgleich zwischen den beiden Versorgungsspannungen vorgenommen wird (Massen zusammenführen) Es darf nur der Pluspol abgeschaltet werden, der Motor-GND muss immer verbunden sein. 	

HINWEIS

Ist die Spannungsversorgung von Logik und Motor getrennt, kann von der Steuerung die Leistung mittels Relais abgeschaltet werden, das Modul ist aber weiter am Bussystem aktiv.

Bei der Versorgung des Leistungsteils (Motor) ist ein Netzteil einzusetzen, welches in der Lage ist ausreichend Strom für das jeweilige Modul zu liefern. Bei der Verkabelung ist auf ausreichende Kabelquerschnitte zu achten.

Der Spannungsabfall am Kabel lässt sich durch folgende Formel berechnen:

$$\Delta U = (2 \times I \times l) / (\chi \times A) \text{ mit:}$$

I: Stromaufnahme der Last

l: Länge der Leitung

χ : elektrische Leitfähigkeit

Cu: $\chi = 56 \text{ ((m) / (\Omega \times mm^2))}$

Al: $\chi = 35 \text{ ((m) / (\Omega * mm^2))}$

A: Leiterquerschnitt

1.3 Anzeigenelemente

Alle Module besitzen 3 Anzeigenelemente in Form von LEDs.

HINWEIS

Bei einigen Bauformen sind diese Anzeigen allerdings nicht herausgeführt und somit nicht sichtbar.

Eine grüne LED (POW LED) dient der Zustandsanzeige der Motorspannung. Sollte die LED nicht bzw. ganz schwach leuchten ist die Motorversorgungsspannung zu prüfen (24V - 48V DC).

Die beiden weiteren LEDs (grün bzw. rot) zeigen den Status der Logikschaltkreise an:

Bedeutung der LED

LED1 (grün)	LED2 (rot)	Bedeutung
dauerhaft an	dauerhaft an	Modul befindet sich im <i>Flash Modus</i> (☞ 2.6.8, Seite 97) bzw. kein Bussystem ist aktiv.
an	blinkt	Neue Firmware wird übertragen (ab V1.20)
kurz an	kurz an	Modul bootet.
an	aus	Modul ist betriebsbereit und Bussystem ist aktiv.
flackert	aus	Daten werden ausgetauscht.
aus	aus	Logikversorgungsspannung fehlt. Sind vorher beide LEDs kurzfristig an gewesen (Bootphase), so ist die Initialisierung des angeschlossenen Bussystems bisher nicht erfolgreich gewesen. Bitte Buskabel prüfen. Ist Master aktiv?
aus	an / blinkt	Im Modul ist ein Fehler (☞ 2.8.1, Seite 106) aufgetreten.
aus / an	an / aus	Profibus ist aktiv, aber noch nicht im „Data Exchange“ Modus, bzw. automatische Schnittstellenerkennung arbeitet.
flackert	blinken / flackern / an	Daten auf der „Hauptschnittstelle“ werden ausgetauscht, gleichzeitig ist die <i>Diagnoseschnittstelle</i> (☞ 1.6.2, Seite 25) aktiv und es werden auch hierüber Daten ausgetauscht.
aus / an schnell	an / aus schnell	Firmware ist in einem undefinierten Zustand. (Sollte niemals passieren!)

1.3.1 Werkseinstellungen

Wird das Modul auf Werkseinstellung zurückgesetzt, so werden folgende Werte im EEPROM überschrieben bzw. neu gesetzt:

- Modul ID
- Baudrate für CAN Bus
- Baudrate für Serielle Kommunikation
- Kommunikationssystem Die genauen Werte sind dem Datenblatt des jeweiligen Produktes zu entnehmen.

1.3.2 Booten

Nach dem erfolgreichen Starten des Moduls sind einige Parameter für Bewegungen bereits auf Startwerte gesetzt. Es ist damit möglich das Modul direkt in Betrieb zu nehmen ohne vorher Parameter setzen zu müssen. Folgende Parameter sind gesetzt:

- „Zielgeschwindigkeit“ ([☞ 2.2.11, Seite 47](#))
in [%] vom maximal Wert ([☞ 3.2.1, Seite 119](#)). -> 10%
- „Zielbeschleunigung“ ([☞ 2.2.12, Seite 48](#))
in [%] vom maximal Wert ([☞ 3.2.1, Seite 119](#)). -> 10%
- „Zielruck“ ([☞ 2.2.13, Seite 48](#))
in [%] vom maximal Wert ([☞ 3.2.1, Seite 119](#)). -> 50%
- „Zielstrom“ ([☞ 2.2.14, Seite 49](#))
Nennstrom ([☞ 3.2.1, Seite 119](#)).
- Spontanmeldungen ([☞ 2.3.7, Seite 58](#)) aktiviert.
- Nutzer wird auf „User“ ([☞ 1.6, Seite 25](#)) gestellt

1.4 Protokoll

1.4.1 Daten Format

Daten werden von den Modulen im Intel Format (little Endian) versendet und beim Empfangen in eben diesem Format interpretiert.

HINWEIS

Ist man sich bei der Erstellung eigener Treiber nicht sicher, kann man die „Bytedreherei“ per „CHECK MC PC COMMUNICATION“ ([☞ 2.6.11, Seite 100](#)), bzw. „CHECK PC MC COMMUNICATION“ ([☞ 2.6.12, Seite 102](#)) mit definierten Testdaten prüfen.

Gleitkommazahlen Die Norm IEEE 754 (Gleitkommazahlen) wurde in den frühen 1980er entwickelt, um unter anderem eine konsistente Gleitkommazahlenrepräsentation über verschiedene Rechnerarchitekturen zu erreichen. Dieser Standard wird eingehalten, wenn die Parameter als Gleitkommazahlen zum Modul gesendet bzw. von diesem zur Steuerung übermittelt werden. Eine Gleitkommazahl wird hierbei mit 32-Bit dargestellt.

Vorzeichen Bit	Exponent	Mantisse (normiert)
1 Bit (bit 32)	8 Bit (bit 23.. bit 30)	23 Bit (bit 1.. bit 22)
s	e	f

Da die Mantisse immer auf „1“ normiert ist, werden nur die Nachkommastellen gespeichert. Die führende „1“ wird nicht mit abgespeichert. Ein Float-Wert kann wie folgt berechnet werden.
 $(-1)^s \times 2^{e-127} \times (1.f)_{\text{bin}}$

Einige Beispiele:

	Vorzeichen	Exponent	Mantisse
	1 Bit	8 Bit	23 Bit
7/4	0	01111111	1100000000000000000000000
-34.432175	1	10000100	0001001101101010001100
-959818	1	10010010	11010100101010010100000
+0	0	00000000	0000000000000000000000000
-0	1	00000000	0000000000000000000000000
$2^{-126} = 1.175 \times 10^{-38}$			
kleinste positive Zahl	0	00000001	0000000000000000000000000
$(2 - 2^{-23}) 2^{127} = 3.403 \times 10^{38}$			
größte positive Zahl	0	11111110	1111111111111111111111111
unendlich	0	11111111	1111111111111111111111111
NaN	0	11111111	nicht alle „0“ oder „1“
$2^{-23} = 1.192 \times 10^{-7}$			
kleinste aufzulösende Zahl	0	01101000	0000000000000000000000000
2^{-128}	0	00000000	0100000000000000000000000

Zweierkomplement

Das Zweierkomplement ist eine Möglichkeit um negative Zahlen im Binärsystem darzustellen. Bei dem Modul wird das Zweierkomplement für die Darstellung negativer Ganzzahlen benötigt. (Integer-Einheitensystem ([☞ 1.5.2, Seite 24](#))).

Positive Zahlen werden im Zweierkomplement mit einer führenden 0 (Vorzeichenbit) versehen und ansonsten nicht verändert. Negative Zahlen werden mit einer führenden 1 als Vorzeichenbit versehen und wie folgt kodiert: Sämtliche Ziffern der entsprechenden positiven Zahl werden negiert. Zum Ergebnis wird 1 addiert. Beispielhafte Umwandlung der negativen Dezimalzahl -4(dez) ins Zweierkomplement:

- 1 Vorzeichen ignorieren und ins Binärsystem umrechnen: $4_{\text{dez}} = 00000100_{\text{bin}} = 0x\ 04_{\text{hex}}$
- 2 Invertieren, da negativ: $11111011_{\text{bin}} = 0x\ FB_{\text{hex}}$
- 3 Eins addieren, da negativ: $11111011_{\text{bin}} + 00000001_{\text{bin}} = 11111100_{\text{bin}} = 0x\ FC_{\text{hex}} = -4_{\text{dez}}$

Etwas mathematischer:

Ist x eine negative Zahl, so errechnet sich x im Zweierkomplement

(x_z) mit n Stellen wie folgt:

$$x_z = 2^n - |x|$$

Dementsprechend gilt auch:

$$x_z + |x| = 2^n$$

Da die Module im „*Integer-Einheitensystem*“ ([☞ 1.5, Seite 23](#)) immer mit Int32 (4 Byte) arbeiten, lassen sich die Bytefolgen für negative Zahlen x (z.B. -112) einfach wie folgt berechnen:

$$\begin{aligned} y &= 4294967296_{\text{dez}} - |x| \rightarrow y = 4294967296_{\text{dez}} - 112_{\text{dez}} = \\ &4294967184_{\text{dez}} \\ y &= 0x\ 10000000_{\text{hex}} - |x| \rightarrow y = 0x\ 10000000_{\text{hex}} - 0x\ 70_{\text{hex}} = 0x \\ &\text{FFFFF90}_{\text{hex}} \end{aligned}$$

1.4.2 Datenrahmen

Der Datenrahmen des Motion-Protokolls umfasst immer folgende Elemente:

- D-Len (1 Byte)
- KOMMANDO CODE (1 Byte)



Abb. 1 Datenrahmen

D-Len (Data Length) gibt die Anzahl der nachfolgenden Nutzdaten einschließlich des Kommando Bytes an. Der Datenrahmen besteht aus einem Byte, deshalb können mit einer Motion-Protokoll Nachricht maximal 255 Daten Bytes übertragen werden.

Im Anschluss an das D-Len Byte folgt immer der aus einem Byte bestehende Kommando-Code. Dem Kommando-Code folgen, falls notwendig, die jeweilig benötigten Parameter. Falls erforderlich, kann der Kommando-Code noch durch „Unter Kommando-Codes“ erweitert werden.

Ein Beispiel hierfür ist der *GET CONFIG* ([2.4.2, Seite 62](#)) Befehl.

Optional kann am Ende der „Achs-Index“ ([2.1, Seite 30](#)) eingefügt werden.

HINWEIS

Der Achs-Index wird nur für Systeme benötigt, bei denen über eine Adresse mehrere Motoren angesprochen werden können. Soll bei Systemen mit einem Motor der Achs-Index verwendet werden ist dieser immer „0“

Alle abgesandten Befehle werden sofort vom Modul mit einer Antwort (Acknowledge) bestätigt. Diese Antwort benutzt ebenfalls den oben beschriebenen Datenrahmen. (D-Len, Kommando-Code, evtl. Parameter). Wurde die Anfrage erfolgreich verarbeitet, besitzt D-Len immer einen Wert ungleich „0x02“. Ist die Anfrage fehlerhaft gewesen, besitzt D-Len genau den Wert „0x02“.

In den beiden folgenden Bytes ist die Ursache der *fehlerhaften Anfrage* ([☞ 2.8.2, Seite 109](#)) beschrieben.

Die Module haben darüber hinaus die Eigenschaft, ohne dass eine Anfrage gestellt worden ist, von sich aus Meldungen abzusetzen. Der Datenrahmen wird bei solchen „Spontanmeldungen“ ebenfalls eingehalten. Bei folgenden Ereignissen wird eine „Spontanmeldung“ ausgelöst:

- Es ist ein schwerwiegender Fehler aufgetreten.
- Eine Bewegung wurde korrekt beendet.
- Regelmäßige *Statusmeldung* ([☞ 2.3, Seite 54](#)), falls aktiviert.

1.4.3 Besonderheiten bei Serieller Kommunikation

Da die serielle Schnittstelle bei der Entwicklung nicht als Bussystem gedacht war, müssen zusätzlich zum Datenrahmen einige Elemente eingefügt werden, damit mehrere Module mit einer einzigen seriellen Schnittstelle bedient werden können.

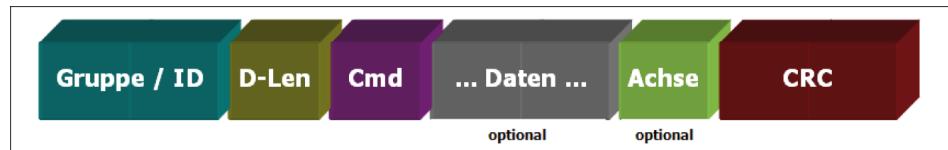


Abb. 2 Datenrahmen serielle Kommunikation

Zunächst werden vor dem Datenrahmen zwei Bytes (Group/ID) angehängt, mit welchen bestimmt wird, welches Modul angesprochen werden soll, bzw. von welchem Modul die Antwort stammt. Hierbei werden vom ersten Byte nur die ersten drei Bits verwendet. Das zweite Byte stellt die eindeutige Modul ID dar. => es können bis zu 255 Module adressiert werden. Die ersten drei Bits des ersten Bytes sind wie folgt kodiert:

- 0x03 Fehlernachricht eines Moduls
- 0x05 Nachricht vom Master zu einem Modul
- 0x07 Antwort vom Modul

Alle anderen Zustände werden nicht benutzt.

HINWEIS

Diese Art ein Modul mit 11Bit eindeutig zu identifizieren wurde aus dem CAN Protokoll übernommen.

Um eine sichere serielle Datenübertragung gewährleisten zu können, werden über alle Daten, einschließlich Group/ID, D-Len und Cmd eine Checksumme (CRC16 => 2 Byte) gebildet, welche am Ende des Datenrahmens angehängt wird. Ein Algorithmus, um eine CRC16 zu berechnen, befindet sich im Anhang ([☞ 6.2, Seite 196](#)).

1.4.4 Besonderheiten bei CAN

CAN als nachrichtenorientiertes Bussystem, benötigt neben dem Datenrahmen einen eindeutigen der Nachricht zugeordneten Identifier. Die Module unterstützen den Standard 11-Bit Identifier. Die unteren 8Bit werden hierbei für die eindeutige Modul ID verwendet. Es können also maximal 255 Module adressiert werden. Die noch freien 3Bit sind wie folgt kodiert:

- 0x03 Fehlernachricht eines Moduls
- 0x05 Nachricht vom Master zu einem Modul
- 0x07 Antwort vom Modul

Alle anderen Zustände werden nicht benutzt.

- Eine Nachricht zum Modul besitzt also folgenden Identifier 0x5XX. (XX = Moduladresse in hex Darstellung)
- Eine Nachricht vom Modul besitzt den Identifier 0x7XX. (XX = Moduladresse in hex Darstellung)
- Im Fehlerfall werden die Nachrichten vom Modul an den Master mit dem Identifier 0x3XX. (XX = Moduladresse in hex Darstellung) versendet.

In einer CAN Nachricht können maximal 8 Byte verschickt werden. Es ist unter Umständen notwendig einen längeren Datenrahmen (D-Len > 7) in mehrere CAN Nachrichten zu verpacken. Dies geschieht mittels des Fragmentierungsprotokolls ([☞ 1.4.6, Seite 21](#)).

HINWEIS

Eine Fragmentierung ist normalerweise nicht notwendig, da alle für den Betrieb der Module notwendigen Befehle in eine CAN Nachricht verpakt werden können.

1.4.5 Besonderheiten bei Profibus

Beim Profibus DPV0 gibt es folgende Besonderheiten: Die maximale Länge der auf einmal vom Master zum Modul zu übertragenden Daten ist auf 8 Byte begrenzt. Hiermit kann ein Modul komplett bedient werden. (Es werden maximal 7 Byte für eine Nachricht vom Master zum Modul benötigt)

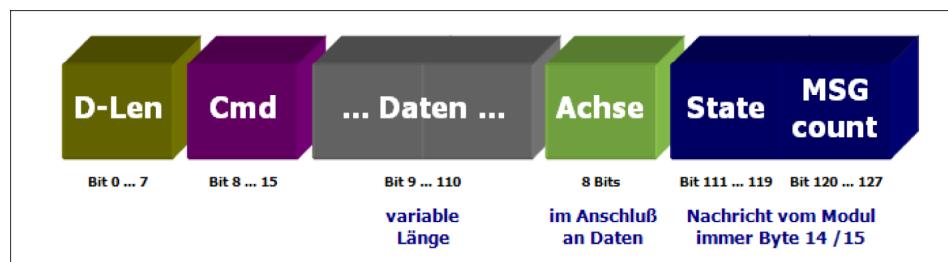


Abb. 3 Datenrahmen Profibus

Die maximale Länge der vom Modul zum Master gesendeten Daten (Antwort) ist auf 16 Byte begrenzt (GSD-Datei). Will man größere Datenmengen versenden / empfangen kann eine *Fragmentierung* ([☞ 1.4.6, Seite 21](#)) notwendig werden. In 16 Byte findet die längste im Normalbetrieb vorkommende Nachricht vom Modul zum Master Platz, es bleiben sogar noch 2 Byte übrig. In diesen 2 Byte, welche immer am Ende der Profibusnachricht stehen (Byte 14, Byte 15) ist,

- 1 der aktuelle *Status* ([☞ 2.6.1, Seite 84](#)) des Moduls (Byte 14) und
- 2 ein so genannter Kommando Zähler (MsgCount) (Byte 15) enthalten.

Werden die Bytes 10-13 nicht benötigt, kann hier die aktuelle Position im jeweiligen *Einheitensystem* ([☞ 1.5, Seite 23](#)) ausgelesen werden.

HINWEIS

Bytes 10-13 werden nur in der Antwort auf ein GET STATE ([☞ 2.6.1, Seite 84](#)) verwendet, wenn alle Daten (Position, Geschwindigkeit und Strom) angefordert werden. Bei fragmentierten Nachrichten werden die Bytes 10-15 für Daten verwendet.

HINWEIS

Nur die oberen 8 Bit des Status-Wortes werden geschrieben. Der Fehlercode entfällt. Hierzu gibt es zum einen die erweiterte Diagnose unter Profibus, zum anderen wird im Fehlerfall der Fehlercode ([☞ 2.8.1, Seite 106](#)) in den Ausgangsdaten dargestellt. Wird eine Nachricht vom Master zum Modul versandt, so wird bei Profibus zusätzlich zur Antwort der MsgCount um 1 erhöht. Hiermit wird sichergestellt, dass trotz möglicher Spontanmeldungen jede Anfrage bestätigt wird.

HINWEIS

Eine Spontanmeldung ([☞ 2.3, Seite 54](#)) erhöht den MsgCount nicht!

Will man zum Beispiel auf eine Position fahren, an der sich das Modul aktuell befindet, meldet das Modul „Befehl verstanden“ und sofort im nächsten Profibus-Zyklus „Position erreicht“. Da eine mit dem Profibus verbundene Steuerung unter Umständen nicht in jedem Profibus-Zyklus die Daten abfragt, könnte die Acknowledge (Antwort) auf den Verfahrbefehl verloren gehen. Durch den MsgCount ist gewährleistet, dass eine Bestätigung auf die Anfrage eingegangen ist. Über das *Status-Byte* ([☞ 2.6.1, Seite 84](#)) (Byte 14) erhält man immer die aktuelle Information über den Zustand des vorhandenen Moduls.

HINWEIS

Das letzte Bit des MsgCount kann als Toggle-Bit ausgewertet werden. (Modul zum Master) Bei der Übertragung von Daten vom Master zum Modul kann das nicht benutzte Byte 8 als Toggle-Byte verwendet werden, bzw. Bit 63 als Toggle-Bit.

Gruppen werden vollständig durch den in Profibus implementierten SYNC, FREEZE Mechanismus unterstützt.

Eine Änderung der Adresse ist jederzeit über den Service „Set Slave Address“ (SAP 55) möglich. „Real No Add Change“ wird im *Gruppen Byte* ([☞ 3.2.5, Seite 136](#)) gespeichert. Ein gesetztes „Real No Add Change“ (0xFF) kann somit wieder über eine *Neukonfiguration des Gruppenbytes* ([☞ 2.4.1, Seite 59](#)) gelöscht werden.

Sollte eine konsistente Datenübertragung nicht möglich sein, so gibt es folgende Möglichkeiten das Modul zu betreiben:

- SYNC, UNSYNC Mechanismus einsetzen.
- D-Len auf „0“ setzen. Alle Daten auffüllen und sobald alle Daten vorhanden sind D-Len setzen.

1.4.6 Fragmentierung

HINWEIS

Eine Fragmentierung von Nachrichten ist für den Normalbetrieb nicht erforderlich!

Sollte eine Fragmentierung von Meldungen erforderlich sein, geschieht dies wie folgt:

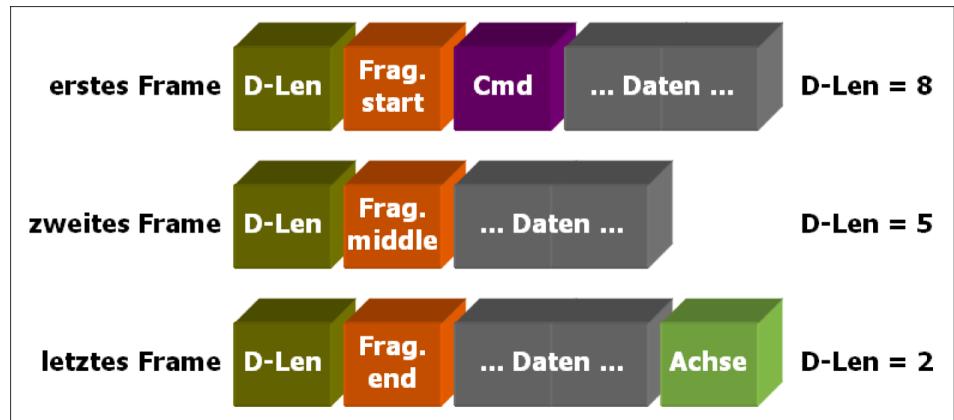


Abb. 4 Fragmentierung

Zu Beginn jeder Nachricht wird die Länge der noch folgenden Nutzdaten gesendet. Anschließend wird eine Fragmentkennung gesendet. Diese Fragmentkennung wird **nicht** im Längenbyte (D-Len) erfasst.

- FragStart -> *erstes Fragment* ([☞ 2.7.2, Seite 104](#)).
- FragMiddle -> *ein mittleres Fragment* ([☞ 2.7.3, Seite 104](#)).
- FragEnd -> *letztes Fragment* ([☞ 2.7.4, Seite 105](#)).

Diese einzelnen Fragmente können somit wieder zu einem kompletten *Datenrahmen* ([☞ 1.4.2, Seite 15](#)) zusammengebaut werden, welcher anschließend interpretiert werden kann.

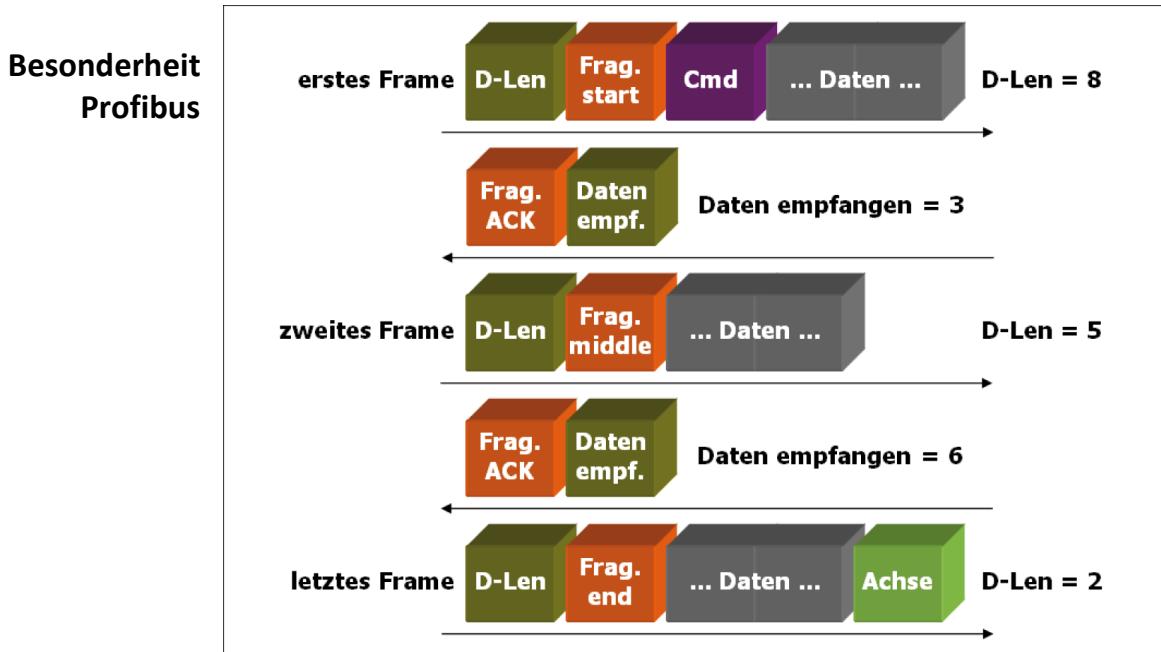


Abb. 5 Fragmentierung Profibus

Da beim Profibus ständig ein „Token“ unterwegs ist, aus dem sich die jeweiligen Teilnehmer die für sie gültigen Daten herausfischen bzw. die für den Master notwendigen einschreiben, muss jedes erhaltene Fragment mit „FRAG ACK“ ([☞ 2.7.1, Seite 104](#)) und dem D-Len Byte des erhaltenen Fragments bestätigt werden. Wird eine fragmentierte Nachricht an den Master gesendet, muss dieser jedes einzelne Fragment mit „FRAG ACK“ und dem D-Len Byte des erhaltenen Fragments bestätigen, damit das Modul das nächste Fragment verschicken kann. Sendet der Master eine fragmentierte Nachricht an das Modul, so muss er mit dem Senden des folgenden Fragmentes solange warten bis das Modul den Erhalt des Fragmentes bestätigt hat („FRAG ACK“ und dem D-Len Byte des erhaltenen Fragments). Das letzte Fragment muss nicht bestätigt werden.

1.5 Einheitensystem

Alle Parameterdaten, welche mit Einheiten behaftet sind, werden im *voreingestellten Einheitensystem* ([☞ 3.2.5, Seite 136](#)) übermittelt.

1.5.1 Float

- [mm] alle Parameter werden als Float Werte übermittelt => Position [mm], Geschwindigkeit [(mm)/(s)], Beschleunigung [(mm)/(s²)], Ruck [(mm)/(s³)], Stromwerte [A], Zeiten [s]
- [m] alle Parameter werden als Float Werte übermittelt => Position [m], Geschwindigkeit [(m)/(s)], Beschleunigung [(m)/(s²)], Ruck [(m)/(s³)], Stromwerte [A], Zeiten [s]
- [Inch] alle Parameter werden als Float Werte übermittelt => Position [Inch], Geschwindigkeit [(Inch)/(s)], Beschleunigung[(Inch)/(s²)], Ruck [(Inch)/(s³)], Stromwerte [A], Zeiten [s]
- [rad] alle Parameter werden als Float Werte übermittelt => Position [rad], Geschwindigkeit [(rad)/(s)], Beschleunigung[(rad)/(s²)], Ruck [(rad)/(s³)], Stromwerte [A], Zeiten [s]
- [Grad] alle Parameter werden als Float Werte übermittelt => Position [Grad], Geschwindigkeit [(Grad)/(s)], Beschleunigung[(Grad)/(s²)], Ruck [(Grad)/(s³)], Stromwerte [A], Zeiten [s]
- [intern] alle Parameter werden als Float Werte übermittelt => Position [intern], Geschwindigkeit [(intern)/(s)], Beschleunigung [(intern)/(s²)], Ruck [(intern)/(s³)], Stromwerte [A], Zeiten [s]
Dieses Einheitensystem sollte nur für Testzwecke eingestellt werden! Intern werden alle Daten in diesem Einheitensystem gerechnet, welches auf Motorumdrehungen basiert. Getriebe-über- bzw. unterstellungen werden nicht berücksichtigt.

1.5.2 Integer

- [μm] alle Parameter werden als Integer Werte übermittelt =>
Position [μm], Geschwindigkeit [$(\mu\text{m})/(s)$], Beschleunigung [$(\mu\text{m})/(s^2)$], Ruck [$(\mu\text{m})/(s^3)$], Stromwerte [mA], Zeiten [ms]
- [μGrad] alle Parameter werden als Integer Werte übermittelt
=>
Position [μGrad], Geschwindigkeit [$(\mu\text{Grad})/(s)$], Beschleunigung [$(\mu\text{Grad})/(s^2)$], Ruck [$(\mu\text{Grad})/(s^3)$], Stromwerte [mA], Zeiten [ms]
- [μInch] alle Parameter werden als Integer Werte übermittelt =>
Position [μInch], Geschwindigkeit [$(\mu\text{Inch})/(s)$], Beschleunigung [$(\mu\text{Inch})/(s^2)$], Ruck [$(\mu\text{Inch})/(s^3)$], Stromwerte [mA], Zeiten [ms]
- [Milligrad] alle Parameter werden als Integer Werte übermittelt
=>
Position [Milligrad], Geschwindigkeit [$(\text{Milligrad})/(s)$],
Beschleunigung [$(\text{Milligrad})/(s^2)$], Ruck [$(\text{Milligrad})/(s^3)$],
Stromwerte [mA], Zeiten [ms]

HINWEIS

Die Konfigurationsparameter ([☞ 3.2, Seite 118](#)) sind im jeweiligen Einheitensystem zu übermitteln!

1.6 Benutzerverwaltung

Das Modul ist mit einer Benutzerverwaltung ausgerüstet, um bestimmte Aktionen besonders zu schützen. Die Benutzerrechte können über „*CHANGE USER*“ ([☞ 2.6.10, Seite 99](#)) umgeschaltet werden.

1.6.1 Nutzer

Ist der Standard-Nutzer, welcher nach dem Einschalten des Moduls immer aktiviert wird. Dieser kann das Modul vollständig bedienen. Parametrieren ist nur für die wichtigsten *Parameter* ([☞ 3.2, Seite 118](#)) gestattet.

1.6.2 Diagnose

Ist der Diagnose-Nutzer. Meldet man diesen Benutzer an, so wird eine weitere Schnittstelle geöffnet*. Über diese kann der Busverkehr auf der Primären Schnittstelle mitgeschnitten werden, oder auch sehr eingeschränkt gezielte Informationen können abgerufen werden. Parametrierungen der Module sind möglich.

* CAN oder PROFIBUS aktiv => Serielle Kommunikation wird zusätzlich geöffnet; Serielle Kommunikation aktiv => CAN wird zusätzlich geöffnet.

Wird über die primäre Schnittstelle der *Diagnose-Nutzer* ([☞ 2.6.10, Seite 99](#)) aktiviert, ist ein „Mitschnitt“ aktiv.

Mit dem Kommando „*CMD TOGGLE IMPULSE MESSAGE*“ ([☞ 2.3.7, Seite 58](#)) kann der Zustand der sekundären Schnittstelle von „Lauschen“ auf „Parametrieren“ umgestellt werden. Das Kommando muss über die sekundäre Schnittstelle gesendet werden.

Im Zustand „Parametrieren“ können Kommandos über die sekundäre Schnittstelle gesendet werden.

Ist das Modul im Fehlerfall, besteht die Möglichkeit sich direkt über die sekundäre Schnittstelle als *Diagnose-Nutzer* ([☞ 2.6.10, Seite 99](#)) an dem Modul anzumelden.

Über Parametriersoftware "Diagnose finden" unter "Module" auswählen (nur bei RS232).

HINWEIS

Bitte beachten! Eine aktive Steuerung der Module ist nicht möglich!

HINWEIS

Nach Abschluss der „Diagnosetätigkeit“ ist das Modul neu zu starten (Versorgungsspannung Logik trennen und erneut anschließen)!

HINWEIS

Das Verhalten des Moduls kann bei aktiver Diagnoseschnittstelle von dem „normalen“ Verhalten abweichen!

1.6.3 Profi

Ist der Profi-Nutzer, welcher den vollen Funktionsumfang des „Nutzer“ hat und zusätzlich weitere Parameter verstehen kann. Bei falscher Parametrierung kann es zu unvorhergesehenem Verhalten des Moduls kommen. Das Modul kann jedoch nicht zerstört werden.

Das Standard-Kennwort für die Profi-Rechte lautet „Schunk“.

Das Standard-Kennwort für die Profi-Rechte lautet „HDAG“.

1.6.4 Advanced

Ist der Advanced-Nutzer, welcher den vollen Funktionsumfang des „Profi“ hat und zusätzlich weitere Parameter verstehen kann.

	ACHTUNG Eine Fehlbedienung oder eine falsche Parametrierung kann zur Zerstörung der Elektronik oder des Motors führen.
--	---

1.6.5 Root

Ist der Root-Nutzer, welcher vollen Zugriff auf das Modul hat. Es sind alle Parameter einstellbar und weitere Funktionen für Testzwecke zugänglich.

	ACHTUNG Eine Fehlbedienung oder eine falsche Parametrierung kann zur Zerstörung der Elektronik oder des Motors führen.
---	---

1.7 Pseudoabsolutwertgeber

1.7.1 Voraussetzungen

Die Module unterstützen die Pseudoabsolutwertgeberfunktion, wenn folgende Voraussetzungen erfüllt sind:

- Bremse
- FRAM (*Hardware Version* ([☞ 2.4.2, Seite 62](#)) ungerade)
- *Positionsmesssystem* ([☞ 3.2.6, Seite 146](#)) Resolver
- oder Positionsmesssystem Encoder mit Indexspur und
 - Motortyp DC ([☞ 3.2.1, Seite 119](#))
 - oder Motortyp BLDC
 - oder Motortyp PMSM mit *Stillstandskommutierung* ([☞ 1.8, Seite 29](#))

1.7.2 Funktion

Beim Einfall der Bremse wird die aktuelle Position in einen nicht-flüchtigen Speicher abgespeichert. Wird die Logikspannung abgeschaltet, so wird versucht mit der verbleibenden Restenergie die aktuelle Position abzuspeichern.

- Resolver** Beim erneuten Einschalten des Moduls wird nun die zuvor gespeicherte Position mit einem Kontrollwert verglichen. Ist diese Kontrolle erfolgreich, wird die abgespeicherte Position mit der aktuellen Position des Resolvers verglichen. Sind auch diese Positionen gleich, so ist das Modul nicht neu zu referenzieren.

HINWEIS

Wird der Resolver im unbestromten Zustand um genau eine Umdrehung gedreht, so wird beim Wiedereinschalten die angezeigte Position fehlerhaft sein.

- Encoder mit Indexspur** Beim erneuten Einschalten des Moduls wird die zuvor gespeicherte Position mit einem Kontrollwert verglichen. Ist diese Kontrolle erfolgreich, wird die abgespeicherte Position zur aktuellen Position des Moduls. Gleichzeitig wird der Abstand zum nächsten Indexpuls berechnet. Beim ersten folgenden Verfahrbefehl wird beim Erreichen des Indexpuls der errechnete Abstand mit dem gemessenen Abstand verglichen. Stimmen beide Werte überein, so gilt das Modul als referenziert. Der Indexpuls muss nach dem Senden des ersten Verfahrbefehls darüber hinaus innerhalb einer gewissen Zeit erreicht werden.
Tritt während der Bewegung zum Indexpuls ein *Fehler* Fehler Behandlung auf, so wird die Referenzierung gelöscht.

HINWEIS

Nach einer erfolgreichen Referenzfahrt muss der Indexpuls mindestens einmal überfahren werden, um die Funktion zu aktivieren.

HINWEIS

Wird der Encoder im unbestromten Zustand bewegt, so ist es möglich, dass das Modul nach dem Wiedereinschalten mit der falschen Position eine Bewegung zum nächsten Indexpuls ausführt (max. eine Motorumdrehung).

HINWEIS

Wird der Encoder im unbestromten Zustand um genau eine Umdrehung gedreht, so wird beim Wiedereinschalten die angezeigte Position fehlerhaft sein.

1.8 Stillstandskommutierung

1.8.1 Voraussetzungen

- Motortyp DC ([☞ 3.2.1, Seite 119](#))
- oder Motortyp BLDC
- oder Motortyp PMSM und
 - *Positionsmesssystem* ([☞ 3.2.6, Seite 146](#)) „Encoder mit Indexspur“ und vorhandenen Hallgebern
 - oder Positionsmesssystem Resolver

	<p>ACHTUNG</p> <p>Bewegungsrichtung bei Blockkommutierung und Sinuskommutierung müssen übereinstimmen. Bei unterschiedlichen Drehrichtungen sind Phasen zu tauschen und die Kommutertabelle (☞ 3.2.1, Seite 119) anzupassen.</p>
---	--

1.8.2 Funktion

Sind alle Voraussetzungen gegeben, wird das Modul versuchen die Stillstandkommutierung durchzuführen. Bei Modulen mit Absolutwertmesssystemen kann direkt nach dem Einschalten die Sinus-Kommutierung aktiviert werden, da die Lage des „Sinus-Zeigers“ bekannt ist.

Bei Modulen mit Encodermesssystem ist die Lage des „Sinus-Zeigers“ erst nach Erreichen des Indexpulses bekannt. Daher wird das Modul mit Blockkommutierung zum ersten Indexpuls gefahren (bei der Fahrt mit Blockkommutierung kann die Kraft etwas geringer sein) und anschließend auf Sinus-Kommutierung umgestellt.

Die Lage des „Sinus-Zeigers“ zum Indexpuls wird über den Parameter *Positionierung Offset* ([☞ 3.2.6, Seite 146](#)) eingestellt bzw. „nachjustiert“. Wird dieser Wert zu „0“ gesetzt wird beim nächsten Bewegungsbefehl durch Bestromen der Motorphasen ein „Sinus-Zeigers“ gesucht, welcher *abgespeichert* ([☞ 3.2.6, Seite 146](#)) wird. Die *Referenzierung* ([☞ 1.7, Seite 27](#)) wird hierbei gelöscht.

HINWEIS

Zur Zeigersuche sollte das Modul in alle Richtungen frei beweglich sein. Modul wird ruckartig bis zu zwei Motorumdrehungen bewegt. Eine Kommunikation mit dem Modul ist in dieser Zeit nicht möglich.

2 Kommandos

Alle Beispiele führen nur den Datenrahmen auf. Die Besonderheiten der verschiedenen Bussysteme sind ab Kapitel „*Besonderheiten bei Serieller Kommunikation*“ ([☞ 1.4.3, Seite 17](#)) beschrieben. Einige ausgesuchte Beispiele für die verschiedenen Bussysteme sind im *Anhang* ([☞ 6.1, Seite 185](#)) aufgeführt.

HINWEIS

Bei allen aufgeführten Beispielen wird davon ausgegangen, dass als Einheitensystem [mm] eingestellt ist

Bei allen Beispielen sind nur die *notwendigen Parameter* aufgeführt, die *optionalen Parameter* werden nicht aufgeführt. In den Beispielen steht „M“ für Master und „S“ für Slave (Modul).

2.1 Achsindex

Bis auf wenige Ausnahmen kann bei den in den folgenden Abschnitten beschriebenen Kommandos optional ein Achsindex als zusätzlicher Parameter hinzugefügt werden. Dies ist notwendig bei Modulen, die mehrere Achsen oder sogar mehrere Roboter ansteuern. Wenn vorhanden, beschränkt der Zusatz „Achsindex“ die Wirkung des jeweiligen Kommandos entweder auf eine bestimmte Achse oder auf einen bestimmten Roboter.

Ist das höchstwertige Bit (MSB, Bit Nr. 7) gelöscht, so bezeichnet der „Achsindex“ Parameter die Nummer einer Achse. Ist das höchstwertige Bit (MSB, Bit Nr. 7) gesetzt, so bezeichnen die niederwertigeren Bits (Bits Nr. 6..0) des „Achsindex“ Parameter die Nummer eines Roboters.

- Der Parameter „Achsindex“ wird mit einem Byte übertragen.
- Der Parameter „Achsindex“ wird am Ende der normalen Parameter eines Kommandos hinzugefügt.
- Der Parameter erhöht die D-Len des Kommandos um eins.
- Ein Wert von 0 für den „Achsindex“ kann bei Modulen mit nur einer Achse verwendet werden.
- Ein Wert von [1..128] für den „Achsindex“ bezeichnet die Nummer einer Achse [1..128].
- Ein Wert von [129..255] für den „Achsindex“ bezeichnet die Nummer eines Roboters [1..127] („Achsindex“ - 128).

Ausnahmen: Bei folgenden Kommandos kann *kein* „Achsindex“ hinzugefügt werden:

- *CMD DIO* ([☞ 2.6.4, Seite 91](#))

Wenn hier dennoch ein „Achsindex“ benötigt wird, dann kann alternativ das ([☞ 2.6.5, Seite 92](#)) Kommando verwendet werden.

- *GET STATE* ([☞ 2.6.1, Seite 84](#))

Wenn hier dennoch ein „Achsindex“ benötigt wird, dann kann alternativ das *GET STATE AXIS* ([☞ 2.6.2, Seite 87](#)) Kommando verwendet werden.

2.2 Bewegung

2.2.1 CMD REFERENCE (0x92)

KOMMANDO CODE: 0x92

BESCHREIBUNG: Es wird eine Referenzfahrt ausgeführt. Die Art der Referenzierung wird einmalig in den *Konfigurationsparameter* ([☞ 3.2.3, Seite 125](#)) festgelegt.

PARAMETER (Master -> Slave): Keine

BEISPIEL:

	D-Len	Cmd	Param
M->S	0x01	0x92	
S->M	0x03	0x92	0x4F 0x4B

SONSTIGES: Spontanantwort möglich. Je nach *Referenzierart* ([☞ 3.2.3, Seite 125](#)) kann es zu „CMD MOVE BLOCKED“ ([☞ 2.3.2, Seite 55](#)) oder „CMD POS REACHED“ ([☞ 2.3.3, Seite 56](#)) kommen. Dies hängt vom Flag „MOVE ZERO AFTER REFERENCING“ ([☞ 3.2.3, Seite 125](#)) ab. Ein gesetztes Flag löst nach der Referenzierung eine Positions fahrt aus => „CMD POS REACHED“ ([☞ 2.3.3, Seite 56](#)). Während der Referenzierung werden für die Bewegungen die *eingestellten Parameter* ([☞ 3.2.3, Seite 125](#)) übernommen. Nach der Referenzierung werden die zuletzt eingestellten Werte (vor Referenzfahrt) wieder hergestellt. Bitte auch Besonderheiten des jeweiligen *Positionsmesssystems* ([☞ 3.2.6, Seite 146](#)) beachten.

HINWEIS

Eine einmal erfolgreich durch geführte Referenzierung bleibt nach Abschalten der Betriebsspannungen unter bestimmten Voraussetzungen erhalten. Siehe hierzu „Pseudoabsolutwertgeber“ ([☞ 1.7, Seite 27](#))

HINWEIS

Vor einer Referenzfahrt sind bei einem Greifer alle Werkstücke zu entnehmen.

2.2.2 CMD REFERENCE HAND (0x97)

KOMMANDO CODE: 0x97

BESCHREIBUNG: Es wird eine manuelle Referenzfahrt ausgeführt. Falls notwendig, werden zunächst evtl. notwendige Initialisierungsbewegungen durchgeführt (Sinuszeigersuche, Indexpulssuche). Dieser Vorgang kann bis zu 30sek. benötigen. Hat das Modul die Initialisierungsfahrt erfolgreich durchgeführt wird *Warnung* ([☞ 2.8.1, Seite 106](#)), „Subcode“ „NOT REFERENCED“ ([☞ 2.8.2, Seite 109](#)) gemeldet. Nach erneutem Senden von „CMD REFERENCE HAND“ kann nun mit der Hilfe von „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)) zu einer definierten Referenzmarke gefahren werden. Dieser nächste Schritt wird mit „NOT REFERENCED“ bestätigt. (Beschleunigungen und Geschwindigkeiten dürfen dabei die im Abschnitt *Referenzieren* ([☞ 3.2.3, Seite 125](#))) Werte nicht überschreiten. Ein erneutes Senden von „CMD REFERENCE HAND“ setzt die aktuelle Position auf den zuvor parametrierten *Referenzoffset* ([☞ 3.2.3, Seite 125](#)). Die Referenzierung ist abgeschlossen.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich. Modul führt nächsten Schritt aus.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0x97		
S->M	0x03	0x97	0x4F 0x4B	
S->M	0x02	0x89	0x06	Initialisierung beendet
M->S	0x01	0x97		
S->M	0x02	0x97	0x06	„Jog-Modus“ aktiviert
M->S	0x05	0xB9	0xCD 0xCC 0x4C 0x3E	Modul wird verfahren
.				
.				
.				
M->S	0x01	0x97		
S->M	0x03	0x97	0x4F 0x4B	Referenzposition gesetzt

SONSTIGES: Die Referenzierung kann jederzeit mit *Stop* ([☞ 2.2.18, Seite 52](#)) abgebrochen werden. Ein gesetztes "MOVE ZERO AFTER REFERENCING" ([☞ 3.2.3, Seite 125](#)) wird ignoriert. Nach der Referenzierung werden die zuletzt eingestellten Verfahrwerte (vor Referenzfahrt) wieder hergestellt.

HINWEIS

Ist einmal eine manuelle Referenzierung erfolgreich durchgeführt worden, wird die zuvor eingestellte Referenzierart automatisch zu „Manuell“ umgestellt.

HINWEIS

Diese Prozedur lässt sich einfach mit Hilfe der mitgelieferten Software "Motion Tool Schunk" ([☞ 4.4.5, Seite 177](#)) durchführen.

HINWEIS

Eine einmal erfolgreich durchgeführte Referenzierung bleibt nach Abschalten der Betriebsspannungen unter bestimmten Voraussetzungen erhalten. Siehe hierzu „Pseudoabsolutwertgeber“ ([☞ 1.7, Seite 27](#))

2.2.3 MOVE POS (0xB0)

KOMMANDO CODE: 0xB0

BESCHREIBUNG: Bewegt das Modul an eine festgelegte Position. Die Position wird im konfigurierten *Einheitensystem* ([☞ 1.5, Seite 23](#)) vorgegeben. Für die Positionsfahrt wird das *konfigurierte Verfahrprofil* ([☞ 3.2.5, Seite 136](#)) zugrundegelegt.

PARAMETER (Master -> Slave)*:

- *Position* ([☞ 2.2.16, Seite 50](#)) (optional), im konfigurierten Einheitensystem.
- *Geschwindigkeit* ([☞ 2.2.11, Seite 47](#)) (optional), welche für die Positionsfahrt verwendet wird. Bei Verfahrprofil „No Ramp“ ([☞ 3.2.5, Seite 136](#)) nicht relevant.
- *Beschleunigung* ([☞ 2.2.12, Seite 48](#)) (optional), welche für die Positionsfahrt verwendet wird. Bei Verfahrprofil „No Ramp“ ([☞ 3.2.5, Seite 136](#)) nicht relevant.
- *Strom* ([☞ 2.2.14, Seite 49](#)) (optional), welcher bei der Positionsfahrt nichtüberschritten werden darf. Sollte die Reglerstruktur „CURRENT SPEED“ ([☞ 3.2.4, Seite 133](#)) aktiv sein, muss dieser Wert übermittelt werden (Ruck wird benötigt). Der Wert muss mindestens „0“ sein, ansonsten wird „INFO WRONG PARAMETER“ ([☞ 2.8.2, Seite 109](#)) erzeugt.
- *Ruck* (optional), welcher für die Positionsfahrt verwendet wird. Sollte das Verfahrprofil ungleich „Ruckbegrenzt“ ([☞ 3.2.5, Seite 136](#)) sein, kann dieser Wert nicht mit übermittelt werden („INFO WRONG PARAMETER“ ([☞ 2.8.2, Seite 109](#))).

ANTWORT (Slave->Master): Falls möglich, wird die Zeit zurückgegeben die das Modul voraussichtlich für die Bewegung braucht. Wenn keine Berechnung der Zeit möglich sein sollte, wird die Anfrage beim Erfolg mit „OK“ (0x4F4B) bestätigt und das Modul führt die Bewegung aus.

HINWEIS

Sollte das Verfahrprofil auf „Sprung“ konfiguriert sein, wird das ruckbegrenzte Verfahrprofil verwendet. Bei Verfahrprofil „Sprung“ würde der Antrieb zu heftige Bewegungen ausführen.

SONSTIGES: Spontanantwort wird beim Erreichen der Position „*CMD POS REACHED*“ ([☞ 2.3.3, Seite 56](#)) oder bei Blockierung der Positionsfahrt „*MOVE BLOCKED*“ ([☞ 2.3.2, Seite 55](#)) erfolgen.

Alle Parameter sind in der angegebenen Reihenfolge zu übermitteln. Soll nur der Strom vorgegeben werden, so muss zwingend die Position, Geschwindigkeit und Beschleunigung mit angegeben werden. Nachfolgende Parameter müssen nicht mit übermittelt werden. Alle Parameter bleiben bis zum Neustart oder einer Änderung dieser Parameter erhalten.

HINWEIS

Eine neue Positions vorgabe während der Bewegung kann zu einem kurzfristigen Aussetzen der Bewegung führen. Möchte man während der Bewegung neue Positionen vorgeben um z.B. Kurvenbahnen abzufahren oder eigene Bahninterpolatoren verwenden, so ist der Befehl „*MOVE POS TIME*“ ([☞ 2.2.5, Seite 39](#)) zu verwenden.

2.2.4 MOVE POS REL (0xB8)

KOMMANDO CODE: 0xB8

BESCHREIBUNG: Bewegt das Modul um eine festgelegte Strecke. Die Positionsänderung wird im konfigurierten *Einheitensystem* ([☞ 1.5, Seite 23](#)) vorgegeben. Für die Positionsfahrt wird das *konfigurierte Verfahrprofil* ([☞ 3.2.5, Seite 136](#)) zugrundegelegt.

PARAMETER (Master -> Slave)*:

- *Positionsänderung* ([☞ 2.2.17, Seite 51](#)) (optional), im konfigurierten Einheitensystemwerden)
- *Geschwindigkeit* ([☞ 2.2.11, Seite 47](#)) (optional), welche für die Positionsfahrt verwendet wird. Bei Verfahrprofil „No Ramp“ ([☞ 3.2.5, Seite 136](#)) nicht relevant.
- *Beschleunigung* ([☞ 2.2.12, Seite 48](#)) (optional), welche für die Positionsfahrt verwendet wird. Bei Verfahrprofil „No Ramp“ ([☞ 3.2.5, Seite 136](#)) nicht relevant.
- *Strom* ([☞ 2.2.14, Seite 49](#)) (optional), welcher bei der Positions fahrt nichtüberschritten werden darf. Sollte die Reglerstruktur „CURRENT SPEED“ ([☞ 3.2.4, Seite 133](#)) aktiv sein, muss dieser Wertübermittelt werden (da Ruck benötigt wird). Der Wert muss mindestens „0“ sein, ansonsten wird „INFO WRONG PARAMETER“ ([☞ 2.8.2, Seite 109](#)) erzeugt.
- *Ruck* (optional), welcher für die Positionsfahrt verwendet wird. Sollte das Verfahrprofil ungleich „Ruckbegrenzt“ ([☞ 3.2.5, Seite 136](#)) sein kann dieser Wert nicht mit übermittelt werden („INFO WRONG PARAMETER“ ([☞ 2.8.2, Seite 109](#))).

HINWEIS

Sollte das Verfahrprofil auf „Sprung“ konfiguriert sein, wird das ruckbegrenzte Verfahrprofil verwendet. Bei Verfahrprofil „Sprung“ würde der Antrieb zu heftige Bewegungen ausführen.

ANTWORT (Slave->Master): Falls möglich, wird die Zeit zurückgegeben die das Modul für die Bewegung voraussichtlich braucht. Wenn keine Berechnung der Zeit möglich sein sollte wird die Anfrage beim Erfolg mit „OK“ (0x4F4B) bestätigt und das Modul führt die Bewegung aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB8	0x00 0x00 0x20 0x41	Fahre um 10.0 [mm] weiter
S->M	0x05	0xB8	0xCD 0xCC 0x04 0x41	Werde hierzu 8.3 [s] voraussichtlich benötigen

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB8		Fahre um die zuletzt eingestellte Distanz weiter
S->M	0x05	0xB8	0xCD 0xCC 0x04 0x41	Werde hierzu 8.3 [s] voraussichtlich benötigen

Sonstiges: Spontanantwort wird beim Erreichen der Position „CMD POS REACHED“ ([☞ 2.3.3, Seite 56](#)) oder bei Blockierung der Positions fahrt „MOVE BLOCKED“ ([☞ 2.3.2, Seite 55](#)) erfolgen. Alle Parameter sind in der angegebenen Reihenfolge zu übermitteln. Soll nur der Strom vorgegeben werden, so muss zwingend die Positionsänderung, Geschwindigkeit und Beschleunigung mit angegeben werden. Nachfolgende Parameter müssen nicht mit übermittelt werden. Alle Parameter bleiben bis zum Neustart oder einer Änderung dieser Parameter erhalten.

HINWEIS

Eine neue Positions vorgabe während der Bewegung kann zu einem kurzfristigen Aussetzen der Bewegung führen. Möchte man während der Bewegung neue Positionen vorgeben um z.B. Kurvenbahnen abzufahren oder eigene Bahninterpolatoren verwenden, so ist der Befehl „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)) zu verwenden.

2.2.5 MOVE POS TIME (0xB1)

KOMMANDO CODE: 0xB1

BESCHREIBUNG: Das Modul bewegt sich an eine festgelegte Position. Die Position wird im konfigurierten *Einheitensystem* ([☞ 1.5, Seite 23](#)) vorgegeben. Während der Bewegung können neue Positionen vorgegeben werden, die sofort angefahren werden. Bei der Berechnung der Bahn werden hierbei Soll-Geschwindigkeiten und Soll-Beschleunigung, sowie die aktuellen Ist-Geschwindigkeiten und Ist-Beschleunigungen berücksichtigt. Falls der Parameter Zeit mit angegeben wird, werden die Geschwindigkeit und Beschleunigung so angepasst, dass unter Berücksichtigung der vorgegebene Geschwindigkeits- und Beschleunigungsgrenzen die Position in der vorgegebenen Zeit erreicht wird.

PARAMETER (Master -> Slave):

- *Position* ([☞ 2.2.16, Seite 50](#)) (optional), im konfigurierten Einheitensystem. Position muss gegenüber der Startposition um mindestens *Positionsabweichung* ([☞ 3.2.4, Seite 133](#)) versetzt sein.
- *Geschwindigkeit* ([☞ 2.2.11, Seite 47](#)) (optional), welche nicht überschritten werden darf.
- *Beschleunigung* ([☞ 2.2.12, Seite 48](#)) (optional), welche nicht überschritten werden darf.
- *Strom* ([☞ 2.2.14, Seite 49](#)) (optional), welcher bei der Positions fahrt nichtüberschritten werden darf. Sollte die Reglerstruktur gleich „*CURRENT SPEED*“ ([☞ 3.2.4, Seite 133](#)) sein, kann dieser Wert nicht mit übermittelt werden („*INFO WRONG PARAMETER*“ ([☞ 2.8.2, Seite 109](#))).
- *Zeit* ([☞ 2.2.15, Seite 50](#)) (optional), in welcher die Positions fahrt abgeschlossen sein soll unter Einhaltung der Geschwindigkeit und Beschleunigung.

ANTWORT (Slave->Master): Die Zeit wird zurückgegeben, welche das Modul für die Bewegung braucht und das Modul führt die Bewegung aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB1	0x00 0x00 0x20 0x41	Fahre auf Position 10.0[mm]
S->M	0x05	0xB1	0x00 0x00 0xA0 0x40	Werde Position in 5.0[sek.] erreichen

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB1		Fahre auf zuletzt einge- stellte Position
S->M	0x05	0xB1	0x00 0x00 0xA0 0x40	Werde Position in 5.0[sek.] erreichen

SONSTIGES: Spontanantwort wird beim Erreichen der Position „CMD POS REACHED“ ([☞ 2.3.3, Seite 56](#)) oder bei Blockierung der Positionsfahrt „MOVE BLOCKED“ ([☞ 2.3.2, Seite 55](#)) erfolgen. Alle Parameter sind in der angegebenen Reihenfolge zu übermitteln. Soll nur der Strom vorgegeben werden, so muss zwingend Position, Geschwindigkeit und Beschleunigung mit angegeben werden. Nachfolgende Parameter müssen nicht mit übermittelt werden. Alle Parameter bleiben bis zum Neustart oder einer Änderung dieser Parameter erhalten. Das Verfahrprofil wird für diese Art der Bewegung temporär auf „Trapez“ ([☞ 3.2.5, Seite 136](#)) umgestellt. (Aufgrund von Rechenzeitproblemen bei Kurvenbahnen ist derzeit nur dieses Profil möglich)

HINWEIS

Eine erneute Positioneingabe während der Bewegung ist möglich. Die neue Bewegung wird anschließend mit den angegebenen Parametern und den vorhandene Ist-Geschwindigkeiten und Ist-Beschleunigungen neu berechnet. Es ist hiermit möglich Kurvenbahnen abzufahren.

HINWEIS

Wird das „Sprung“ ([☞ 3.2.5, Seite 136](#)) Profil eingeschalten, wird der interne Rampengenerator abgeschalten. Somit können von extern eigene Positionsrampen vorgegeben werden. Hierbei kann je nach Interpolationstakt der externen Vorgabe eine Anpassung der Reglerparameter ([☞ 3.2.4, Seite 133](#)) notwendig werden.

2.2.6 MOVE POS TIME REL (0xB9)

KOMMANDO CODE: 0xB9

BESCHREIBUNG: Das Modul bewegt sich um eine festgelegte Strecke. Die Positionsänderung wird im konfigurierten *Einheitensystem* ([☞ 1.5, Seite 23](#)) vorgegeben. Während der Bewegung können neue Positionen vorgegeben werden, die sofort angefahren werden. Bei der Berechnung der Bahn werden hierbei Soll-Geschwindigkeiten und Soll-Beschleunigung, sowie die aktuellen Ist-Geschwindigkeiten und Ist-Beschleunigungen berücksichtigt. Falls der Parameter Zeit mit angegeben wird, werden die Geschwindigkeit und Beschleunigung so angepasst, dass unter Berücksichtigung der vorgegebene Geschwindigkeits- und Beschleunigungsgrenzen die Position in der vorgegebenen Zeit erreicht wird.

PARAMETER (Master -> Slave):

- *Positionsänderung* ([☞ 2.2.17, Seite 51](#)) (optional), im konfigurierten Einheitensystem. Die Positionsänderung darf nicht kleiner als *Positionsabweichung* ([☞ 3.2.4, Seite 133](#)) sein.
- *Geschwindigkeit* ([☞ 2.2.11, Seite 47](#)) (optional), welche nicht überschritten werden darf.
- *Beschleunigung* ([☞ 2.2.12, Seite 48](#)) (optional), welche nicht überschritten werden darf.
- *Strom* ([☞ 2.2.14, Seite 49](#)) (optional), welcher bei der Positionsfahrt nichtüberschritten werden darf. Sollte die Reglerstruktur gleich „*CURRENT SPEED*“ ([☞ 3.2.4, Seite 133](#)) sein, kann dieser Wert nicht mit übermittelt werden („*INFO WRONG PARAMETER*“ ([☞ 2.8.2, Seite 109](#))).
- *Zeit* ([☞ 2.2.15, Seite 50](#)) (optional), in welcher die Positionsfahrt abgeschlossen sein soll unter Einhaltung der parametrierten Geschwindigkeits- und Beschleunigungsgrenzen.

ANTWORT (Slave->Master): Die Zeit wird zurückgegeben, welche das Modul für die Bewegung braucht. Das Modul führt Kommando aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB9	0x00 0x00 0x20 0x41	Fahre um 10.0 [mm] weiter
S->M	0x05	0xB9	0x00 0x00 0xA0 0x40	Werde hierzu voraussichtlich 5.0 [s] benötigen

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB9		Fahre um zuletzt eingesetzte Strecke weiter
S->M	0x05	0xB9	0x00 0x00 0xA0 0x40	Werde hierzu voraussichtlich 5.0 [s] benötigen

SONSTIGES: Spontanantwort wird beim Erreichen der Position „CMD POS REACHED“ ([☞ 2.3.3, Seite 56](#)) erfolgen. Alle Parameter sind in der angegebenen Reihenfolge zu übermitteln. Soll nur der Strom vorgegeben werden, so muss zwingend Positionsänderung, Geschwindigkeit und Beschleunigung mit angegeben werden. Nachfolgende Parameter müssen nicht mit übermittelt werden. Alle Parameter bleiben bis zum Neustart oder einer Änderung dieser Parameter erhalten. Das Verfahrprofil wird für diese Art der Bewegung temporär auf „Trapez“ ([☞ 3.2.5, Seite 136](#)) (Aufgrund von Rechenzeitproblemen bei Kurvenbahnen ist derzeit nur dieses Profil möglich.)

HINWEIS

Eine erneute Eingabe der Positionsänderung während der Bewegung ist möglich. Die neue Bewegung wird anschließend mit den angegebenen Parametern und den vorhandenen Ist-Geschwindigkeiten und Ist-Beschleunigungen neu berechnet. Es ist hiermit möglich Kurvenbahnen abzufahren.

HINWEIS

Wird das „Sprung“ ([☞ 3.2.5, Seite 136](#)) Profil eingeschalten, wird der interne Rampengenerator abgeschalten. Somit können von extern eigene Positionsrampen vorgegeben werden. Hierbei kann je nach Interpolationstakt der externen Vorgabe eine Anpassung der Reglerparameter ([☞ 3.2.4, Seite 133](#)) notwendig werden.

2.2.7 MOVE POS LOOP (0xBA)

KOMMANDO CODE: 0xBA

BESCHREIBUNG: Variante des Kommandos *MOVE POS*

(☞ [2.2.3, Seite 35](#)). Das Modul führt eine zyklische Bewegung zwischen der Startposition (aktuelle Position) und der festgelegten Position aus.

2.2.8 MOVE CUR (0xB3)

KOMMANDO CODE: 0xB3

BESCHREIBUNG: Eine Stromfahrt wird ausgeführt.

PARAMETER (Master -> Slave):

- Strom ([☞ 2.2.14, Seite 49](#)) im konfigurierten *Einheitensystem* ([☞ 1.5, Seite 23](#)).

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich.
Modul führt Kommando aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB3	0x00 0x00 0x60 0x40	Führe Stromfahrt mit 3.5 [A] aus
S->M	0x03	0xB3	0x4F 0x4B	

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB3		Führe Stromfahrt mit zuletzt eingestelltem Strom aus
S->M	0x03	0xB3	0x4F 0x4B	

SONSTIGES: Spontanmeldung („CMD MOVE BLOCKED“) ([☞ 2.3.2, Seite 55](#)) kann erfolgen.

HINWEIS

Aufgrund der verwendeten Reglerstrukturen kann das Modul „durchgehen“. Überschreitet das Modul die konfigurierte maximal Geschwindigkeit ([☞ 3.2.1, Seite 119](#)) kommt es aus Sicherheitsgründen zu einem „ERROR TOW“ ([☞ 2.8.2, Seite 109](#)).

HINWEIS

Bei eingestellter Reglerstruktur ([☞ 3.2.4, Seite 133](#)) „CASCADE“ wird dieser Befehl nicht benötigt. Alle Bewegungsarten arbeiten mit einstellbarem ([☞ 2.2.14, Seite 49](#)) unterlagertem Stromregler.

2.2.9 MOVE VEL (0xB5)

KOMMANDO CODE: 0xB5

BESCHREIBUNG: Eine Geschwindigkeitsfahrt wird ausgeführt.

PARAMETER (Master -> Slave):

- *Geschwindigkeit* ([2.2.11, Seite 47](#)) im konfigurierten *Einheitensystem* ([1.5, Seite 23](#)).
- *Strom* ([2.2.14, Seite 49](#)) (optional), welcher bei der Geschwindigkeitsfahrt nicht überschritten werden darf. Erzeugt bei Controller Struktur „*CURRENT SPEED*“ ([3.2.4, Seite 133](#)) „*INFO WRONG PARAMETER*“ ([2.8.2, Seite 109](#)).

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich. Modul führt Kommando aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB5	0x9A 0x99 0x31 0x41	Führe Geschwindigkeitsfahrt mit 11.1 [mm/s] aus
S->M	0x03	0xB5	0x4F 0x4B	

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x05	0xB5	0x9A 0x99 0x31 0x41	Führe Geschwindigkeitsfahrt mit zuletzt eingestellter Geschwindigkeit aus
S->M	0x03	0xB5	0x4F 0x4B	

SONSTIGES: Sonstiges: Spontanmeldung „*CMD MOVE BLOCKED*“ ([2.3.2, Seite 55](#)) kann erfolgen, wenn das Modul während der Fahrt blockiert wurde.

2.2.10 MOVE GRIP (0xB7)

KOMMANDO CODE : 0xB7

Beschreibung: Eine "Greiffahrt" wird ausgeführt.

PARAMETER (Master -> Slave):

- Strom ([2.2.14, Seite 49](#)) im konfigurierten *Einheitensystem* ([1.5, Seite 23](#)).
- max. Geschwindigkeit ([2.2.11, Seite 47](#)) (optional), die während der Greiffahrt nicht überschritten werden darf im konfigurierten *Einheitensystem* ([1.5, Seite 23](#)).

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich.

Modul führt Kommando aus.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x05	0xB7	0x00 0x00 0x60 0x40	Führe "Greiffahrt" mit 3.5[A] aus
S->M	0x03	0xB7	0x4F 0x4B	

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0xB7		Führe "Greiffahrt" mit zuletzt eingestelltem Strom aus
S->M	0x03	0xB7	0x4F 0x4B	

SONSTIGES: Spontanmeldung („CMD MOVE BLOCKED“ ([2.3.2, Seite 55](#)) kann erfolgen, wenn die Fahrt blockiert wurde (ein Gegenstand wurde gegriffen)).

HINWEIS

Der Befehl funktioniert nur ordnungsgemäß, wenn ein unterlagerter Stromregler ([3.2.4, Seite 133](#)) zur Verfügung steht.

HINWEIS

Greifvorgänge lassen sich ebenfalls mit einer Geschwindigkeitsfahrt ([2.2.9, Seite 45](#)) bzw. Positionsfahrtfahrt ([2.2.3, Seite 35](#)) mit unterlagertem Stromregler ([3.2.4, Seite 133](#)) realisieren.

2.2.11 SET TARGET VEL (0xA0)

KOMMANDO CODE: 0xA0

BESCHREIBUNG: Der Parameter Geschwindigkeit wird hiermit gesetzt.

PARAMETER (Master -> Slave):

- Geschwindigkeit im vorgegeben *Einheitensystem* ([☞ 1.5, Seite 23](#)).

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA0	0x33 0x33 0x43 0x41	Setze Geschwindigkeit auf 12.2 [mm/s]
S->M	0x03	0xA0	0x4F 0x4B	

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichen Schreiben erhalten, bis das Modul neu gestartet, oder dieser Wert erneut geändert wird. Der Wert wird erst für den nächsten Bewegungsbefehl übernommen und ist nicht sofort aktiv.

2.2.12 SET TARGET ACC (0xA1)

KOMMANDO CODE: 0xA1

BESCHREIBUNG: Der Parameter Beschleunigung wird hiermit gesetzt.

PARAMETER (Master -> Slave):

- Beschleunigung im vorgegebenen *Einheitensystem* ([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA1	0x00 0x00 0xF0 0xA2	Setze Beschleunigung [mm/s ²]
S->M	0x03	0xA1	0x4F 0x4B	

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichen Schreiben erhalten, bis das Modul neu gestartet, oder dieser Wert erneut geändert wird. Der Wert wird erst für den nächsten Bewegungsbefehl übernommen und ist nicht sofort aktiv.

2.2.13 SET TARGET JERK (0xA2)

KOMMANDO CODE: 0xA2

BESCHREIBUNG: Der Parameter Ruck wird hiermit gesetzt.

PARAMETER (Master -> Slave): Ruck im vorgegebenen *Einheitensystem* ([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA2	0x00 0x00 0x7A 0x44	Setze Ruck [mm/s ³]
S->M	0x03	0xA2	0x4F 0x4B	

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichen Schreiben erhalten, bis das Modul neu gestartet, oder dieser Wert erneut geändert wird. Der Wert wird erst für den nächsten Bewegungsbefehl übernommen und ist nicht sofort aktiv.

2.2.14 SET TARGET CUR (0xA3)

KOMMANDO CODE: 0xA3

BESCHREIBUNG: Der Parameter Strom wird hiermit gesetzt.

PARAMETER (Master -> Slave): *Strom* im vorgegeben
Einheitensystem ([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA3	0xCD 0xCC 0x2C 0x40	Setze Strom auf 2.7 [A]
S->M	0x03	0xA3	0x4F 0x4B	

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichem Setzen erhalten bis das Modul neu gestartet oder dieser Wert erneut geändert wird. Der Wert wird sofort übernommen und ist auch sofort aktiv.

HINWEIS

Hiermit können während der Bewegung die Stromsollvorgaben geändert werden. Es ist somit bei Greifern möglich einen bereits gegriffenen Gegenstand „nachzugreifen“ (Greifkraft erhöhen bzw. verringern).

HINWEIS

Wird bei Bewegungen nur dann berücksichtigt, wenn die Reglerstruktur dies zulässt. Reglerstruktur ([☞ 3.2.4, Seite 133](#)) „CURRENT SPEED“ lässt keine unterlagerte Stromregelung zu.

2.2.15 SET TARGET TIME (0xA4)

KOMMANDO CODE: 0xA4

BESCHREIBUNG: Der Parameter „Zeit“ für den nächsten Befehl „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)) oder „MOVE POS TIME' REL“ wird hiermit gesetzt.

PARAMETER (Master -> Slave):

- Zeit im vorgegeben Einheitensystem ([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA4	0x66 0x66 0x96 0x40	Setze Zeit auf 4.7 [s]
S->M	0x03	0xA4	0x4F 0x4B	

SONSTIGES: Gilt nur für das nächste Kommando „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)), „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)).

Nach Senden eines der zuvor genannten Kommandos wird dieser Wert wieder gelöscht.

2.2.16 SET TARGET POS (0xA6)

KOMMANDO CODE: 0xA6

BESCHREIBUNG: Der Parameter „Position“ für den nächsten „MOVE POS“ ([☞ 2.2.3, Seite 35](#)) oder „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)) Befehl wird hiermit gesetzt.

PARAMETER (Master -> Slave):

- Position im vorgegeben Einheitensystem ([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA6	0x00 0x00 0xc8 0x43	Setze Position auf 400,0 [°]
S->M	0x03	0xA6	0x4F 0x4B	„OK“

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichen Schreiben erhalten, bis das Modul neu gestartet, oder dieser Wert erneut geändert wird. Der Wert wird erst für den nächsten Bewegungsbefehl übernommen und ist nicht sofort aktiv.

2.2.17 SET TARGET POS REL (0xA7)

KOMMANDO CODE: 0xA7

BESCHREIBUNG: Der Parameter „Relativposition“ für das nächste „MOVE POS REL“ ([☞ 2.2.4, Seite 37](#)) oder „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)) Kommando wird hiermit gesetzt.

PARAMETER (Master -> Slave):

- *relativ Position* im vorgegeben *Einheitensystem*
([☞ 1.5, Seite 23](#))

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x05	0xA7	0x00 0x00 0xc8 0x43	Setze Relativposition auf 400,0 [°]
S->M	0x03	0xA7	0x4F 0x4B	„OK“

SONSTIGES: Dieser Wert bleibt nach einmaligem erfolgreichen Schreiben erhalten, bis das Modul neu gestartet, oder dieser Wert erneut geändert wird. Der Wert wird erst für den nächsten Bewegungsbefehl übernommen und ist nicht sofort aktiv.

2.2.18 CMD STOP (0x91)

KOMMANDO CODE: 0x91

BESCHREIBUNG: Das Modul wird abgebremst und in der aktuellen Position gehalten. Bei Modulen mit entsprechend *konfigurierter Haltebremse* ([3.2.7, Seite 150](#)) fällt diese ein, ansonsten wird das Modul aktiv geregelt.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0x91		
S->M	0x03	0x91	0x4F 0x4B	

SONSTIGES: Kann der Befehl nicht ordnungsgemäß ausgeführt werden, so wird dieser mit einer *Fehlermeldung* ([2.8.2, Seite 109](#)) beantwortet. z.B. Modul soll in aktiver Regelung bleiben, kann dies aber nicht, da ein schwerwiegender Fehler anliegt.

HINWEIS

Bei Modulen ohne Bremse bzw. entsprechend konfigurierter Haltebremse ([3.2.7, Seite 150](#)) wird der maximal zulässige Strom für die Regelung auf den Nominalstrom ([3.2.1, Seite 119](#)) begrenzt um eine Überhitzung des Motors zu vermeiden. Das Modul kann daher evtl. „durchsacken“.

2.2.19 CMD FAST STOP (0x90)

KOMMANDO CODE: 0x90

BESCHREIBUNG: Ist eine Bremse vorhanden und entsprechend konfiguriert ([☞ 3.2.7, Seite 150](#)), fällt diese sofort ein, und die Motorphasen werden kurzgeschlossen. Motor wird „stromlos geschaltet“.

PARAMETER (Master -> Slave): Keine

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0x90		
S->M	0x02	0x88	0xD9	Schnellstop ausgeführt

ANTWORT (Slave->Master): Fehlermeldung „ERROR FAST STOP“ ([☞ 2.8.2, Seite 109](#)) wird ausgelöst.

SONSTIGES: Kann nur durch „CMD ACK“ ([☞ 2.8.1, Seite 106](#)) wieder zurückgesetzt werden.

	! WARNUNG	
Verletzungsgefahr! Bei Modulen ohne Bremse kann das Modul „durchsacken“, da der Motor beim Schnellstop stromlos geschaltet wird!		

HINWEIS

Diese Art des Anhaltens führt zu einem starken mechanischen Verschleiß der Bremse.

2.3 Spontanmeldungen

Bei bestimmten Ereignissen kann sich das Modul selbständig melden. Es wird eine sog. „Spontanmeldung“ ausgelöst. Diese Meldungen werden über den Standard *Datenrahmen* ([☞ 1.4.2, Seite 15](#)) gesendet. (D-Len, CmdCode, Parameter). Es ist möglich Spontanmeldungen zu *deaktivieren* ([☞ 2.3.7, Seite 58](#)).

HINWEIS

Bei Profibus wird der MsgCount bei solchen Meldungen nicht erhöht, da von der Steuerung keine Daten angefordert wurden.

2.3.1 CMD INFO (0x8A)

KOMMANDO CODE: 0x8A

BEISPIEL:

	D-Len	Cmd	Param	
S->M	0x02	0x8A	0x10	„INFO TIMEOUT“ (☞ 2.8.2, Seite 109)

BESCHREIBUNG: Das Modul sendet eine Informationsnachricht.

BEMERKUNG: Bei behobenen Fehlern und einem Neustart des Moduls werden ebenfalls „Info Nachrichten“ versendet.

(„INFO BOOT“ ([☞ 2.8.2, Seite 109](#)), „INFO NO ERROR“
([☞ 2.8.2, Seite 109](#))

2.3.2 CMD MOVE BLOCKED (0x93)

KOMMANDO CODE: 0x93

BESCHREIBUNG: Der aktuelle Bewegungsbefehl wurde unterbrochen. Der Motor war kurzzeitig blockiert, bzw. ist immer noch blockiert.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Aktuelle *Position* im eingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#))

SONSTIGES: Der Motor gilt als blockiert, wenn alle der folgende Voraussetzungen erfüllt sind:

- Der Motor dreht sich mit einer Geschwindigkeit unterhalb der *Bewegungsschwelle* ([☞ 3.2.6, Seite 146](#))
- der Zielstrom erreicht ist ($\pm 15\%$)
- der *Bremse-Timeout* ([☞ 3.2.7, Seite 150](#)) ist abgelaufen.

Siehe auch *Flag „Bewegung blockiert“* ([☞ 2.6.1, Seite 84](#)).

HINWEIS

Bei Erkennung der Blockade wird bei Sinuskommutierung der eingestellte Strom um Wurzel(2) verringert. (Formfaktor Effektivwert zu Spitzenwert bei Sinusförmigen Strömen.)

HINWEIS

Eine absolut sichere Erkennung, ob ein Gegenstand gegriffen wurde, ist hiermit nicht möglich.

2.3.3 CMD POS REACHED (0x94)

KOMMANDO CODE: 0x94

BESCHREIBUNG: Eine Positionsfahrt hat die Zielposition erreicht.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): *Position* (aktuell) in eingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#))

BEISPIEL:

	D-Len	Cmd	Param	
S->M	0x05	0x94	0xCD 0xCC 0x2C 0x40	Habe Position 2.7 [mm] erreicht.

SONSTIGES: Das Modul steht. Bei vorhandener Bremse fällt diese je nach *Konfiguration* ([☞ 3.2.7, Seite 150](#)) ein.

2.3.4 CMD ERROR (0x88)

KOMMANDO CODE: 0x88

BESCHREIBUNG: Ein schwerwiegender Fehler ist aufgetreten, welcher einen Benutzereingriff notwendig macht. Diese Fehler müssen mit „CMD ACK“ ([☞ 2.8.1, Seite 106](#)) quittiert werden. Das Modul ist nicht betriebsbereit. Der Motor ist stromlos geschaltet, und die Bremse eingefallen. Diese Fehlermeldungen werden regelmäßig alle 15 Sekunden vom Modul zur Steuerung gesendet, bis der Fehler quittiert ist.

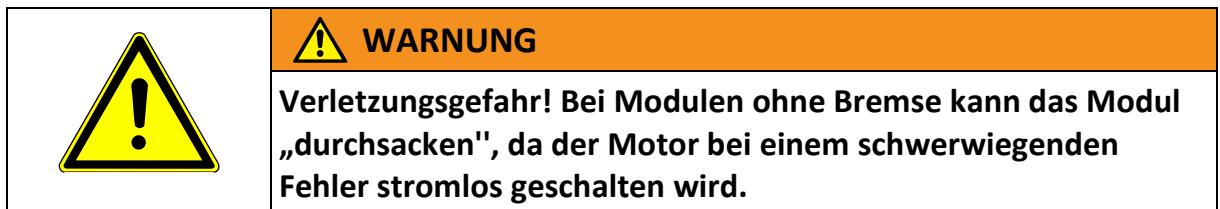
- Serielle Kommunikation: das erste Byte einer Nachricht ändert sich zu „0x03“
- CAN: die ersten drei Bit des Identifiers ergeben sich zu „0x3“
- Profibus: eine erweiterte Diagnose wird generiert

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): *Fehlercode* ([☞ 2.8.2, Seite 109](#))

BEISPIEL:

	D-Len	Cmd	Param	
S->M	0x02	0x88	0xDE	Fehler „ERROR CURRENT“ (☞ 2.8.2, Seite 109) ist aufgetreten.

**2.3.5 GET STATE (0x95)**

KOMMANDO CODE: 0x95

BESCHREIBUNG: Liefert den Status des Moduls, sowie einige weitere Informationen, wenn gewünscht. Das Modul kann diesen Status selbständig in regelmäßigen Abständen versenden.

Details siehe „GET STATE“ ([☞ 2.6.1, Seite 84](#)).

2.3.6 GET STATE AXIS (0x98)

KOMMANDO CODE: 0x98

BESCHREIBUNG: Liefert den Status einer Achse oder eines Roboters des Moduls, sowie einige weitere Informationen, wenn gewünscht. Das Modul kann diesen Status selbständig in regelmäßigen Abständen versenden. Details siehe „GET STATE AXIS“ ([☞ 2.6.2, Seite 87](#)).

2.3.7 CMD TOGGLE IMPULSE MESSAGE (0xE7)

KOMMANDO CODE: 0xE7

BESCHREIBUNG: Spontanmeldungen können hiermit aktiviert bzw. deaktiviert werden.

Ist die „sekundäre Schnittstelle“ ([☞ 1.6.2, Seite 25](#)) aktiv, wird hiermit der „Lauschbetrieb“ aktiviert bzw. deaktiviert.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Wird der Befehl mit „ON“ (0x4F 0x4E) bestätigt, sind Spontanmeldungen aktiv. Wird der Befehl mit „OFF“ (0x4F 0x46 0x46) bestätigt, sind Spontanmeldungen abgeschaltet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xE7		
S->M	0x04	0xE7	0x4F 0x46 0x46	Spontanmeldung deaktiviert.

SONSTIGES: Spontanmeldungen sind beim Neustart des Moduls immer aktiviert.

(Ausnahme *Serielle Kommunikation ohne Spontanmeldung* ([☞ 3.2.5, Seite 136](#))

HINWEIS

Spontannachrichten sollten nach Möglichkeit nicht deaktiviert werden. Dies kann evtl. bei Serielle Kommunikation notwendig werden, wenn sehr viele Module angesteuert werden und es gehäuft zu Kollisionen kommt. Speziell bei der Serielle Kommunikation ist es möglich Spontanmeldungen automatisch nach einem Neustart zu deaktivieren ([☞ 3.2.5, Seite 136](#)). Spontanmeldungen können dann jederzeit mit CMD TOGGLE IMPULSE MESSAGE aktiviert werden.

2.4 Einstellungen

2.4.1 SET CONFIG (0x81)

KOMMANDO CODE: 0x81

BESCHREIBUNG: Konfigurationsparameter im Modul werden gesetzt und *dauerhaft gespeichert* ([☞ 3.2, Seite 118](#)).

PARAMETER (Master -> Slave): Als Befehlsparameter wird übergeben, welche Konfigurationsparameter geschrieben werden sollen.

- EEPROM (0xFE) + EEPROM Struktur
Alle Konfigurationsparameter werden auf einmal geschrieben (Komplette EEPROM Struktur muss in den Daten übermittelt werden). Je nach Nutzer ([☞ 1.6.1, Seite 25](#)) kann es sein, dass einige Daten nicht geschrieben werden. Nach erfolgtem Schreiben bootet das Modul neu.

HINWEIS

Dieser Befehl sollte in eigenen Applikationen nicht eingesetzt werden, da die Struktur der zu sendenden Daten nicht bekannt ist.

HINWEIS

Das Modul muss hierzu gestoppt sein, und die Regelung muss deaktiviert sein. Z.B. vorher den Schnellstop ([☞ 2.2.19, Seite 53](#)) ausführen.

HINWEIS

Hierzu muss das Fragmentierungsprotokoll ([☞ 1.4.6, Seite 21](#)) benutzt werden (außer Serielle Kommunikation).

- Modul ID (0x01) + Daten (1 Byte)
Die ID des Moduls wird geändert. Gültige Werte (0 .. 255). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.

HINWEIS

Profibus unterstützt „Set Slave Address“ (SAP 55)

- Gruppe ID (0x02) + Daten (1 Byte)
Die Gruppe des Moduls wird geändert. Gültige Werte (0 .. 255). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.

HINWEIS

Bei Profibus wird hier „Real No Add Change“ gespeichert. Gruppen werden vollständig durch den SYNC/ FREEZE Mechanismus unterstützt.

- Baudrate Serielle Kommunikation (0x03) + Daten (2 Byte)
Die Baudrate der Seriellen Kommunikation kann verändert werden. Gültige Werte (1200, 2400, 4800, 9600, 19200, 38400). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.
- CAN Baudrate (0x04) + Daten (2 Byte)
Die CAN Baudrate kann eingestellt werden. Gültige Werte (50, 100, 125, 250, 500, 800, 1000). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.
- Kommunikation (0x05) + Daten (1 Byte)
Die Kommunikationsschnittstelle wird eingestellt. Gültige Werte: AUTO (0x00), seriell (0x01), CAN (0x02), Profibus DPV0 (0x03), seriell ohne Spontanmeldung (0x04). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.
- Einheitensystem (0x06) + Daten (1 Byte)
Das Einheitensystem kann umgestellt werden. Gültige Werte ([mm] = 0x00, [m] = 0x01, [Inch] = 0x02, [Rad] = 0x03, [Grad] = 0x04, [Intern] = 0x05, [μ m] Integer = 0x06, [μ Grad] Integer = 0x07, [μ Inch] Integer = 0x08, [Milligrad] Integer = 0x09). Die neuen Einstellungen werden sofort im EEPROM gespeichert, aber erst nach einem Neustart aktiviert.
- Max. Softwareendanschlag (0x07) + Daten (4 Byte)
Der obere Softwareendanschlag wird temporär geändert (*Einheitensystem* ([☞ 1.5, Seite 23](#)) beachten). Der übertragene Wert wird nicht ins EEPROM geschrieben. Die Einstellung ist sofort aktiv.
(Wird für SRU im Teach-Betrieb benötigt. Bei SRU für „NUTZER“ freigeschaltet).

HINWEIS

Diese Funktion ist ab Nutzer „Profi“ ([☞ 1.6.3, Seite 26](#)) freigeschaltet.

- Min. Softwareendanschlag (0x08) + Daten (4 Byte)
Der untere Softwareendanschlag wird temporär geändert (*Einheitensystem* ([☞ 1.5, Seite 23](#)) beachten). Der übertragene Wert wird nicht ins EEPROM geschrieben. Die Einstellung ist sofort aktiv.
(Wird für SRU im Teach-Betrieb benötigt. Bei SRU für „NUTZER“ freigeschaltet.)

HINWEIS

Diese Funktion ist ab Nutzer „Profi“ ([☞ 1.6.3, Seite 26](#)) freigeschaltet

- Getriebeübersetzung (0x18) + Daten (4 Byte Float)
Die *Getriebeübersetzung 1* ([☞ 3.2.2, Seite 124](#)) wird geändert (der Befehl gilt nicht für die *Integer-Einheitensysteme* ([☞ 1.5, Seite 23](#))). Der übertragene Wert wird ins EEPROM geschrieben und ist sofort aktiv.

HINWEIS

Diese Funktion ist ab Nutzer „Profi“ ([☞ 1.6.3, Seite 26](#)) freigeschaltet.

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn erfolgreich. Um erkennen zu können, welcher Parameter erfolgreich gesetzt wurde, wird im Anschluss an „OK“ der Parameter Code angehängt (1 Byte).

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x03	0x81	0x01 0x0C	Setze Modul ID auf 12
S->M	0x04	0x81	0x4F 0x4B 0x01	

SONSTIGES: Zum komfortablen Setzen der Konfigurationsparameter kann das mitgelieferte *Software Tool* ([☞ 3.2.7, Seite 150](#)) verwendet werden. Sollen alle Parameter auf einmal geschrieben werden, so ist dies nur beim Stillstand des Moduls möglich.

2.4.2 GET CONFIG (0x80)

KOMMANDO CODE: 0x80

BESCHREIBUNG: Verschiedene Konfigurationsparameter können aus dem Modul ausgelesen werden.

PARAMETER (Master -> Slave):

- Keiner

Es wird folgende Information zurückgeliefert:

- Modultyp im Klartext (8 Zeichen ASCII)
- Bestellnummer (UInt32)
- Firmware Version (UInt16)
- Protokoll Version (UInt16)
- Hardware Version (UInt16)
- Firmware Erstellungsdatum und -zeit (21 Zeichen ASCII)

HINWEIS

Module mit gerader Hardwareversion haben EEPROM verbaut.

Module mit ungerader Hardwareversion haben FRAM verbaut.

Siehe auch Pseudoabsolutwertgeber ([☞ 1.7, Seite 27](#))

- EEPROM (0xFE)

Alle Konfigurationsparameter werden auf einmal ausgelesen.

(PC-Parametriersoftware ([☞ 3.2.7, Seite 150](#)) nutzt diesen Subcode)

HINWEIS

Das Modul muss hierzu gestoppt und die Regelung deaktiviert sein (z.B. vorher Schnellstop ([☞ 2.2.19, Seite 53](#))).

HINWEIS

Dieser Befehl sollte in eigenen Applikationen nicht eingesetzt werden, da die Struktur der empfangenen Daten nicht bekannt ist.

HINWEIS

Hierzu muss das Fragmentierungsprotokoll ([☞ 1.4.6, Seite 21](#)) benutzt werden (außer Serielle Kommunikation).

- Modul ID (0x01)

Die ID des Moduls wird gelesen (1 Byte).

- Gruppe ID (0x02)

Die Gruppenzugehörigkeit des Moduls wird gelesen (1 Byte).

(zur Zeit besteht nur Bedeutung für Profibus ([☞ 1.4.5, Seite 19](#))
Set Salve Address)

- Serielle Baudrate (0x03)
Die Baudrate der Seriellen Kommunikation wird gelesen (2 Byte).
- CAN Baudrate (0x04)
Die CAN Baudrate wird gelesen (2 Byte).
- Kommunikation (0x05)
Die konfigurierte Kommunikationsschnittstelle wird ausgelesen (1 Byte). Gültige Werte: AUTO (0x00), Seriell (0x01), CAN (0x02), Profibus DPV0 (0x03), Serielle Kommunikation ohne Spontanmeldung (0x04).
- Einheitensystem (0x06)
Das konfigurierte Einheitensystem wird ausgelesen. Mögliche Werte ([mm] = 0x00, [m] = 0x01, [Inch] = 0x02, [Rad] = 0x03, [Grad] = 0x04, [Intern] = 0x05, [μm] Integer = 0x06, [μGrad] Integer = 0x07, [μInch] Integer = 0x08, [Milligrad] Integer = 0x09)
- Max. Softwareendanschlag (0x07)
Aktueller max. Softwareendanschlag wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Min. Softwareendanschlag (0x08)
Aktueller min. Softwareendanschlag wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Max. Geschwindigkeit (0x09)
Max. zulässige Geschwindigkeit des Moduls wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Max. Beschleunigung (0x0A)
Max. zulässige Beschleunigung des Moduls wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Max. Strom (0x0B)
Max. zulässiger Strom des Moduls wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Nom. Strom (0x0C)
Nom. Strom des Moduls wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Max. Ruck (0x0D)
Max. zulässiger Ruck des Moduls wird zurückgeliefert (4 Byte). [Einheitensystem \(☞ 1.5, Seite 23\)](#) beachten!
- Offset Phase A (0x0E)
Der Offset des Stromsensors A wird zurückgeliefert (2 Byte).
- Offset Phase B (0x0F)
Der Offset des Stromsensors B wird zurückgeliefert (2 Byte).

- Daten CRC (0x13)
Die CRC über die EEPROM Einträge ohne individuelle Daten (Seriennummer, Sensor spezifische Offsets (Strom, Resolver)) eines Moduls wird übertragen.
- Referenz Offset (0x14)
Der Referenz-Offset wird zurückgeliefert (4 Byte). *Einheitensystem* ([☞ 1.5, Seite 23](#)) beachten!
- Serien Nummer (0x15)
Die Seriennummer des Geräts wird zurückgeliefert (4 Byte).
- Bestellnummer (0x16)
Die Bestellnummer des Geräts wird zurückgeliefert (4 Byte).
- Error History (0xFD)
Ringspeicher der letzten 20 Fehler wird ausgelesen.

ANTWORT (Slave->Master): Parameter ID (um nachvollziehen zu können welche Daten angefordert wurden). Im Anschluss daran die angeforderten Daten.

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x02	0x80	0x06	Lese Einheitensystem
S->M	0x03	0x80	0x06 0x00	Einheitensystem ist [mm]

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0x80		Lese Modulinformation
S->M	0x28	0x80	0x50 0x52 0x2D 0x37 0x30 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x79 0x00 0x03 0x00 0x12 0x02 0x31 0x31 0x3A 0x32 0x32 0x3A 0x32 0x37 0x20 0x20 0x4A 0x75 0x6C 0x20 0x20 0x33 0x20 0x32 0x30 0x30 0x38	Modultyp „PR-70“, Be- stellnummer 0, Firmware Version 1.21, Protokoll Version 3, Hardware Ver- sion 5.30, Firmware er- stellt am „11:22:27 Jul 3 2008“

SONSTIGES: Zum komfortablen Lesen der Konfigurationsparameter kann das mitgelieferte *Software Tool* ([☞ 3.2.7, Seite 150](#)) verwendet werden. Sollen alle Parameter auf einmal gelesen werden, so ist dies nur beim Stillstand des Moduls möglich.

2.4.3 SET CONFIG EXT (0x83)

KOMMANDO CODE: 0x83

BESCHREIBUNG: Einzelne Konfigurationsparameter im Modul werden gesetzt und *dauerhaft gespeichert* ([☞ 3.2, Seite 118](#)). Das Kommando „SET CONFIG EXT“ ist eine Verallgemeinerung von *SET CONFIG* ([☞ 2.4.1, Seite 59](#)) und unterscheidet sich wie folgt von diesem:

- Pro Kommando kann immer nur ein einziger Konfigurationsparameter gezielt gesetzt werden.
- Die Konfigurationsparameter werden über einen 16-Bit breiten Konfigurationscode identifiziert.
- Die Bedeutung eines Konfigurationscodes hängt von dem angeschlossenen Gerät ab. Siehe hierzu entsprechende Ergänzungsanleitung.
- Der Wert eines zu setzenden Konfigurationsparameters kann geräteabhängig in verschiedenen Datenformaten übermittelt werden (siehe Definition vom „Datentyp“ unten).

PARAMETER (Master -> Slave):

- Konfigurationscode (2 Byte)
- Datentyp (1 Byte)
- Konfigurationsparameter (n Bytes)
- Achsindex (1 Byte) (optional)

Der „Konfigurationscode“ identifiziert den zu setzenden Konfigurationsparameter. Der Code ist gerätespezifisch. Eine Beschreibung der gültigen Konfigurationscodes findet sich in der jeweiligen Ergänzungsanleitung. Der optionale „Achsindex“ identifiziert den Index der Achse oder des Roboters, für den die Konfigurationsparameter gesetzt werden sollen. Wenn nicht angegeben, dann wird eine Standardeinstellung verwendet. Der „Datentyp“ ist wie folgt definiert:

DATENTYPEN:

Name	Wert	Beschreibung
DT_UNKNOW	0	Unbekannter Datentyp, darf nicht verwendet werden
DT_UINT8	1	Vorzeichenlose Ganzzahl mit 8 Bit (1 Byte)
DT_INT8	2	Vorzeichenbehaftete Ganzzahl mit 8 Bit (1 Byte)
DT_UINT16	3	Vorzeichenlose Ganzzahl mit 16 Bit (2 Byte)
DT_INT16	4	Vorzeichenbehaftete Ganzzahl mit 16 Bit (2 Byte)
DT_UINT32	5	Vorzeichenlose Ganzzahl mit 32 Bit (4 Byte)
DT_INT32	6	Vorzeichenbehaftete Ganzzahl mit 32 Bit (4 Byte)
DT_UINT64	7	Vorzeichenlose Ganzzahl mit 64 Bit (8 Byte)
DT_INT64	8	Vorzeichenbehaftete Ganzzahl mit 64 Bit (8 Byte)
DT_FLOAT	9	Fließkommazahl mit einfacher Genauigkeit (4 Byte)
DT_DOUBLE	10	Fließkommazahl mit doppelter Genauigkeit (8 Byte)
DT_STRING	11	„Stringlänge“ + ASCII-String, siehe unten (n Byte)

Die Anzahl n der Bytes des Konfigurationsparameters hängt vom „Datentyp“ ab. Beim Datentyp „DT_STRING“ ist im ersten Byte des Konfigurationsparameters die Anzahl der folgenden ASCII Zeichen enthalten.

Dieser zusätzliche Parameter „Stringlänge“ wird benötigt um entscheiden zu können, ob das letzte Byte einer SMP-Nachricht den optionalen Achsindex darstellt oder noch Teil des Strings ist.

ANTWORT (Slave->Master): wenn erfolgreich

- „OK“ (0x4F4B)
- *Konfigurationscode* (2 Byte)
- *Achsindex* (1 Byte) (optional)

Die zurückgelieferten Werte von „Konfigurationscode“ und ggf. „Achsindex“ entsprechen den angeforderten und erlauben so eine Zuordnung der Antwort zur Anfrage. Wenn die Anfrage einen „Achsindex“ enthielt, dann wird auch die Antwort ebenfalls mit „Achsindex“ versendet

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x08	0x83	0x12 0x07 0x0B 0x04 0x54 0x65 0x73 0x74	Setze Konfigurationscode 1810 (0x12 0x07) mit Datentyp DT_STRING (0x0b) und Stringlänge 4 (0x04) auf den Wert "Test" (0x54 0x65 0x73 0x74)
S->M	0x05	0x83	0x4F 0x4B 0x12 0x07	Erfolg (0x4f 0x4b) beim Setzen von Konfigurationscode 1810 (0x12 0x07)

SONSTIGES: Zum komfortablen Setzen der Konfigurationsparameter kann das mitgelieferte *Software Tool* ([☞ 3.2.7, Seite 150](#)) verwendet werden.

HINWEIS

Einige Konfigurationsparameter können nur geändert werden, wenn das Modul gestoppt ist, und die Regelung deaktiviert ist. Vorher also ggf. Schnellstop ([☞ 2.2.19, Seite 53](#)) auslösen. Im Fehlerfall wird mit „INFO NO RIGHTS“ ([☞ 2.8.2, Seite 109](#)) geantwortet.

HINWEIS

Einige Konfigurationsparameter können nicht geändert werden. Wenn auf solche „Read-Only“ Daten mit „SET CONFIG EXT“ zugegriffen wird, dann wird mit Fehlercode „INFO NO RIGHTS“ ([☞ 2.8.2, Seite 109](#)) geantwortet.

2.4.4 GET CONFIG EXT (0x82)

KOMMANDO CODE: 0x82

BESCHREIBUNG: Bestimmte Konfigurationsparameter können aus dem Modul ausgelesen werden. Das Kommando „GET CONFIG EXT“ ist eine Verallgemeinerung von *GET CONFIG* ([☞ 2.4.2, Seite 62](#)) und unterscheidet sich wie folgt von diesem:

- Pro Kommando kann immer nur ein einzelner Konfigurationsparameter gezielt abgefragt werden.
- Die Konfigurationsparameter werden über einen 16-Bit breiten Konfigurationscode identifiziert.
- Die Bedeutung eines Konfigurationscodes hängt von dem angeschlossenen Gerät ab. Siehe hierzu die jeweilige Ergänzungsanleitung.
- Der Wert eines zurückgelieferten Konfigurationsparameters kann geräteabhängig in verschiedenen *Datenformaten* ([☞ 2.4.3, Seite 65](#)).

PARAMETER (Master -> Slave):

- Konfigurationscode (2 Byte)
- Achsindex (1 Byte) (optional)

Der „Konfigurationscode“ identifiziert den angefragten Konfigurationsparameter. Der Code ist gerätespezifisch. Eine Beschreibung der gültigen Konfigurationscodes findet sich in der jeweiligen Gerätebeschreibung. Der „Achsindex“ identifiziert den Index der Achse oder des Roboters, von dem die Konfigurationsparameter gelesen werden sollen.

ANTWORT (Slave->Master):

- Konfigurationscode (2 Byte)
- Datentyp (1 Byte)
- Konfigurationsparameter (n Bytes)
- Achsindex (1 Byte) (optional)

Die zurückgelieferten Werte von „Konfigurationscode“ und ggf. „Roboterindex“ entsprechen den angeforderten und erlauben so eine Zuordnung der Antwort zur Anfrage. Der „Datentyp“ ist entsprechend *SET CONFIG EXT* ([☞ 2.4.3, Seite 65](#)) definiert. Die Anzahl n der Bytes des Konfigurationsparameters hängt vom „Datentyp“ ab. Beim Datentyp „DT_STRING“ ist im ersten Byte des Konfigurationsparameters die Anzahl der folgenden ASCII Zeichen enthalten. 1

Dieser zusätzliche Parameter „Stringlänge“ wird benötigt um entscheiden zu können ob das letzte Byte einer SMP-Nachricht den optionalen Achsindex darstellt oder noch Teil des Strings ist. Wenn die Anfrage einen „Achsindex“ enthält, dann wird auch die Antwort ebenfalls mit „Achsindex“ versendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x03	0x82	0x72 0x00 0x00	Lese den Konfig.-Parameter Konfigurationscode=114 (0x072 0x00)
S->M	0x17	0x82	0x72 0x00 0x0B 0x13 0x37 0x32 0x30 0x31 0x2D 0x30 0x30 0x37 0x36 0x2D 0x34 0x35 0x36 0x34 0x2D 0x33 0x34 0x62 0x35 0x00	Gelesener Konfig.-Parameter: Konfigurationscode=114 (0x072 0x00); Daten- typ=DT_STRING (0x0B); Stringlänge=19 (0x13); Daten="7201-0076-4564- 34b5" (0x37 0x32, .., 0x35);

SONSTIGES: Zum komfortablen Lesen der Konfigurationsparameter kann das mitgelieferte *Software Tool* ([3.2.7, Seite 150](#)) verwendet werden.

2.4.5 CMD_DEFAULT (0x8C)

KOMMANDO CODE: 0x8C

BESCHREIBUNG: Modul wird auf Werkseinstellungen gesetzt .

PARAMETER (Master -> Slave): keine

ANTWORT (Slave->Master): Befehl wird mit "OK" (0x4F4B) bestätigt. Werkseinstellungen sind nach Neustart aktiviert.

2.5 Befehle für Interne Programmierung

Die Module können mittels interner Ablaufprogrammen *ohne externe Steuerung* ([☞ 3.2.5, Seite 136](#)) betrieben werden. Zudem besteht die Möglichkeit über die digitalen Ein- bzw. Ausgänge vorher abgelegte Programme ablaufen zu lassen und somit das Modul nur über digitale I/Os zu betreiben. Hierzu müssen die *Ein-
(☞ 3.2.5, Seite 136) bzw. Ausgänge* ([☞ 3.2.5, Seite 136](#)) entsprechend konfiguriert werden.

Darüber hinaus kann das Modul mit 2 Byte bedient werden, indem vorher einprogrammierte komplexere Abläufe über *spezielle Befehle* ([☞ 2.5.5, Seite 73](#)) gestartet werden.

Um die Module programmieren zu können, ist eine *Fragmentierung* ([☞ 1.4.6, Seite 21](#)) der Daten notwendig. Am einfachsten lässt sich das Modul mit dem mitgelieferten *Software Tool* ([☞ 3.2.7, Seite 150](#)) über Serielle Schnittstelle programmieren.

Programme können nur „am Stück“ geschrieben werden. Will man einen Teil ändern, so ist das gesamte Programm auszulesen, die entsprechende Korrektur durchzuführen und das gesamte Programm wieder in das Modul zu schreiben.

Bedingte Sprünge, Schleifen und Verzweigungen sind nicht möglich.

(Die Programmdaten werden im EEPROM möglichst dicht gepackt und mit Checksummen versehen, was eine Änderung mitten in der komplexen Datenstruktur verbietet)

2.5.1 SET PHRASE (0xC0)

KOMMANDO CODE: 0xC0

BESCHREIBUNG: Wird dieses Kommando einem beliebigen Datenrahmen vorangestellt, so wird dieser Datenrahmen im nichtflüchtigen Speicher abgelegt. Somit können alle verfügbaren Kommandos zur Programmierung verwendet werden. Ein abgelegter Datenrahmen wird in diesem Modus auch als „Satz“ (Phrase) bezeichnet. Die Sätze werden automatisch erhöht bis das Kommando SET PHRASE ohne Parameter (Datenrahmen) aufgerufen wird. Dies ist das Zeichen für „Programmierung beendet“.

PARAMETER (Master -> Slave):

- kein
Programmierung beendet.
- gültiger Datenrahmen
Satz wird abgespeichert und Satzzähler erhöht.

ANTWORT (Slave->Master): „OK“ (0x4F4B) wenn der Satz erfolgreich abgespeichert wurde.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x07	0xC0	0x05 0xB1 0x00 0x00 0x60 0x40	Programmiere fahre an Position 3.5 [mm]
S->M	0x03	0xC0	0x06 0x00	

SONSTIGES: Die Programmierung erfolgt am einfachsten mit Hilfe der mitgelieferten Software ([3.2.7, Seite 150](#)).

HINWEIS

Für die Programmierung muss das Fragmentierungsprotokoll ([1.4.6, Seite 21](#)) benutzt werden (außer Serielle Kommunikation).

2.5.2 GET PHRASES (0xC2)

KOMMANDO CODE: 0xC2

BESCHREIBUNG: Liest das komplette Programm aus dem Modul aus

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Erster Datenrahmen enthält Anzahl der Sätze des Programms (UInt16). Anschließend wird der Inhalt des Programms Satz für Satz übertragen.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xC2		
S->M	0x03	0xC2	0x05 0x00	Programm enthält 5 Sätze
S->M	0x05	0xB1	0x00 0x00 0x60 0x40	Satz 0 enthält einen „fahre an Position 3.5 [mm]“ Befehl
S->M	0x05	0xB1	0x00 ...	Sätze bis Programmspeicher Ende

SONSTIGES: Sobald der Befehl einmal angestoßen wurde, wird Zeile für Zeile des Programms gesendet. Die Verwaltung von Programmen erfolgt am einfachsten mit Hilfe der mitgelieferten Software ([☞ 3.2.7, Seite 150](#)).

HINWEIS

Hierzu muss das Fragmentierungsprotokoll ([☞ 1.4.6, Seite 21](#)) benutzt werden (außer Serielle Kommunikation).

2.5.3 PRG EXE (0xCF)

KOMMANDO CODE: 0xCF

BESCHREIBUNG: Ein Programm wird ausgeführt

PARAMETER (Master -> Slave): kein. Programmausführung beginnt bei Zeile „0“

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet und der entsprechende Satz ausgeführt.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xCF		Starte Programm bei Zeile „0“
S->M	0x01	0xCF		Bestätigung vom Modul
S->M	0x04	0xC1	0x00 0x00 0xB0	Zeile „0“ enthält einen „MOVE POS“ Befehl
S->M	0x04	0xC1	0x01 ...	Sätze bis Programm Ende

2.5.4 EXE PHRASE (0xC1)

KOMMANDO CODE: 0xC1

BESCHREIBUNG: Zeile aus dem abgespeicherten Programm wird ausgeführt.

PARAMETER (Master -> Slave): *Programmzeile* (UInt16) welche ausgeführt werden soll.

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x03	0xC1	0x02 0x00	Führe Satz „2“ aus.
S->M	0x04	0xC1	0x02 0x00 0xB0	Satz 2 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der angegebenen Programmzeile „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt.

2.5.5 EXE PHRASE0 (0xD0)

KOMMANDO CODE: 0xD0

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „0“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD0		Führe Satz „0“ aus.
S->M	0x04	0xD0	0x00 0x00 0xB0	Satz 0 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 0 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.6 EXE PHRASE1 (0xD1)

KOMMANDO CODE: 0xD1

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „1“ abrufen zu können.

BESCHREIBUNG: Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD1		Führe Satz „1“ aus.
S->M	0x04	0xD1	0x01 0x00 0xB0	Satz 1 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 1 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.7 EXE PHRASE2 (0xD2)

KOMMANDO CODE: 0xD2

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „2“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD2		Führe Satz „2“ aus.
S->M	0x04	0xD2	0x02 0x00 0xB0	Satz 2 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 2 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.8 EXE PHRASE3 (0xD3)

KOMMANDO CODE: 0xD3

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „3“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD3		Führe Satz „3“ aus.
S->M	0x04	0xD3	0x03 0x00 0xB0	Satz 3 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 3 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([☞ 3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.9 EXE PHRASE4 (0xD4)

KOMMANDO CODE: 0xD4

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „4“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD4		Führe Satz „4“ aus.
S->M	0x04	0xD4	0x04 0x00 0xB0	Satz 4 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 4 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([☞ 3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.10 EXE PHRASE5 (0xD5)

KOMMANDO CODE: 0xD5

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „5“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD5		Führe Satz „5“ aus.
S->M	0x04	0xD5	0x05 0x00 0xB0	Satz 5 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 5 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.11 EXE PHRASE6 (0xD6)

KOMMANDO CODE: 0xD6

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „6“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD6		Führe Satz „6“ aus.
S->M	0x04	0xD6	0x06 0x00 0xB0	Satz 6 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 6 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden. Dieser Befehl mit der entsprechenden Satznummer wird intern generiert, wenn die *digitalen Eingänge* ([3.2.5, Seite 136](#)) zum Betreiben des Moduls verwendet werden.

2.5.12 EXE PHRASE7 (0xD7)

KOMMANDO CODE: 0xD7

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „7“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD7		Führe Satz „7“ aus.
S->M	0x04	0xD7	0x07 0x00 0xB0	Satz 7 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 7 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.13 EXE PHRASE8 (0xD8)

KOMMANDO CODE: 0xD8

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „8“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD8		Führe Satz „8“ aus.
S->M	0x04	0xD8	0x08 0x00 0xB0	Satz 8 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 8 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.14 EXE PHRASE9 (0xD9)

KOMMANDO CODE: 0xD9

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „9“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xD9		Führe Satz „9“ aus.
S->M	0x04	0xD9	0x09 0x00 0xB0	Satz 9 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 9 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.15 EXE PHRASE10 (0xDA)

KOMMANDO CODE: 0xDA

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „10“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDA		Führe Satz „10“ aus.
S->M	0x04	0xDA	0x0A 0x00 0xB0	Satz 10 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 10 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.16 EXE PHRASE11 (0xDB)

KOMMANDO CODE: 0xDB

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „11“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDB		Führe Satz „11“ aus.
S->M	0x04	0xDB	0x0B 0x00 0xB0	Satz 11 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 11 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.17 EXE PHRASE12 (0xDC)

KOMMANDO CODE: 0xDC

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „12“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDC		Führe Satz „12“ aus.
S->M	0x04	0xDC	0x0C 0x00 0xB0	Satz 12 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 12 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.18 EXE PHRASE13 (0xDD)

KOMMANDO CODE: 0xDD

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „13“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDD		Führe Satz „13“ aus.
S->M	0x04	0xDD	0x0D 0x00 0xB0	Satz 13 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 13 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.19 EXE PHRASE14 (0xDE)

KOMMANDO CODE: 0xDE

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „14“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDE		Führe Satz „14“ aus.
S->M	0x04	0xDE	0x0E 0x00 0xB0	Satz 14 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 14 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.20 EXE PHRASE15 (0xDF)

KOMMANDO CODE: 0xDF

BESCHREIBUNG: Spezielles nur 1 Byte großes Kommando um Satz „15“ abrufen zu können.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Programmzeile (UInt16) mit enthaltenem Kommando Code wird gesendet.

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xDF		Führe Satz „15“ aus.
S->M	0x04	0xDF	0x0F 0x00 0xB0	Satz 15 enthält einen „MOVE POS“ Befehl

SONSTIGES: Ist in der Programmzeile 15 „PRG EXE“ enthalten, so wird automatisch das komplette Unterprogramm ausgeführt. So können mit nur 1 Byte Unterprogramme ausgeführt werden.

2.5.21 PRG GOTO (0xC3)

KOMMANDO CODE: 0xC3

BESCHREIBUNG: Springt an Satznummer

PARAMETER (Master -> Slave): *Satznummer*

ANTWORT (Slave->Master): Keine

BEISPIEL: Macht nur in Zusammenhang mit Programmierung Sinn.

SONSTIGES: Spezieller Befehl, welcher nur in Ablaufprogrammen benötigt wird. Bei einem direkten Aufruf passiert gar nichts. Zur Programmierung der Module sollte spezielle *Programmiersoftware* ([☞ 3.2.7, Seite 150](#)) verwendet werden.

2.5.22 PRG WAIT (0xC4)

KOMMANDO CODE 0xC4

BESCHREIBUNG Wartet angegebene Zeit in [ms]

PARAMETER (Master -> Slave) Zeit [ms]

ANTWORT (Slave->Master) Keine

BEISPIEL Macht nur in Zusammenhang mit Programmierung Sinn.

SONSTIGES Spezieller Befehl, welcher nur in Ablaufprogrammen benötigt wird. Bei einem direkten Aufruf pausiert der Prozessor die angegebene Zeit. Zur Programmierung der Module sollte spezielle Programmiersoftware ([☞ 3.2.7, Seite 150](#)) verwendet werden.

2.6 Sonstige

2.6.1 GET STATE (0x95)

KOMMANDO CODE: 0x95

BESCHREIBUNG: Liefert den Status des Moduls, sowie einige weitere Informationen, wenn gewünscht. Das Modul kann diesen Status selbstständig in regelmäßigen Abständen updaten.

PARAMETER (Master -> Slave):

- Keine
Modul liefert einmalig die Daten. Hiermit kann das vorher eingesetzte zyklische Senden von Daten abgeschaltet werden.
- *Zeit* (4 Byte)
Das Modul sendet selbstständig im eingegebenen Zeitintervall (im jeweiligen *Einheitensystem* ([☞ 1.5, Seite 23](#)) seinen Status.
- *Zeit* (4 Byte) *Modus* (1 Byte)
Das Modul sendet selbstständig im eingegebenen Zeitintervall (im jeweiligen *Einheitensystem* ([☞ 1.5, Seite 23](#)) seinen Status.
Mit dem Code „Modus“ wird parametriert, welche Daten zusätzlich zum Status mitgeliefert werden sollen:
Bit 1 (0x01): Position
Bit 2 (0x02): Geschwindigkeit
Bit 3 (0x04): Strom

ANTWORT (Slave->Master): Optionale Daten (den Code „Modus“ beachten), im Anschluss daran der Status (2 Byte), welcher wie folgt aufgebaut ist:

Referenziert	Bit 1	0x01
Bewegung	Bit 2	0x02
Programmablauf	Bit 3	0x04
Warnung	Bit 4	0x08
Fehler	Bit 5	0x10
Bremse	Bit 6	0x20
Bewegung blockiert	Bit 7	0x40
Position erreicht	Bit 8	0x80

- Bit 1: Das Modul ist referenziert.
- Bit 2: Das Modul bewegt sich.
- Bit 3: Das Modul befindet sich im Programm Modus (Ein internes Ablaufprogramm ist aktiv).
- Bit 4: Es liegt eine *Warnung* ([☞ 2.8.1, Seite 106](#)) vor.
- Bit 5: Es liegt ein *Fehler* ([☞ 2.8.1, Seite 106](#)) vor.
- Bit 6: Die Bremse ist eingefallen.
- Bit 7: Eine *Bewegung wurde unterbrochen* ([☞ 2.3.2, Seite 55](#)).
- Bit 8: Die *Zielposition Position wurde erreicht* ([☞ 2.3.3, Seite 56](#)).
- Bit 9-16 enthalten im Fehlerfall zusätzlich den *Fehlercode* ([☞ 2.8.2, Seite 109](#)).

BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x06	0x95	0x00 0x00 0x80 0x3F 0x07	Status-Information soll zyklisch jede Sekunde vom Modul geliefert werden. Außer Status-Bits werden auch Position, Geschwindigkeit und Strom mitversendet.
S->M (zyklisch jede Sekunde)	0x0F	0x95	0xD6 0xA3 0x70 0x41 0x56 0xC9 0x41 0x40 0x3C 0x41 0xEB 0x3E 0x03 0x00	Position: 0xD6..0x41, Geschwindigkeit: 0x56..0x40, Strom: 0x3C..0x3E; Modul ist in Bewegung und Referenziert (0x03); kein Fehler (0x00)

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x01	0x95		Status-Information einmalig abfragen. Dabei werden zuletzt abgefragte Parameter mitversendet.
S->M	0x0F	0x95	0x0x53 0x63 0xB7 0x41 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x61 0xD9	Position: 0x53..0x41, Geschwindigkeit: 0x00..0x00, Strom: 0x00..0x00; Modul referenziert, Bewegung beendet, Bremse betätigt (0x61); Schnellstop betätigt (0xD9)

BEISPIEL 3:

	D-Len	Cmd	Param	
M->S	0x06	0x95	0x00 0x00 0x00 0x00 0x01	Status-Information einmalig abfragen. Außer Status-Bits wird auch Position mitversendet.
S->M	0x07	0x95	0x00 0x00 0x00 0x00 0x20 0x00	Position: 0x00..0x00; Modul nicht referenziert, Bremse betätigt (0x20); kein Fehler (0x00)

SONSTIGES: Will man Position, Geschwindigkeit und Strom unter CAN in einer Nachricht erhalten, so ist das *Fragmentierprotokoll* ([☞ 1.4.6, Seite 21](#)) zu verwenden. Unter Profibus finden alle Informationen in einer Profibus-Nachricht Platz. Ein einmal gesetzter Datencode „Daten“ wird beibehalten und braucht somit nicht jedes mal neu gesetzt werden. Wird das Modul eingeschaltet, ist der Datencode auf „0x07“ gesetzt, es werden also alle möglichen Status-Informationen komplett übermittelt.

HINWEIS

Unter Profibus werden, wenn alle Parameter (Position, Geschwindigkeit, Strom) übermittelt wurden nur die unteren 8 Bit des Status angezeigt. Diese kommen nun auf Byte 14 zu liegen, wo Profibus spezifisch ([☞ 1.4.5, Seite 19](#)) der Status immer aktuell geliefert wird. Auf Byte 15 folgt der MsgCount, welcher die oberen 8 Bit des Status Wortes überschreibt.

HINWEIS

Unter Profibus wird standardgemäß in Byte 10-13 die aktuelle Position übermittelt. Werden mit GET STATE alle Parameter angefordert, so wird die aktuelle Position in Bytes 10-13 mit dem aktuellen Stromwert überschrieben. Daher ist es empfehlenswert unter Profibus das „Modus“ Byte maximal auf „0x06“ zu setzen (Geschwindigkeit und Strom werden übermittelt).

HINWEIS

Unter Profibus sollte man mit dem „automatischen“ Abrufen vorsichtig sein. Hier kann es unter Umständen günstiger sein die Daten zu „polen“. Insbesondere dann, wenn der FREEZE Mechanismus genutzt wird.

2.6.2 GET STATE AXIS (0x98)

KOMMANDO CODE: 0x98

BESCHREIBUNG: Liefert den Status einer Achse oder eines Roboters des Moduls, sowie einige weitere Informationen, wenn gewünscht. Das Modul kann diesen Status selbständig in regelmäßigen Abständen aktualisieren.

PARAMETER (Master -> Slave):

- Nur *Achsindex* (1 Byte)
Modul liefert einmalig die Daten der Achse mit dem Index „Achsindex“. Hiermit kann das vorher eingestellte zyklische Senden von Daten abgeschaltet werden.
- *Zeit* (4 Byte) + *Achsindex* (1 Byte)
Das Modul sendet selbständig im eingegebenen Zeitintervall (im jeweiligen *Einheitensystem* ([☞ 1.5, Seite 23](#)) den Status der Achse mit dem Index „Achsindex“.
- *Zeit* (4 Byte) + *Modus* (1 Byte) *Achsindex* (1 Byte)
Das Modul sendet selbständig im eingegebenen Zeitintervall (im jeweiligen *Einheitensystem* ([☞ 1.5, Seite 23](#)) den Status der Achse mit dem Index „Achsindex“. Mit dem Code „Modus“ wird parametriert, welche Daten zusätzlich zum Status mitgeliefert werden sollen:
Bit 1 (0x01): Position
Bit 2 (0x02): Geschwindigkeit
Bit 3 (0x04): Strom

ANTWORT (Slave->Master): Optionale Daten (entsprechend dem eingestellten „Modus“), im Anschluss daran der Status (2 Byte) gefolgt vom Achsindex. Die Bits des Status sind gleich aufgebaut wie beim *GET STATE* ([☞ 2.6.1, Seite 84](#)) Kommando.

Beispiele: BEISPIEL 1:

	D-Len	Cmd	Param	
M->S	0x07	0x98	0x00 0x00 0x80 0x3F 0x07 0x01	Status-Information soll zyklisch jede Sekunde von Achse 1 des Moduls geliefert werden. Außer Status-Bits werden auch Position, Geschwindigkeit und Strom mitversendet.
S->M (zyklisch jede Sekunde)	0x10	0x98	0xD6 0xA3 0x70 0x41 0x56 0xC9 0x41 0x40 0x3C 0x41 0xEB 0x3E 0x03 0x00 0x01	Position: 0xD6..0x41, Geschwindigkeit: 0x56..0x40, Strom: 0x3C..0x3E; Modul ist in Bewegung und referenziert (0x03); kein Fehler (0x00); Achsindex: 1 (0x01)

BEISPIEL 2:

	D-Len	Cmd	Param	
M->S	0x02	0x98	0x02	Status-Information von Achse 2 einmalig abfragen. Dabei werden zuletzt abgefragte Parameter mitversendet.
S->M	0x10	0x98	0x0x53 0x63 0xB7 0x41 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x61 0xD9 0x02	Position: 0x53..0x41, Geschwindigkeit: 0x00..0x00, Strom: 0x00..0x00; Modul referenziert, Bewegung beendet, Bremse betätigt (0x61); Schnellstop betätigt (0xD9); Achsindex: 2 (0x02)

BEISPIEL 3:

	D-Len	Cmd	Param	
M->S	0x07	0x98	0x00 0x00 0x00 0x00 0x01 0x03	Status-Information von Achse 3 einmalig abfragen. Außer Status-Bits wird auch die Position mitversendet.
S->M	0x07	0x98	0x00 0x00 0x00 0x00 0x20 0x00 0x03	Position: 0x00..0x00; Modul nicht referenziert, Bremse betätigt (0x20); kein Fehler (0x00); Achsindex: 3 (0x03)

SONSTIGES: Will man Position, Geschwindigkeit und Strom unter CAN in einer Nachricht erhalten, so ist das *Fragmentierprotokoll* ([☞ 1.4.6, Seite 21](#)) zu verwenden. Unter Profibus finden alle Informationen in einer Profibusnachricht Platz. Ein einmal gesetzter „Modus“ wird beibehalten und braucht somit nicht jedes mal neu gesetzt werden. Wird das Modul eingeschaltet, ist der Modus auf „0x07“ gesetzt, es werden also alle möglichen Status-Informationen komplett übermittelt.

HINWEIS

Beim Profibus werden, wenn alle Parameter (Position, Geschwindigkeit, Strom) übermittelt wurden nur die unteren 8 Bit des Status angezeigt. Diese kommen nun auf Byte 14 zu liegen, wo PROFIBUS spezifisch ([☞ 1.4.5, Seite 19](#)) der Status immer aktuell geliefert wird. Auf Byte 15 folgt der MsgCount, welcher die oberen 8 Bit des Status Wortes überschreibt

HINWEIS

Unter Profibus wird standardgemäß in Byte 10-13 die aktuelle Position übermittelt. Werden mit GET STATE alle Parameter angefordert, so wird die aktuelle Position in Bytes 10-13 mit dem aktuellen Stromwert überschrieben. Daher ist es empfehlenswert unter Profibus das „Modus“ Byte maximal auf „0x07“ zu setzen (Position, Geschwindigkeit und Strom werden übermittelt).

HINWEIS

Beim Profibus sollte man mit dem „automatischen“ Abrufen vorsichtig sein. Hier kann es unter Umständen günstiger sein die Daten zu „polen“. Insbesondere dann, wenn der FREEZE Mechanismus genutzt wird.

2.6.3 CMD REBOOT (0xE0)

KOMMANDO CODE: 0xE0

BESCHREIBUNG: Das Modul wird neu gestartet.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Befehl wird mit „OK“ (0x4F4B) bestätigt. Danach startet das Modul neu und meldet sich nach erfolgreichem Neustart mit „INFO BOOT“ ([☞ 2.8.2, Seite 109](#)).
(Ausnahme Serielle Kommunikation ohne Spontanmeldung)

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xE0		
S->M	0x03	0xE0	0x4F 0x4B	Bestätigung mit „OK“
S->M	0x03	0x8A	0x00 0x01	Modul erfolgreich neu gestartet

2.6.4 CMD DIO (0xE1)

KOMMANDO CODE: 0xE1

BESCHREIBUNG: Digitale Ein- / Ausgänge können gesetzt bzw. gelesen werden.

PARAMETER (Master -> Slave):

- **kein**
Es wird der aktuelle Zustand der digitalen Ein- / Ausgänge gelesen.
- **1 Byte**
Mit den oberen 4 Bit können die 4 digitalen Ausgänge gesetzt werden.

ANTWORT (Slave->Master): Beim Erfolg „OK“ (0x4F4B), mit angehängtem Byte für den aktuellen Zustand der digitalen Eingänge in den unteren 4 Bit und dem Zustand der digitalen Ausgänge in den oberen 4 Bit.

Eingang 1	Bit 1	0x01
Eingang 2	Bit 2	0x02
Eingang 3	Bit 3	0x04
Eingang 4	Bit 4	0x08
Ausgang 1	Bit 5	0x10
Ausgang 2	Bit 6	0x20
Ausgang 3	Bit 7	0x40
Ausgang 4	Bit 8	0x80

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xE1		
S->M	0x04	0xE1	0x4F 0x4B 0x00	Keine Ein-/Ausgänge gesetzt

SONSTIGES: Einstellung der digitalen Ein- und Ausgänge beachten!

	ACHTUNG
Beim Einschalten des Moduls haben die Ausgänge kurzfristig undefinierte Zustände. Dies kann unter Umständen zu Zerstörung angeschlossener Hardware führen!	

CMD DIO AXIS(0xEB)**2.6.5**

KOMMANDO CODE: 0xEB

BESCHREIBUNG: Digitale Ein- / Ausgänge einzelner Achsen können gesetzt bzw. gelesen werden.

PARAMETER (Master -> Slave):

- Achsindex
Es wird der aktuelle Zustand der digitalen Ein- / Ausgänge gelesen.
- Achsindex + 1 Byte
Mit den oberen 4 Bit können die 4 digitalen Ausgänge gesetzt werden.

ANTWORT (Slave->Master): Beim Erfolg „OK“ (0x4F4B), mit angehängtem Byte für den aktuellen Zustand der digitalen Eingänge und dem Achsindex. In den unteren 4 Bit und dem Zustand der digitalen Ausgänge in den oberen 4 Bit.

Eingang 1	Bit 1	0x01
Eingang 2	Bit 2	0x02
Eingang 3	Bit 3	0x04
Eingang 4	Bit 4	0x08
Ausgang 1	Bit 5	0x10
Ausgang 2	Bit 6	0x20
Ausgang 3	Bit 7	0x40
Ausgang 4	Bit 8	0x80

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x02	0xE1	0x01	
S->M	0x05	0xE1	0x4F 0x4B 0x00 0x01	Keine Ein-/Ausgänge bei Achse1 gesetzt

SONSTIGES: Einstellung der digitalen Ein- und Ausgänge beachten!

	ACHTUNG
	Beim Einschalten des Moduls haben die Ausgänge kurzfristig undefinierte Zustände. Dies kann unter Umständen zu Zerstörung angeschlossener Hardware führen!

2.6.6 CMD GET DIO (0xE9)

KOMMANDO CODE: 0xE9

BESCHREIBUNG: Digitale Ein-/ Ausgänge können gelesen werden.

PARAMETER (Master -> Slave):

Es sind 3 Typen von Anfragen möglich, welche sich in der Anzahl der übergebenen Parameter und der Art und Anzahl der angefragten digitalen Ein- Ausgänge unterscheiden. Abhängig vom Typ (1-3) der Anfrage werden auch unterschiedliche Antworten vom Modul zurückgeschickt.

Typ 1: Es wird der aktuelle Zustand aller digitalen Eingänge als Bit-Vektor gelesen. (Die Ausgänge können mit Typ 1 Anfragen nicht gelesen werden.):

PARAMETER (Master -> Slave):

- kein

ANTWORT (Slave->Master):

- *Bit Vektor* (4 Byte)

Typ 2: Ein bestimmter digitaler Eingang oder Ausgang wird gelesen. Welcher Eingang oder Ausgang gelesen werden soll wird durch den Parameter „Adresse“ bestimmt. Der „Zustand“ des adressierten digitalen Ein- oder Ausgangs wird in Bit 0 des übertragenen Datenbytes der Antwort übermittelt:

PARAMETER (Master -> Slave):

- *Adresse* (4 Byte)

ANTWORT (Slave->Master):

- *Zustand* (1 Byte)

Typ 3: Eine aufeinanderfolgende Reihe von digitalen Eingängen oder Ausgängen wird gelesen. Welche Eingänge oder Ausgänge gelesen werden sollen wird durch die Parameter „Adresse“ und „Länge“ bestimmt. Beginnend mit dem Ein- oder Ausgang an „Adresse“ werden so viele Ein-/ Ausgänge gelesen wie der Parameter „Länge“ angibt. In der Antwort wird der Zustand der adressierten digitalen Ein- oder Ausgänge als Bit-Vektor in den übertragenen n Datenbytes „Zustände“ übermittelt. Die Anzahl der übermittelten Datenbytes hängt von der angefragten „Länge“ ab und berechnet sich wie folgt: „(Länge + 7) / 8“, wobei „/“ die Integer Division (Division ohne Rest) bezeichnet. Für „Länge“ von 1-8 wird also mit einem Byte geantwortet, für „Länge“ von 9-16 wird mit zwei Bytes geantwortet, usw.:

PARAMETER (Master -> Slave):

- Adresse (4 Byte)
- Länge (1 Byte)

ANTWORT (Slave->Master):

- Zustände (n Bytes)

BEMERKUNG

- Anfragen vom Typ 1 können nur Eingänge lesen (nicht Ausgänge).
- Welche Adressen für Typ 2 bzw. 3 gültig sind hängt vom Modul ab. Siehe hierzu jeweilige Ergänzungsanleitung.

Beispiel:

Typ 1	D-Len	Cmd	Param	
M->S	0x01	0xE9		Typ1: Alle Eingänge lesen
S->M	0x05	0xE9	0x01 0x02 0x04 0x08	Typ 1: Eingang 1, 10, 19 und 28 sind gesetzt, alle anderen Eingänge sind gelöscht

2.6.7 CMD SET DIO (0xEA)

KOMMANDO CODE: 0xEA

BESCHREIBUNG: Digitale Ausgänge können geschrieben werden.
Es sind 3 Typen von Anfragen möglich:

Typ 1: Es wird der Zustand aller digitalen Ausgänge geschrieben:

PARAMETER (Master -> Slave):

- *Bit-Array* (4 Byte)

ANTWORT (Slave->Master) wenn erfolgreich:

- „OK“ 0x4F4B (2 Byte)

Typ 2: Ein bestimmter digitaler Ausgang wird geschrieben. Welcher Ausgang geschrieben werden soll wird durch den Parameter „Adresse“ bestimmt. Der Ausgang wird entsprechend Bit 0 des übertragenen „Zustandes“ gesetzt oder gelöscht:

PARAMETER (Master -> Slave):

- *Adresse* (4 Byte)
- *Zustand* (1 Byte)

ANTWORT (Slave->Master) wenn erfolgreich:

- „OK“ 0x4F4B (2 Byte)

Typ 3: Eine aufeinanderfolgende Reihe von digitalen Ausgängen wird geschrieben. Welche Ausgänge geschrieben werden sollen wird durch die Parameter „Adresse“ und „Länge“ bestimmt. Beginnend mit dem Ausgang an „Adresse“ werden soviele Ausgänge geschrieben wie der Parameter „Länge“ angibt. Die Werte die geschrieben werden sollen sind als Bit-Vektor im Parameter „Zustände“ kodiert. Die Anzahl der hier zu übermittelnden Datenbytes hängt von der „Länge“ ab und berechnet sich wie folgt: „(Länge + 7) / 8“, wobei „/“ die Integer Division (Division ohne Rest) bezeichnet. Für „Länge“ von 1-8 wird also ein Byte, für „Länge“ von 9-16 werden zwei Bytes übermittelt, usw.:

PARAMETER (Master -> Slave):

- *Adresse* (4 Byte)
- *Länge* (1 Byte)
- *Zustände* (n Bytes)

ANTWORT (Slave->Master) wenn erfolgreich:

- „OK“ 0x4F4B (2 Byte)

Unabhängig vom Typ (1-3) der Anfrage wird also immer mit „OK“ (0x4F4B) geantwortet, wenn erfolgreich. Das Modul setzt/ löscht dann die adressierten digitalen Ausgänge.

BEMERKUNG

- Welche Adressen für Typ 2 bzw 3 gültig sind hängt vom Modul ab. Siehe hierzu jeweilige Ergänzungsanleitung.

Beispiel:

Typ 1	D-Len	Cmd	Param	
M->S	0x05	0xEA	0x00 0x00 0x00 0x00	Typ 1: Alle Ausgänge auf 0 setzen
S->M	0x03	0xEA	0x4F 0x4B	„OK“, Ausgänge wurden gesetzt



ACHTUNG

Beim Einschalten des Moduls haben die Ausgänge kurzfristig undefinierte Zustände. Dies kann unter Umständen zu Zerstörung angeschlossener Hardware führen.

2.6.8 FLASH MODE (0xE2)

KOMMANDO CODE: 0xE2

BESCHREIBUNG: Das Modul wird für ein *Firmware-Update* ([☞ 4.4.4, Seite 171](#)) vorbereitet.

PARAMETER (Master -> Slave): Flash-Kennwort

ANTWORT (Slave->Master): Nach erfolgreicher Überprüfung des Kennwertes „OK“ (0x4F4B).

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x??	0xE2	<Kennwort>	
S->M	0x03	0xE2	0x4F 0x4B	

SONSTIGES: War die Kennworteingabe erfolgreich, so dauert es ca. 30 [s] (*) bis sich das Modul selbstständig in den Flash Mode versetzt (LED leuchtet dauerhaft grün und rot).

- * Zeit wird benötigt um einen Kondensator aufzuladen.

HINWEIS

Nach dem Updatevorgang über die Serielle Schnittstelle mit mehreren Modulen, müssen abgeschaltete Module („CMD DISCONNECT“ ([☞ 2.6.9, Seite 98](#))) neu gestartet werden. (Versorgungsspannung trennen.)

2.6.9 CMD DISCONNECT (0xE6)

KOMMANDO CODE: 0xE6

BESCHREIBUNG: Das Modul wird vom Bussystem getrennt und deaktiviert.

PARAMETER (Master -> Slave): Flash-Kennwort

ANTWORT (Slave->Master): Nach erfolgreicher Überprüfung des Kennwertes „OK“ (0x4F4B).

BEISPIEL:

	D-Len	Cmd	Param	
M->S	0x01	0xE6	<Kennwort>	
S->M	0x03	0xE6	0x4F 0x4B	

SONSTIGES: War die Kennworteingabe erfolgreich, so schaltet sich das Modul ab. Die LEDs erlöschen (außer für Motorspannung). Dies ist bei einem Firmware-Update über die Serielle Schnittstelle notwendig, wenn der Bus nicht aufgetrennt werden soll. Es kann immer nur ein einziges Modul aktualisiert werden. Abgeschaltete Module können nur durch Abschalten und erneutes Einschalten der Logikspannung aktiviert werden.

2.6.10 CHANGE USER (0xE3)

KOMMANDO CODE: 0xE3

BESCHREIBUNG: Der aktuelle Benutzer des Moduls wird gewechselt.

PARAMETER (Master -> Slave):

- kein
Nutzer „*Nutzer*“ ([☞ 1.6.1, Seite 25](#)) wird eingestellt.
- *Kennwort*
für den jeweiligen Nutzer. ANTWORT (Slave->Master): Der Befehl wird immer mit „OK“ (0x4F4B) bestätigt. Angehängt (1 Byte) wird der aktuelle Benutzer (0 = User, 1 = Diag, 2 = Profi, 3 = Advanced).

BEISPIEL

	D-Len	Cmd	Param	
M->S	0x01	0xE3	<Kennwort>	
S->M	0x04	0xE3	0x4F 0x4B 0x00	Benutzer „User“ aktiv.

SONSTIGES: Bei der Eingabe eines falschen Kennwertes wird immer „*Nutzer*“ eingestellt. Bei Neustart eines Moduls ist immer „*Nutzer*“ aktiv.

HINWEIS

Beim Nutzer „Diagnose“ ([☞ 1.6.2, Seite 25](#)) wird eine zweite Schnittstelle (Seriell, CAN, Bluetooth) geöffnet, über die keine Bewegungskommandos an das Modul abgesendet werden können. „CMD FAST STOP“ ([☞ 2.2.19, Seite 53](#)) ist aber möglich.

2.6.11 CHECK MC PC COMMUNICATION (0xE4)

KOMMANDO CODE: 0xE4

BESCHREIBUNG: Die Kommunikation vom Modul zur Steuerung kann überprüft werden. Es werden definierte Daten vom Modul zur Steuerung geschickt. Es können sowohl einzelne Daten angefordert werden, sowie alle Daten auf einmal. Bei der Anforderung von Einzeldaten ist eine Fragmentierung **nicht** notwendig.

PARAMETER (Master -> Slave):

- keine
Alle Testdaten werden am Stück gesendet, eine Fragmentierung ist notwendig.
- TEST FLOAT 1 (0x0101)
Der Float Wert „-1.2345“ (0x19 0x04 0x9E 0xBF) wird gesendet, daran angehängt der Parameter Code (0x0101).
- TEST FLOAT 2 (0x0202)
Der Float Wert „47.11“ (0xA4 0x70 0x3C 0x42) wird gesendet, daran angehängt der Parameter Code (0x0202).
- TEST INT32 1 (0x0303)
Der Int32 Wert „0x11223344“ (287454020) wird gesendet, daran angehängt der Parameter Code (0x0303).
- TEST INT32 2 (0x0404)
Der Int32 Wert „0xFFEEDDCC“ (-1122868) wird gesendet, daran angehängt der Parameter Code (0x0404).
- TEST INT16 1 (0x0505)
Der Int16 Wert „0x0200“ (512) wird gesendet, daran angehängt der Parameter Code (0x0505).
- TEST INT16 2 (0x0606)
Der Int16 Wert „0xAF FE“ (-20482) wird gesendet, daran angehängt der Parameter Code (0x0606).

ANTWORT (Slave->Master): Wenn kein Parameter übergeben wurde, müssen folgende Werte in der gegebenen Reihenfolge bei der Steuerung eintreffen:

Daten Typ	Wert HEX	Wert DEC
Float	0x19 0x04 0x9E 0xBF	-1.2345
Float	0xA4 0x70 0x3C 0x42	47.11
Int32	0x11223344	287454020
Int32	0xFFEEDDCC	-1122868
Int16	0x0200	512
Int16	0xAF FE	-20482

Mit entsprechenden Parametern wird nur einer der Werte (je nach Parameter) übermittelt.

BEISPIEL: Siehe *Ausgesuchte Beispiele* ([☞ 6.1, Seite 185](#)).

SONSTIGES: Mit diesem Befehl können eigene Treiber überprüft werden. Für den normalen Betrieb wird dieser Befehl nicht benötigt. Der Datenaustausch einschließlich Fragmentierung vom Modul zur eigenen Steuerung kann mit definierten Werten getestet werden.

HINWEIS

Der Parameter Code wird in der Antwort an das Ende gestellt, um sicherzustellen, dass die Testdaten genau an den Stellen zu liegen kommen, an welchen die realen Daten liegen werden.

2.6.12 CHECK PC MC COMMUNICATION (0xE5)

KOMMANDO CODE: 0xE5

BESCHREIBUNG: Die Kommunikation von der Steuerung zum Modul kann hiermit überprüft werden.

PARAMETER (Master -> Slave):

- Daten können alle auf einmal in der folgenden Reihenfolge übermittelt werden:

Daten Typ	Wert HEX	Wert DEC
Float	0x19 0x04 0x9E 0xBF	-1.2345
Float	0xA4 0x70 0x3C 0x42	47.11
Int32	0x11223344	287454020
Int32	0xFFEEDDCC	-1122868
Int16	0x0200	512
Int16	0xAF FE	-20482

Hierbei muss fragmentiert werden.

- Um einzelne Daten zu testen, sind zunächst die Testdaten zu übermitteln und daran angehängt der Code, welche Testdaten gemeint sind:
 - Sende Float „-1.2345“ (0x19 0x04 0x9E 0xBF), daran angehängt Parameter Code (0x0101)
 - Sende Float „47.11“ (0xA4 0x70 0x3C 0x42), daran angehängt Parameter Code (0x0202)
 - Sende Int32 „0x11223344“ (287454020), daran angehängt Parameter Code (0x0303)
 - Sende Int32 „0xFFEEDDCC“ (-1122868), daran angehängt Parameter Code (0x0404)
 - Sende Int16 „0x0200“ (512), daran angehängt Parameter Code (0x0505)
 - Sende Int16 „0xAF FE“ (-20482), daran angehängt Parameter Code (0x0606)

ANTWORT (Slave->Master): Das Modul antwortet mit „OK“ (0x4F4B), wenn der entsprechende Testwert richtig interpretiert werden konnte. Bei Übermittlung aller Testdaten auf einmal, antwortet das Modul mit „OK (0x4F4B)“ und daran angehängt 1 Byte, in welchem bitweise kodiert ist, welche Daten nicht richtig interpretiert werden konnten (Bit wird auf „1“ gesetzt).

- Bit 1: erster Float Wert (-1.2345) wurde nicht erkannt
- Bit 2: zweiter Float Wert (47.11) wurde nicht erkannt
- Bit 3: erster Int32 Wert (0x11223344) wurde nicht erkannt
- Bit 4: zweiter Int32 Wert (0xFFEEDDCC) wurde nicht erkannt
- Bit 5: erster Int16 Wert (512) wurde nicht erkannt
- Bit 6: zweiter Int16 Wert (0xAF FE) wurde nicht erkannt

BEISPIEL: Siehe *Ausgesuchte Beispiele* ([☞ 6.1, Seite 185](#))

SONSTIGES: Mit diesem Befehl können eigene Treiber überprüft werden. Für den normalen Betrieb wird dieser Befehl nicht benötigt. Es ist möglich den Datenaustausch von der Steuerung zum Modul an einzelnen definierten Werten zu testen, oder an einem definierten Datenpaket, bei dem eine Fragmentierung notwendig ist.

HINWEIS

Der Parameter Code wird in der Antwort ans Ende gestellt, um sicherzustellen, dass die Testdaten genau an den Stellen zu liegen kommen, an welchen die realen Daten liegen werden.

2.7 Fragmentierung

2.7.1 FRAG ACK (0x87)

KOMMANDO CODE: 0x87

BESCHREIBUNG: Bestätigung für ein ordnungsgemäß verarbeitetes Fragment.

PARAMETER (Master -> Slave): *D-Len Code des erhaltenen Fragments* (UInt16), wenn die Steuerung dem Modul das Fragment bestätigt.

ANTWORT (Slave->Master): *D-Len Code des erhaltenen Fragments* (UInt16), wenn das Modul der Steuerung das Fragment bestätigt.

BEISPIEL: Siehe *Ausgesuchte Beispiele* ([☞ 6.1, Seite 185](#)).

HINWEIS

Dieser Befehl wird nur beim Profibus ([☞ 1.4.6, Seite 21](#)) benötigt, wenn eine Fragmentierung von Nachrichten notwendig wird.

2.7.2 FRAG START (0x84)

KOMMANDO CODE 0x84

BESCHREIBUNG: Zeigt bei einer fragmentierten Nachricht, dass es sich um das erste Fragment handelt.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Keine

BEISPIEL: Siehe *Ausgesuchte Beispiele* ([☞ 6.1, Seite 185](#)).

SONSTIGES: Wird direkt hinter dem D-Len Byte geschrieben. Zählt allerdings nicht zu D-Len, da es nur als Marker dient.

2.7.3 FRAG MIDDLE (0x85)

KOMMANDO CODE: 0x85

BESCHREIBUNG: Zeigt bei einer fragmentierten Nachricht, dass es sich um ein mittleres Fragment handelt.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Keine

BEISPIEL: Siehe *Ausgesuchte Beispiele* ([☞ 6.1, Seite 185](#)).

SONSTIGES: Wird direkt hinter dem D-Len Byte geschrieben. Zählt allerdings nicht zu D-Len, da es nur als Marker dient.

2.7.4 FRAG END (0x86)

KOMMANDO CODE 0x86

BESCHREIBUNG: Zeigt bei einer fragmentierten Nachricht, dass es sich um das letzte Fragment handelt.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Keine

BEISPIEL: Siehe *Ausgesuchte Beispiele*. ([☞ 6.1, Seite 185](#)).

Sonstiges: Wird direkt hinter dem D-Len Byte geschrieben. Zählt allerdings nicht zu D-Len, da es nur als Marker dient.

2.8 Fehlermeldungen

Im Fehlerfall hat D-Len im Datenrahmen vom Modul zur Steuerung immer genau den Wert „2“ bzw. „3“, wenn der optionale Achs-Index verwendet wird. Im Kommando Byte steht entweder das Kommando welches fehlgeschlagen ist, oder eines der folgenden „Fehler Kommandos“. Das Parameter-Byte enthält die Information über die *Fehlerursache* ([☞ 2.8.2, Seite 109](#)).



Abb. 6 Fehlermeldung

2.8.1 Fehlerkommandos

2.8.1.1 CMD ERROR (0x88)

KOMMANDO CODE: 0x88

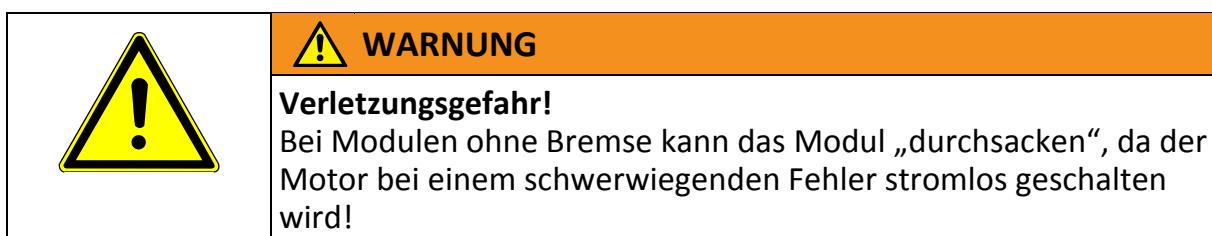
BEISPIEL:

	D-Len	Cmd	Param	
S->M	0x02	0x88	0xDA	Ein Schleppfehler ist aufgetreten „ERROR TOW“ (☞ 2.8.2, Seite 109)).

BESCHREIBUNG:

Ein schwerwiegender Fehler ist aufgetreten, welcher einen Benutzereingriff notwendig macht. Diese Fehler müssen mit „CMD ACK“ quittiert werden. Das Modul ist nicht betriebsbereit. Der Motor ist stromlos geschaltet. Diese Fehlermeldung wird alle 15 Sekunden regelmäßig vom Modul zur Steuerung gesendet. Alle *Fehlercodes* (☞ 6.4, Seite 200) als Übersicht.

- Serielle Kommunikation: das erste Byte einer Nachricht ändert sich zu „0x03“
- CAN: die ersten drei Bit des Identifiers ergeben sich zu „0x3“
- Profibus: eine erweiterte Diagnose wird generiert



2.8.1.2 CMD WARNING (0x89)

KOMMANDO CODE: 0x89

BEISPIEL:

	D-Len	Cmd	Param	
S->M	0x02	0x89	0xD6	Modul steht noch im oberen Softwareendanschlag („ER-ROR SOFT HIGH“) (☞ 2.8.2, Seite 109)

BESCHREIBUNG:

- Softwareendanschläge wurden verletzt. Hier wird zunächst ein Schnellstop ausgelöst und muss quittiert werden. Anschließend ist das Modul bedingt betriebsbereit. (Fahrten aus dem Softwareendanschlag heraus sind gestattet). Ist das Modul aus dem Softwareendanschlag herausgefahren, wird die Warnung gelöscht.
- *Temperaturobergrenze* (☞ [3.2.5, Seite 136](#)) wurde überschritten. Das Modul wird sich in 1 [min] abschalten, wenn die Temperatur nicht gesenkt wird.

Diese Fehlermeldungen werden regelmäßig vom Modul zur Steuerung gesendet (alle 30 [s]). Alle *Fehlercodes* (☞ [6.4, Seite 200](#)) als Übersicht.

- Serielle Kommunikation: das erste Byte einer Nachricht ändert sich zu „0x03“
- CAN: die ersten drei Bit des Identifiers ergeben sich zu „0x03“
- Profibus: eine erweiterte Diagnose wird generiert

2.8.1.3 CMD INFO (0x8A)

Siehe „CMD INFO“ (☞ [2.3.1, Seite 54](#))

2.8.1.4 CMD ACK (0x8B)

KOMMANDO CODE: 0x8B

BESCHREIBUNG: Quittierung einer anstehenden Fehlermeldung.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): „OK“ (0x4F4B)

BEISPIEL:

	D-Len	Cmd	Param
M->S	0x01	0x8B	
S->M	0x03	0x8B	0x4F 0x4B

SONSTIGES: Wenn alle Fehler erfolgreich quittiert werden konnten, wird nach dem Senden von „OK“ (0x4F4B) eine Info Nachricht „INFO NO ERROR“ ([2.8.2, Seite 109](#)) versandt.

2.8.1.5 GET DETAILED ERROR INFO (0x96)

KOMMANDO CODE: 0x96

BESCHREIBUNG: Erweiterte Information zum aktuellen Fehler. Das Kommando muss unmittelbar nach dem aufgetretenen Fehler gesendet werden. Der angezeigte Wert dient dem Schunk-Service zur genaueren Fehlerdiagnose.

PARAMETER (Master -> Slave): Keine

ANTWORT (Slave->Master): Kommando (1 Byte), Fehlercode (1 Byte), Daten (Float)

2.8.1.6 Beispiele

BEISPIEL:

	D-Len	Cmd	Param
M->S	0x01	0x96	
S->M	0x07	0x96	0x88 0xD9 0x00 0x00 0x00 0x00

SONSTIGES: Wenn kein Fehler oder keine erweiterte Information dazu vorhanden ist, wird der Befehl mit „INFO FAILED“ ([2.8.2, Seite 109](#)) beantwortet.

2.8.2 Info- und Fehlercodes

2.8.2.1 ERROR CABLE BREAK (0x76)

CODE: 0x76

BESCHREIBUNG: Die Kommunikation zur Steuerung war unterbrochen.

BEMERKUNG: Bei einem defekten Kommunikationskabel wird dieser Fehler erzeugt. Er wird erst angezeigt, wenn die Kommunikation wieder hergestellt ist.

2.8.2.2 ERROR COMMUTATION (0xDD)

CODE: 0xDD

BESCHREIBUNG: Motor kann nicht kommutieren.

BEMERKUNG: Tritt dieser Fehler gehäuft auf, so ist entweder die Kommutierart falsch gewählt, oder bei Block-Kommutierung sind die Hallgeber defekt bzw. nicht angeschlossen. Bei Sinus-Kommutierung liegt ein Fehler im Positionsmesssystem vor.

2.8.2.3 ERROR CONFIG MEMORY (0xD2)

CODE: 0xD2

BESCHREIBUNG: Der *Konfigurationsbereich* ([☞ 3.2, Seite 118](#)) ist fehlerhaft.

BEMERKUNG: Das Schreiben des EEPROMs ist fehlgeschlagen oder das EEPROM ist defekt.

2.8.2.4 ERROR CURRENT (0xDE)

CODE: 0xDE

BESCHREIBUNG: Der *maximale Strom* ([☞ 3.2.1, Seite 119](#)) wurde überschritten.

BEMERKUNG: Belastung des Motors reduzieren, evtl. Zwischenschritte einfügen.

2.8.2.5 ERROR FAST STOP (0xD9)

CODE: 0xD9

BESCHREIBUNG: Es wurde ein Schnellstop ausgelöst.

2.8.2.6 ERROR FRAGMENTATION (0xDC)

CODE: 0xDC

BESCHREIBUNG: Im *Fragmentierungsprotokoll* ([☞ 1.4.6, Seite 21](#)) ist ein Fehler aufgetreten.

BEMERKUNG: Es sind Datenpakete verloren gegangen.

2.8.2.7 ERROR I2T (0xDF)

CODE: 0xDF

BESCHREIBUNG : Ein I²T Fehler ist aufgetreten.

BEMERKUNG: Belastung des Moduls reduzieren.

2.8.2.8 ERROR INITIALIZE (0xE0)

CODE: 0xE0

BESCHREIBUNG: Das Modul konnte nicht richtig initialisiert werden.

BEMERKUNG: *Konfigurationsparameter* ([☞ 3.2, Seite 118](#)) prüfen.

2.8.2.9 ERROR INTERNAL (0xE1)

CODE: 0xE1

BESCHREIBUNG: Ein interner Fehler ist aufgetreten.

BEMERKUNG : Die Firmware befindet sich in einem undefinierten Zustand. Bitte setzen Sie sich mit unserem Service in Verbindung.

2.8.2.10 ERROR INVALID PHRASE (0xD4)

CODE: 0xD4

BESCHREIBUNG: Der programmierte Satz, welcher ausgeführt werden soll, ist fehlerhaft.

BEMERKUNG: Programmierung prüfen. Grenzen der Parameter prüfen.

2.8.2.11 ERROR LOGIC HIGH (0x73)

CODE: 0x73

BESCHREIBUNG: Die Logikspannung ist oberhalb des Grenzwertes.

BEMERKUNG: Logikspannung prüfen.

2.8.2.12 ERROR LOGIC LOW (0x72)

CODE: 0x72

BESCHREIBUNG: Die Logikspannung ist unterhalb des Grenzwertes.

BEMERKUNG: Logikspannung prüfen.

2.8.2.13 ERROR MATH (0xEC)

CODE: 0xEC

BESCHREIBUNG: Es ist ein „mathematischer“ Fehler aufgetreten, z.B. Division durch Null.

BEMERKUNG: Meist ist ein Konfigurationsparameter falsch, und es kommt zu einer Wertebereichsüberschreitung. In den meisten Fällen ist ein *Reglerparameter* ([3.2.4, Seite 133](#)) falsch eingestellt.

Mit der *detaillierten Fehlerinformation* ([2.8.1, Seite 106](#)) kann der Fehler durch den Schunk-Service genau lokalisiert werden.

2.8.2.14 ERROR MOTOR VOLTAGE HIGH (0x75)

CODE: 0x75

BESCHREIBUNG: Die Motorspannung ist oberhalb des Grenzwertes.

BEMERKUNG: Motorspannung prüfen.

2.8.2.15 ERROR MOTOR VOLTAGE LOW (0x74)

CODE: 0x74

BESCHREIBUNG: Die Motorspannung ist unterhalb des Grenzwertes.

BEMERKUNG: Motorspannung prüfen. Tritt dieser Fehler häufiger auf, kann das Netzteil der Motorspannung unterdimensioniert sein, oder die Kabel von der Spannungsversorgung zum Modul sind falsch dimensioniert.

HINWEIS

Dieser Fehler kann direkt nach dem Einschalten des Moduls auftreten, wenn das Modul zunächst mit Logik Spannung verbunden wurde und anschließend die Motorspannung zugeschalten wird.

2.8.2.16 ERROR OVERSHOOT (0x82)

CODE: 0x82

BESCHREIBUNG: Das Modul ist über die Zielposition um „max. Überschwinger“ ([☞ 3.2.4, Seite 133](#)) hinausgefahren.

BEMERKUNG:

- Der zu Abbremsen benötigte Strom ist zu gering. Stromvorgabe erhöhen.
- Reglerparameter überprüfen.

2.8.2.17 ERROR PROGRAM MEMORY (0xD3)

CODE: 0xD3

BESCHREIBUNG: Der Programmspeicher ist fehlerhaft.

BEMERKUNG: Der komplette Programmspeicher muss gelöscht werden.

2.8.2.18 ERROR RESOLVER CHECK FAILED (0xEB)

CODE: 0xEB

BESCHREIBUNG: Ein Parameter der *Resolvereinstellung* ([☞ 3.2.6, Seite 146](#)) ist fehlerhaft.

BEMERKUNG: -

2.8.2.19 ERROR SERVICE (0xD8)

CODE: 0xD8

BESCHREIBUNG: Bitte mit dem Kundendienst in Verbindung setzen..

2.8.2.20 ERROR SOFT HIGH (0xD6)

CODE: 0xD6

BESCHREIBUNG: Das Modul hat den oberen Softwareendanschlag überfahren.

BEMERKUNG: Falls „CMD ERROR“ ([☞ 2.8.1, Seite 106](#)) anliegt, ist dieser zu quittieren. => Aus dem Fehler wird ein „CMD WARNING“ ([☞ 2.8.1, Seite 106](#)). Erst jetzt ist es möglich das Modul aus dem Softwareendanschlag mit einem beliebigen *Verfahrbefehl* ([☞ 2.2, Seite 32](#)) heraus zubewegen.

2.8.2.21 ERROR SOFT LOW (0xD5)

CODE: 0xD5

BESCHREIBUNG: Das Modul hat den unteren Softwareendanschlag überfahren.

BEMERKUNG: Falls das Kommando-Byte „CMD ERROR“ ([☞ 2.8.1, Seite 106](#)) anliegt, ist dieses zu quittieren. => Aus dem Fehler wird ein „CMD WARNING“ ([☞ 2.8.1, Seite 106](#)). Erst jetzt ist es möglich das Modul aus dem Softwareendanschlag mit einem beliebigen *Verfahrbefehl* ([☞ 2.2, Seite 32](#)) heraus zubewegen.

2.8.2.22 ERROR TEMP HIGH (0x71)

CODE: 0x71

BESCHREIBUNG: Der zulässige *Temperaturbereich wurde überschritten* ([☞ 3.2.5, Seite 136](#)).

BEMERKUNG: Modul abkühlen lassen. Belastung reduzieren.

2.8.2.23 ERROR TEMP LOW (0x70)

CODE: 0x70

BESCHREIBUNG: Der zulässige *Temperaturbereich wurde unterschritten* ([☞ 3.2.5, Seite 136](#)).

BEMERKUNG: Modul aufwärmen.

2.8.2.24 ERROR TOO FAST (0xE4)

CODE: 0xE4

BESCHREIBUNG: Die maximale *Geschwindigkeit* ([☞ 3.2.1, Seite 119](#)) wurde überschritten. (Motor geht durch)

2.8.2.25 ERROR TOW (0xDA)

CODE: 0xDA

BESCHREIBUNG: Ein *Schleppfehler* ([☞ 3.2.5, Seite 136](#)) ist aufgetreten.

BEMERKUNG: Belastung des Moduls reduzieren.

2.8.2.26 ERROR VPC3 (0xDB)

CODE: 0xDB

BESCHREIBUNG: Der Profibus Controller arbeitet fehlerhaft.

BEMERKUNG: Kann an Profibus nur mit dem Nutzer „*Diagnose*“ ([☞ 1.6.2, Seite 25](#)) ermittelt werden, da diese Meldung nicht über Profibus gesendet werden kann (der Profibus-Controller ist ja defekt!).

2.8.2.27 ERROR WRONG RAMP TYPE (0xC8)

CODE: 0xC8

BESCHREIBUNG: Für die Positionsfahrt ist kein gültiges *Verfahrprofil* ([☞ 3.2.5, Seite 136](#)) ausgewählt.

2.8.2.28 INFO BOOT (0x0001)

CODE: 0x0001

BESCHREIBUNG: Das Modul hat erfolgreich gebootet.

(*Info Nachricht* ([☞ 2.3.1, Seite 54](#)))

BEMERKUNG: Wird nach dem Einschalten und erfolgreicher Initialisierung des Moduls ausgelöst, nach einem *Neustart* ([☞ 2.6.3, Seite 90](#)). Tritt diese Meldung während des Betriebes gehäuft auf, sollte die Logikspannung überprüft werden, bzw. besteht die Möglichkeit eines defekten Leistungstreibers.

HINWEIS

Der Code besteht aus 2 Byte => D-Len > „2“ => wird nur das D-Len Byte abgefragt wird diese Meldung nicht als Fehler interpretiert. Die Regel D-Len == „2“ => Fehler kann somit immer eingehalten werden.

HINWEIS

Sollte die Schnittstelle Serielle Kommunikation ohne Spontanmeldung ([☞ 3.2.5, Seite 136](#)) ausgewählt sein, wird diese Meldung unterdrückt.

2.8.2.29 INFO CHECKSUM (0x19)

CODE: 0x19

BESCHREIBUNG: Die Checksumme ist fehlerhaft => Daten sind ungültig.

2.8.2.30 INFO COMMUNICATION ERROR (0x09)

CODE: 0x09

BESCHREIBUNG: Ein schwerwiegender Fehler in der Seriellen Kommunikation ist aufgetreten.

2.8.2.31 INFO FAILED (0x05)

CODE: 0x05

BESCHREIBUNG: Das Kommando ist fehlgeschlagen.

BEMERKUNG: Die Parameter sind alle richtig, aber aus anderen Gründen ist die Ausführung des Kommandos derzeit nicht möglich. Z.B. befindet sich das Modul im „Schnellstop“. Bei „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)), „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)), tritt diese Meldung auf, wenn die Parameter prinzipiell in Ordnung sind, mit den Werten die Position in der geforderten Zeit jedoch nicht erreicht werden kann. Die Fehlermeldung erscheint auch beim Ausführen von einem Schleifenfahrtbefehl „MOVE POS LOOP“ ([☞ 2.2.7, Seite 43](#)), wenn das Modul sich im Softwareendanschlag ([☞ 2.8.1, Seite 106](#)) befindet.

2.8.2.32 INFO MESSAGE LENGTH (0x1D)

CODE: 0x1D

BESCHREIBUNG: D-Len passt nicht zu den erhaltenen Daten.

2.8.2.33 INFO NO ERROR (0x0008)

CODE: 0x0008

BESCHREIBUNG: Es liegen keine weiteren Fehlermeldungen an.

BEMERKUNG: Wird direkt nach einem „CMD ACK“ generiert, wenn keine weiteren Fehler vorhanden sind, bzw. wenn das Modul aus den Softwareendanschlägen herausgefahren ist.

HINWEIS

Der Code besteht aus 2 Byte => D-Len > „2“ => wird nur das D-Len Byte abgefragt, wird diese Meldung nicht als Fehler interpretiert. Die Regel D-Len == „2“ => Fehler kann somit immer eingehalten werden.

2.8.2.34 INFO NO RIGHTS (0x03)

CODE: 0x03

BESCHREIBUNG: Sie haben nicht die passenden Rechte das Kommando auszuführen.

2.8.2.35 INFO SEARCH SINE VECTOR (0x0007)

CODE: 0x0007

BESCHREIBUNG: Der Raumzeiger für die Sinus-Kommutierung wird gesucht. Hierbei werden die Phasen mit 60% des Maximalstromes bestromt.

BEMERKUNG: Wird nach dem Einschalten einmalig vor der Durchführung eines Bewegungsbefehles ausgeführt, wenn *Stillstandkommunikation* ([☞ 1.8, Seite 29](#)) nicht aktiv ist, und *Kommutierart* ([☞ 3.2.1, Seite 119](#)) auf PMSM steht.

HINWEIS

Der Code besteht aus 2 Byte => D-Len > „2“ => wird nur das D-Len Byte abgefragt, wird diese Meldung nicht als Fehler interpretiert. Die Regel D-Len == „2“ => Fehler kann somit immer eingehalten werden.

2.8.2.36 INFO TIMEOUT (0x10)

CODE: 0x10

BESCHREIBUNG: Es ist in der Kommunikation eine Zeitüberschreitung aufgetreten.

BEMERKUNG: Daten konnten nicht verschickt werden, bzw. es wurden noch weiter Daten erwartet, welche nicht rechtzeitig eingetroffen sind.

2.8.2.37 INFO UNKNOWN AXIS INDEX (0x11)

CODE: 0x11

BESCHREIBUNG: Das Modul hat den falschen Achsenindex empfangen.

2.8.2.38 INFO UNKNOWN COMMAND (0x04)

CODE: 0x04

BESCHREIBUNG: Das gesendete Kommando ist unbekannt.

2.8.2.39 INFO WRONG BAUDRATE (0x16)

CODE: 0x16

BESCHREIBUNG: Falsche Baudrate bei der Kommunikation.

2.8.2.40 INFO WRONG PARAMETER (0x1E)

CODE: 0x1E

BESCHREIBUNG: Einer der angegebenen Parameter liegt außerhalb des zulässigen Bereiches.

BEMERKUNG: Wird ein Parameter als falsch erkannt, werden alle alten Parameter beibehalten, auch wenn die übrigen Parameter im gültigen Bereich liegen sollten.

2.8.2.41 ERROR REFERENCED (0x06) oder NOT REFERENCED (0x06)

CODE: 0x06

BESCHREIBUNG: Modul ist nicht Referenziert und kann daher das Kommando nicht ausführen.

BEMERKUNG: Um eine Positionsfahrt auszuführen ist eine Referenzierung notwendig.

HINWEIS

Erweiterte Informationen zum aktuellen Fehler über „GET DETAILED ERROR INFO“ ([☞ 2.8.1.5, Seite 108](#)) Bei Meldung „REF SWITCH MOVED“ prüfen ob der Indexpuls gegenüber dem Referenzpunkt verschoben worden ist. Lösung des Problems über „Abstand zum Index prüfen“ ([☞ 3.2.3.9, Seite 132](#))

2.8.2.42 ERROR HARDWARE VERSION (0x83)

Die Hardwareversion der Steuerungsplatine konnte nicht ermittelt werden.

3 Konfigurationsparameter

3.1 Allgemein

Alle Konfigurationsparameter werden im internen EEPROM dauerhaft gesichert.

3.2 EEPROM

Der Zugriff auf die jeweiligen Elemente ist in der Firmware der Module geregelt. Es gibt hierbei verschiedene Nutzer, welche jeweils über Passwörter geschützt sind. Eine Änderung des Nutzers ist über den Befehl „*CHANGE USER*“ ([☞ 2.6.10, Seite 99](#)) möglich. Folgende Benutzer sind dem Modul bekannt:

- Nutzer:
Standard Nutzer. Ist beim Einschalten des Moduls aktiv. Kann das Modul sehr eingeschränkt parametrieren, aber vollständig bedienen.
- Diag:
Es wird eine zweite Schnittstelle (Seriell, Bluetooth, CAN) für Diagnosezwecke aktiviert. Hierüber dürfen keine Bewegungskommandos gesendet werden!
- Profi:
Kann alle wichtigen Parameter ändern. Bei einer falschen Parametrierung kann es zu unvorhergesehenem Verhalten kommen. Das Modul kann aber nicht zerstört werden.
- Advanced:
Kann alle wichtigen Parameter ändern. Eine falsche Parametrierung kann zur Zerstörung des Moduls führen!
- Root:
Zugriff auf alle Parameter. Eine falsche Parametrierung kann zur Zerstörung des Moduls führen!

3.2.1 Motor

3.2.1.1 Serien Nummer

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Seriennummer des eingebauten Motors.

DATEN: (UInt32) => 0 .. 4294967296

3.2.1.2 Spannung

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Nennspannung des Motors. Hiervon wird die Ansteuerung der Bremse abgeleitete (Tastverhältnis), sowie die maximal zulässige PWM für Testzwecke berechnet. Für automatische Reglerkonfiguration wird dieser Wert ebenfalls benötigt.

DATEN: (UInt16) => 0 .. 65535

	<p>ACHTUNG</p> <p>Eine falsche Eingabe kann zur Zerstörung der Elektronik und des Motors führen.</p>
---	---

3.2.1.3 Typ

ZUGRIFFSRECHTE: Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Der Motortyp kann ausgewählt werden.

- DC (0x00): Bürstenbehafteter Gleichstrommotor
- BLDC (0x01): Elektronisch kommutierter bürstenloser Gleichstrommotor mit Block-Kommutierung.
- PMSM (0x02): Elektronisch kommutierter bürstenloser Gleichstrommotor mit Sinus-Kommutierung.

DATEN: (UInt16) => 0 .. 65535

	<p>ACHTUNG</p> <p>Eine falsche Eingabe kann zur Zerstörung der Elektronik und des Motors führen.</p>
---	---

3.2.1.4 I²T

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Die I²T Überwachung kann aktiviert werden. Bei zu hoher Belastung wird ein I²T Fehler ([☞ 2.8.2, Seite 109](#)) ausgelöst.

Bei der I²T Überwachung wird davon ausgegangen, dass der maximal Strom für 3 Sekunden (entspricht 100%) anliegen darf. Wird ein Wert < 100% eingetragen verkürzt sich die Zeit dementsprechend (Die I²T Überwachung löst früher aus). Werte grösser 100% sowie 0% führen zu einem *ERROR INITIALIZE* ([☞ 2.8.2, Seite 109](#)).

DATEN : (UInt8) => 1 .. 100%

3.2.1.5 Polpaare

ZUGRIFFSRECHTE: *Root* ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Elektrische Polpaare des Motors. Wird nur bei bürstenlosen DC Motoren benötigt. Hat Auswirkungen auf Berechnung von Geschwindigkeiten, Positionen und Kommutierungsmuster.

DATEN: (UInt16) => 0 .. 65535

3.2.1.6 Anschlußwiderstand

ZUGRIFFSRECHTE:

Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Anschlusswiderstand wird für Testfunktionen zur Begrenzung von maximalen Strömen benötigt, sowie für die automatische Reglerkonfiguration.

DATEN: (Float) [Ohm]

	ACHTUNG
	Eine falsche Eingabe kann zur Zerstörung der Elektronik und des Motors führen.

3.2.1.7 Induktivität

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Anschlussinduktivität wird für automatische Reglerkonfiguration benötigt.

DATEN: (Float) [mH]

	<p>ACHTUNG</p> <p>Eine falsche Eingabe kann zur Zerstörung der Elektronik führen.</p>
---	--

3.2.1.8 Max. Strom

ZUGRIFFSRECHTE: Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Maximal zulässiger Strom, welcher kurzzeitig durch die Phasen des Motors fließen darf. Wird dieser Strom längere Zeit (mehrere ms) überschritten wird ein Schnellstop ausgelöst mit Fehlermeldung *ERROR CURRENT* ([☞ 2.8.2, Seite 109](#)).

DATEN: (4 Byte) 0.00 .. 29.99[A] bzw. 0 .. 29999[mA] je nach *Einheitensystem* ([☞ 1.5, Seite 23](#))

	<p>ACHTUNG</p> <p>Eine falsche Eingabe kann zur Zerstörung der Elektronik führen.</p>
---	--

3.2.1.9 Nom. Strom

ZUGRIFFSRECHTE: Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Strom der dauerhaft durch die Phasen des Motors fließen darf. Wird dieser längere Zeit überschritten, wird ein $n/^\wedge(2)T$ Fehler ([☞ 2.8.2, Seite 109](#)) ausgelöst.

Typ: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

	<p>ACHTUNG</p> <p>Eine falsche Eingabe kann zur Zerstörung der Elektronik des Motors führen.</p>
---	---

3.2.1.10 Max. Geschwindigkeit

ZUGRIFFSRECHTE:

Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Maximal zulässige Geschwindigkeit des Systems (abtriebsseitig).

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.1.11 Max. Beschleunigung

ZUGRIFFSRECHTE:

Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Maximale zulässige Beschleunigung des Systems (abtriebsseitig).

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.1.12 Max. Ruck

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Maximal zulässiger Ruck des Systems (abtriebsseitig). Der Ruck ist die zeitliche Änderung der Beschleunigung. Dieser Parameter wird nur ausgewertet, wenn eine *Positionsfahrt mit Ruckbegrenzung* ([☞ 3.2.5, Seite 136](#)) ausgeführt wird.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.1.13 Kommutiertabelle

ZUGRIFFSRECHTE:

Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Für die Block-Kommutierung mit Hilfe von Hallgeber wird hier die für die Einheit gültige Hallgeber-Tabelle eingesetzt. Bei fehlerhaftem Eintrag bewegt sich der Motor gar nicht oder erzeugt sehr wenig Moment.

DATEN: (UInt16) 0 .. 12

3.2.1.14 Offset Phase A

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Nullpunktabgleich des ersten Stromsensors. Dieser Wert sollte sich in einem Bereich von 1700 - 2200 bewegen. Ansonsten besteht der Verdacht auf eine defekte Hardware.

DATEN: (UInt16) 0 .. 65535

	ACHTUNG
<p>Ein falscher Wert kann zu unvorhersehbaren Verhalten des Antriebs führen (Fährt nur in eine Richtung, ruckelt stark, geht durch).</p>	

3.2.1.15 Offset Phase B

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Nullpunktabgleich des zweiten Stromsensors. Dieser Wert sollte sich in einem Bereich von 1700 - 2200 bewegen. Ansonsten besteht der Verdacht auf eine defekte Hardware.

DATEN: (UInt16) 0 .. 65535

	ACHTUNG
<p>Ein falscher Wert kann zu unvorhersehbaren Verhalten des Antriebs führen (Fährt nur in eine Richtung, ruckelt stark, geht durch).</p>	

3.2.1.16 Strommessbereich

ZUGRIFFSRECHTE:

Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Maximaler Strommessbereich des intern eingesetzten Stromsensors.

DATEN: (Float)

	ACHTUNG
<p>Eine falsche Eingabe kann zur Zerstörung der Elektronik führen.</p>	

3.2.2 Getriebe

3.2.2.1 Serien Nummer

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Seriennummer des Getriebes.

DATEN: (UInt32)

3.2.2.2 Übersetzung 1

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Getriebeübersetzung bzw. -untersetzung vom Motor zum Abtrieb.

DATEN: (Float)

3.2.2.3 Übersetzung 2

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Getriebeübersetzung bzw. -untersetzung vom Positionsmesssystem zum Abtrieb. Wird nur benötigt, wenn das *Positionsmesssystem zwischen den Getriebeübersetzungen* ([☞ 3.2.6, Seite 146](#)) eingebaut ist.

DATEN: (Float)

3.2.3 Referenzierung

3.2.3.1 Typ

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Die Art der Referenzierung kann festgelegt werden. (Abschnitt *Pos Type* ([☞ 3.2.6, Seite 146](#)) bei Einsatz von Encoder mit Indexspur beachten).

- Interner Schalter links (0x00) / rechts (0x01)
Der Interne Referenzschalter wird zur Referenzierung herangezogen. Die Bewegungsrichtung bei aktivem Referenzschalter wird über Richtung „links“ bzw. „rechts“ bestimmt.
- Externer Schalter IN0 links (0x02) / rechts (0x03)
Ein externer Referenzschalter (IN0) (*digitale Eingänge* ([☞ 3.2.5, Seite 136](#))) wird zur Referenzierung herangezogen. Die Bewegungsrichtung bei aktivem Referenzschalter wird über Richtung „links“ bzw. „rechts“ bestimmt.

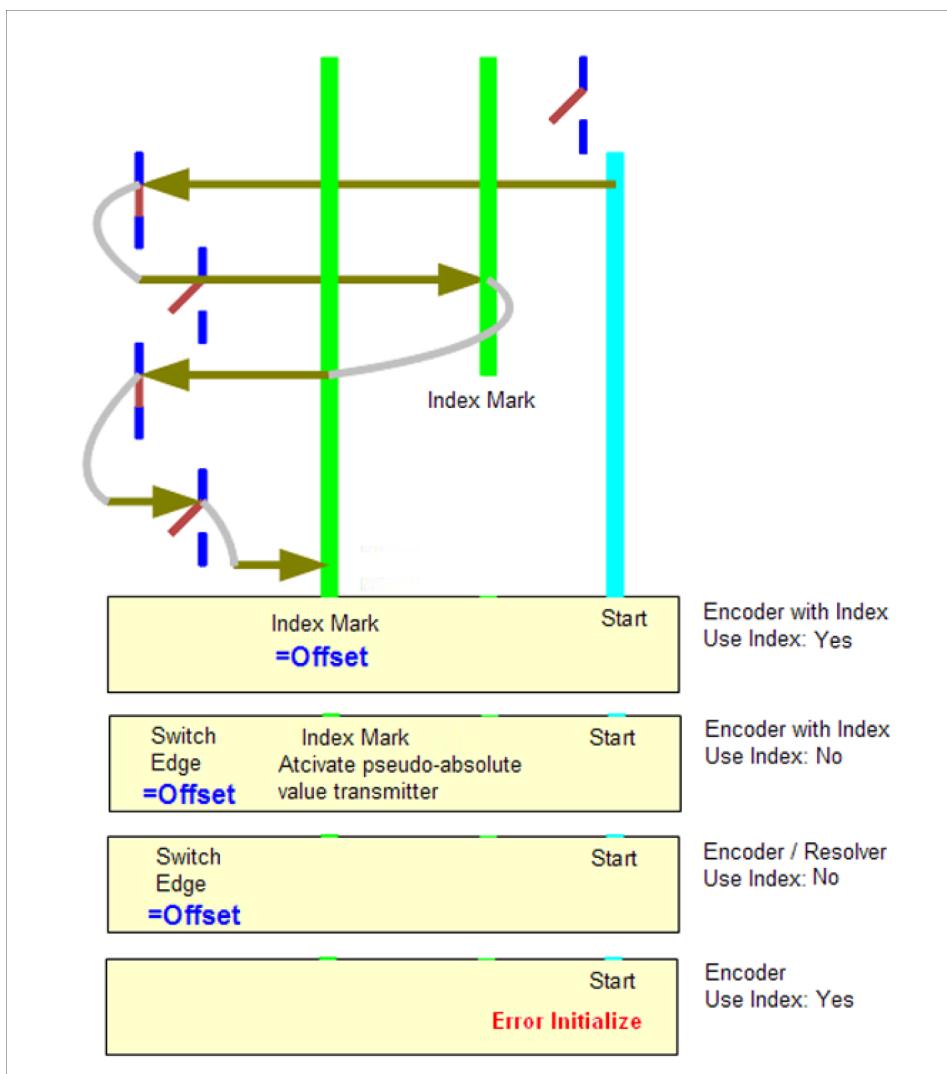


Abb. 7 Referenzierung mit Schalter

HINWEIS

Es ist sicherzustellen, dass die Schaltflanke des Näherungsschalters für mind. 200ms anliegt. (Referenziergeschwindigkeit anpassen, Schaltnocken anpassen).

Funktionsweise:

Ist der Parameter „Abstand zum Index“ zu „0“ gesetzt wird während der Referenzfahrt der Abstand vom Referenzereignis (Schalterflanke dedektiert) zum Indexpuls gemessen und abgespeichert. Bei nachfolgenden Referenzfahrten wird dieser Abstand erneut ermittelt und mit dem gespeicherten Wert verglichen. Liegen die Werte innerhalb eines intern festgelegten Toleranzfensters kann die Referenzierung erfolgreich abgeschlossen werden. Des weiteren kann ein „ungünstig“ liegender Indexpuls (Indexpuls kurz vor bzw. hinter Schalterflanke) „korrigiert“ werden.

HINWEIS

Wurde der Abstand zwischen Indexpuls und Referenzereignis über die zulässige Toleranz hinaus geändert, wird die Referenzierung mit „ERROR REFERENCED“ abgebrochen. Eine neue Vermessung des Abstandes vom Referenzereignis zum Indexpuls kann nötig sein. Hierzu muss der Parameter „Abstand zum Index“ zu „0“ gesetzt werden.

HINWEIS

Das Modul sollte immer in der gleichen Lage referenziert werden.

HINWEIS

Bei regelmäßigm Auftreten des Fehlers kontaktieren sie bitte den SCHUNK-Service.

- Geschwindigkeit links (0x04) / rechts (0x05)

Zur Referenzierung wird eine Geschwindigkeitsfahrt ausgeführt. Fährt das Modul auf einen festen Anschlag, wird dieser als Referenzpunkt erkannt. Die Drehrichtung wird über „links“ bzw. „rechts“ festgelegt.

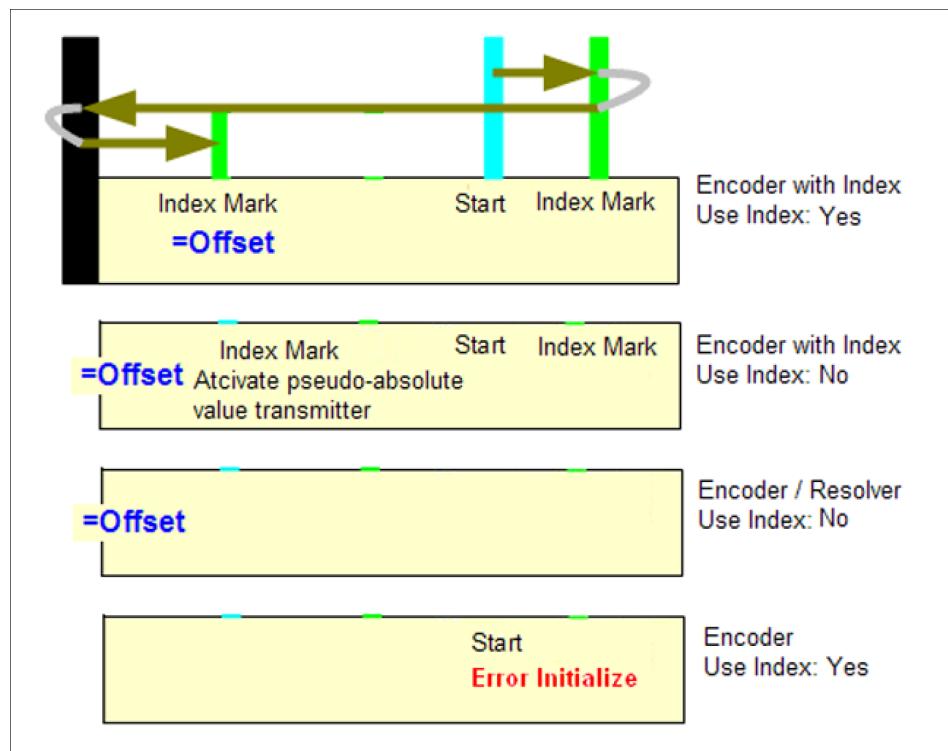


Abb. 8

HINWEIS

Diese Art von Referenzierung ist nur empfehlenswert, wenn ein fester Endanschlag vorhanden ist!

- Geschwindigkeit mit Hubüberwachung links (0x06) / rechts (0x07)

Zusätzlich zu oben genannter Vorgehensweise wird nach Anfahren des ersten festen Anschlages zum gegenüberliegenden festen Anschlag gefahren. Der zurückgelegte Weg muss größer als die Differenz der *Softwareendanschläge* ([☞ 3.2.5, Seite 136](#)) sein => Referenzierung erfolgreich.

HINWEIS

Diese Art von Referenzierung ist nur empfehlenswert, wenn ein fester Endanschlag vorhanden ist!

- Strom links (0x08) / rechts (0x09)

Es wird eine Stromfahrt ausgeführt. Der Strom wird solange erhöht bis sich das Modul bewegt. Überschreitet der Strom den *max. Referenzstrom* ([☞ 3.2.3, Seite 125](#)), wird davon ausgingen, dass ein fester Anschlag erreicht worden ist, der als Referenzpunkt erkannt wird.

HINWEIS

Diese Art von Referenzierung ist nur empfehlenswert, wenn ein fester Endanschlag vorhanden ist!

HINWEIS

Eine Verklemmung, Schwerkängigkeit in der Mechanik oder ein „vergessenes“ Werkstück kann ebenfalls zu einer Überschreitung des Nennstroms führen. Dies wird dann ebenfalls als fester Endanschlag interpretiert, obwohl keiner vorhanden ist.

- Strom mit Hubüberwachung links (0x0A) / rechts (0x0B)

Zusätzlich zu oben genannter Vorgehensweise wird nach Anfahren des ersten festen Endanschlages zum gegenüberliegenden festen Endanschlag gefahren. Der zurückgelegte Weg muss größer als die Differenz der *Softwareendanschläge* ([☞ 3.2.5, Seite 136](#)) sein => Referenzierung erfolgreich.

HINWEIS

Diese Art von Referenzierung ist nur empfehlenswert, wenn ein fester Endanschlag vorhanden ist!

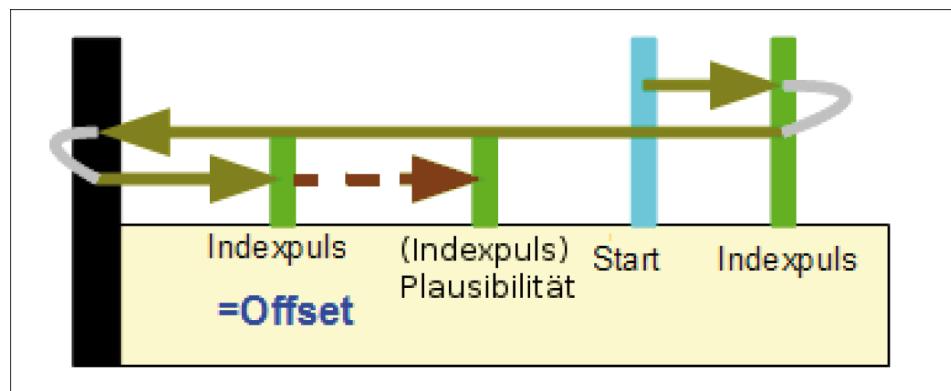


Abb. 9 Referenzierung auf Anschlag mit aktiver Indexspur

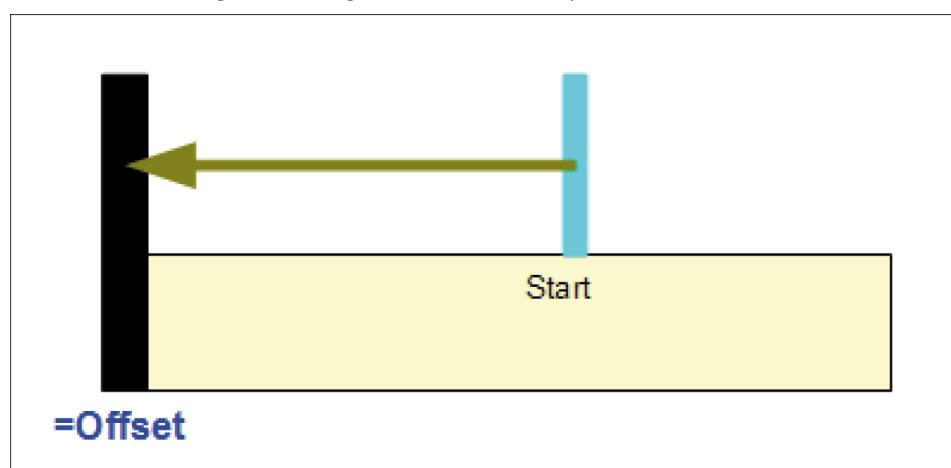


Abb. 10 Referenzierung auf Anschlag

- Keine (0x0C)
Nach Absenden des Befehls *CMD REFERENCE* ([☞ 2.2.1, Seite 32](#)) wird die aktuelle Position als Referenzposition angesehen.

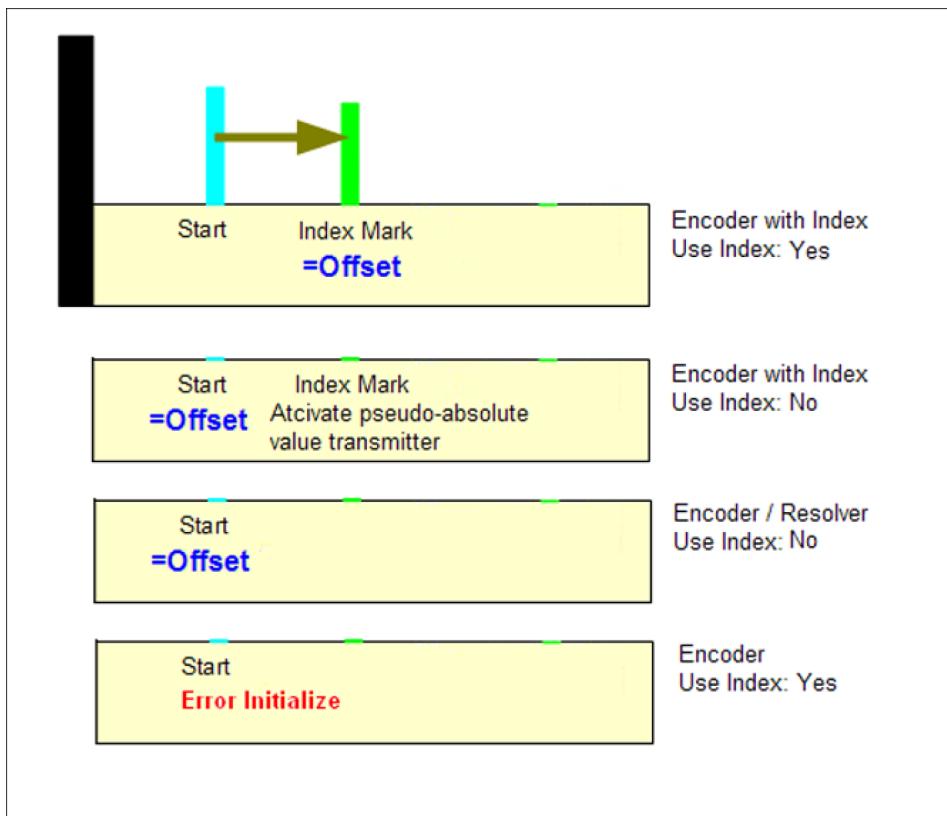


Abb. 11 Referenzierung „Keine“

- Manuell (0x0D)

Nach Absenden des Befehls *CMD REFERENCE*

(☞ [2.2.1, Seite 32](#)) wird, falls das Modul bereits referenziert ist, eine „simulierte“ Referenzfahrt durchgeführt. Das Modul wird sich zur Referenzmarke bewegen und je nach *Einstellung* (☞ [3.2.3, Seite 125](#)) anschließend auf Position „0“ fahren. Sollte das Modul nicht referenziert sein, wird eine *manuelle Referenzierung* (☞ [2.2.2, Seite 33](#)) gestartet.

HINWEIS

Ist bereits einmal eine manuelle Referenzierung (☞ [2.2.2, Seite 33](#)) erfolgreich durchgeführt worden, werden zuvor eingestellte Referenzierarten automatisch zu „Manuell“ umgestellt.

DATEN: (UInt16) 0 .. 65535

3.2.3.2 Max. Referenzstrom

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Stromvorgabe in [%] vom *Nennstrom des Motors* ([☞ 3.2.1, Seite 119](#)). Der Referenzstrom überschreitet den vorgegebenen Wert nicht.

DATEN: (UInt8) 0 .. 255

3.2.3.3 Geschwindigkeit

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Geschwindigkeitsvorgabe für Referenzfahrten mit internem bzw. externem Referenzschalter und Geschwindigkeitsreferenzfahrten. Dies gilt auch für manuelle Referenzierung. Dieser Wert stellt hierbei 100% Geschwindigkeit dar.

Soll nach dem Referenzieren die *Position „0.0“* ([☞ 3.2.3, Seite 125](#)) angefahren werden, wird diese Geschwindigkeit für den Positionierbefehl verwendet.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.3.4 Beschleunigung

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Beschleunigungsvorgabe für Referenzfahrten mit internem bzw. externem Referenzschalter und Geschwindigkeitsreferenzfahrten, sowie manueller Referenzierung. Soll nach dem Referenzieren die *Position „0.0“* ([☞ 3.2.3, Seite 125](#)) angefahren werden, wird diese Beschleunigung für den Positionierbefehl verwendet.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.3.5 Offset Nullpunkt

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Positionsoffset nach erfolgreicher Referenzierung (Nullpunktverschiebung)

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.3.6 Nach Referenzieren 0 anfahren

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Nach erfolgreicher Referenzierung wird die Position „0.0“ angefahren. Die *Geschwindigkeit* ([☞ 3.2.3, Seite 125](#)) und *Beschleunigung* ([☞ 3.2.3, Seite 125](#)) werden beachtet.

DATEN: (Bool) JA / NEIN

3.2.3.7 Timeout

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Zeit, die die Referenzfahrt maximal dauern darf.

DATEN: (Float) Je nach *Einheitensystem* ([☞ 1.5, Seite 23](#))[s] bzw. [ms]

3.2.3.8 Nutzung Index

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Indexspur wird zur Referenzierung benutzt.

	VORSICHT Soll mit Indexpuls referenziert werden. Liegt der Indexpuls an einer ungünstigen Stelle, kann es vorkommen, dass bei mehrmaliger Referenzierung die Positionen jeweils um eine Motordrehung abweichen. Abhilfe: Referenzmarke leicht verschieben. Dies gilt für alle Referenzmarken außer "Interner Schalter" und "Externer Schalter".
---	---

3.2.3.9 Abstand zum Index

ZUGRIFFSRECHTE: Advanced

BESCHREIBUNG: Der Abstand des „Referenzereignisses“ zum Indexpuls wird hier gespeichert. Wird dieser Wert zu „0“ gesetzt, so wird beim nächsten Referenzieren dieser Wert erneut ermittelt.

3.2.3.10 Max. Distanz Schalter

ZUGRIFFSRECHTE: Advanced

BESCHREIBUNG: Maximal erlaubter Abstand vom Referenzereignis (Schalterflanke dedektiert) zum Indexpuls. Wird in Encoderticks angegeben. Siehe auch *Abstand zum Index* ([☞ 3.2.3.9, Seite 132](#)).

3.2.4 Regler

3.2.4.1 KR Strom

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Proportionalanteil des Stromreglers. Bei *Strombegrenzungsregelung* ([☞ 3.2.4, Seite 133](#)) ist dies der Proportionalanteil für die Strombegrenzungsregelung

DATEN: (Float)

3.2.4.2 TN Strom

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Integralanteil des Stromreglers. Wird bei *Strombegrenzungsregelung* ([☞ 3.2.4, Seite 133](#)) nicht benötigt.

DATEN: (Float)

3.2.4.3 KR Geschwindigkeit

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Proportionalanteil des Geschwindigkeitsreglers

DATEN: (Float)

3.2.4.4 TN Geschwindigkeit

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Integralanteil des Geschwindigkeitsreglers

DATEN: (Float)

3.2.4.5 KR Position

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Proportionalanteil des Positionsreglers

DATEN: (Float)

3.2.4.6 Positionsabweichnung

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Positionsfenster, in dem die Positionsregelung abgebrochen wird (je nach *Bremsen Konfiguration* ([☞ 3.2.7, Seite 150](#))), wird weitergeregelt oder die Bremse fällt ein bzw. es wird Position erreicht gemeldet.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.4.7 Struktur

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Struktureller Aufbau des Regelkreises.

- Strom Geschwindigkeit (0x00)
Stromregelung und Geschwindigkeitsregelung funktionieren unabhängig voneinander.
- Kaskade (0x01)
Positions-, Geschwindigkeits- und Stromregler sind in Kaskade geschaltet => Es sind stromgeregelte (eingestellter Strom wird nicht überschritten) Positions-, bzw. Geschwindigkeitsfahrten möglich (z.B. kein Vorpositionieren bei einem Greifvorgang notwendig). In diesem Modus wird der vorgegebene Strom („*SET TARGET CURRENT*“ ([☞ 2.2.14, Seite 49](#)) bei allen Verfahrarten nicht überschritten.
- Geschwindigkeit mit Strombegrenzung (0x02)
Stromregelung ist nicht aktiv. Bei Geschwindigkeits- bzw. Positionsfaehrten wird der vorgegebene Strom („*SET TARGET CURRENT*“ ([☞ 2.2.14, Seite 49](#)) begrenzt. Hierbei wird im Gegensatz zur Kaskade der Strom nicht geregelt, sondern begrenzt (Strombegrenzungsregelung).
- Geschwindigkeit mit PWM Begrenzung (0x02)
Stromregelung ist nicht aktiv. Bei Geschwindigkeits- bzw. Positionsfaehrten wird das Tastverhältnis der PWM begrenzt. Das Verhältnis von Strom zu Tastverhältnis wird über den *Anschlusswiderstand* ([☞ 3.2.1, Seite 119](#)) des Motors berechnet.

HINWEIS

Da das Tastverhältnis der PWM direkt begrenzt wird (Spannungsbegrenzung), ist es möglich, dass der Antrieb nicht mehr seine volle Geschwindigkeit erreicht, daher können Positionsfaahrten eventuell erheblich länger dauern als vorausberechnet.

DATEN: (3Bit) 0 .. 7

HINWEIS

Bei einer Änderung der Reglerstruktur müssen evtl. die Reglerparameter angepasst werden!

3.2.4.8 Max. Überschwinger

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Fährt das Modul bei einer Positionsfahrt über dieses Positionsfenster, so wird die Fehlermeldung *ERROR OVERSHOOT* ([☞ 2.8.2, Seite 109](#)) erzeugt. Dieser Wert muss größer der maximal erlaubten *Positionsabweichung* ([☞ 3.2.4, Seite 133](#)) eingestellt werden.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.5 Gerät

3.2.5.1 Serien Nummer

ZUGRIFFSRECHTE: *Root* ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Seriennummer des Moduls

DATEN: (UInt32)

3.2.5.2 Einheitensystem

ZUGRIFFSRECHTE:

Nutzer ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Alle Ein- und Ausgaben zum Modul bzw. vom Modul benutzen das eingestellte *Einheitensystem* ([☞ 1.5, Seite 23](#)). Wird das Einheitensystem von Float ([mm](0x00),[m](0x01),[Inch](0x02),[rad](0x03),[Grad](0x04)) in Integer umgestellt ([μm](0x06),[μGrad](0x07),[μInch](0x08),[Milligrad](0x09)), werden alle Zeiteinheiten und Stromeinheiten ebenfalls als Ganzzahl interpretiert => [s] -> [ms]; [A] -> [mA]. Internes Einheitensystem hat die Nummer (0x05).

DATEN: (5Bit) 0 .. 31

	ACHTUNG
<p>Eine Umstellung des Einheitensystems (☞ 1.5, Seite 23) betrifft eine große Anzahl von EEPROM Parametern, welche ebenfalls sofort angepasst werden!</p>	

3.2.5.3 Kommunikationsmodus

ZUGRIFFSRECHTE:

Nutzer ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Aktives Kommunikationssystem wird eingestellt.

- Auto (0x00)

Es wird versucht das Kommunikationssystem automatisch zu finden und einzustellen. Bisher geprüfte *Hardware* ([☞ 6.5, Seite 202](#)).

HINWEIS

Ist ein Modul gleichzeitig an Seriell- und Profibus-Schnittstelle angeschlossen, wird die Profibus-Schnittstelle (hardwareabhängig) sofort nach dem Einschalten des Moduls erkannt. In diesem Fall besteht dann keine Möglichkeit über die Serielle-Schnittstelle zu kommunizieren.

HINWEIS

In dieser Betriebsart wird versucht das angeschlossene Kommunikationsinterface selbstständig zu finden. Hierbei können die ersten Datenpaket welche an das Modul gesendet werden vernichtet oder verfälscht werden. Evtl. kann das übergeordnete System dadurch gestört werden.

HINWEIS

Ist die Inbetriebnahme erfolgt, sollte die Kommunikationsart "Auto" zum tatsächlich verwendetem Kommunikationssystem geändert werden.

- Serielle Kommunikation (0x01)
- CAN (0x02)
- Profibus DPV0 (0x03)
- Serielle Kommunikation ohne Spontanmeldung (0x04)
Spontanmeldungen ([☞ 2.3, Seite 54](#)) sind deaktiviert. (inkl. INFO BOOT)

DATEN: (5Bit) 0 .. 31

3.2.5.4 Motordrehrichtung

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Drehrichtung des Motors wird festgelegt.

	! WARNUNG
<p>Falsche Konfiguration kann zu unerwarteten Effekten führen. (Modul dreht unerwartet sehr schnell!)</p>	

DATEN: (Bool) JA / NEIN

3.2.5.5 Positionsmessung invertieren

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Messrichtung des Positionsmesssystems wird festgelegt. Vertauschte A und B Spur des Encoders können per Software getauscht werden.

	! WARNUNG
<p>Falsche Konfiguration kann zu unerwarteten Effekten führen. (Modul dreht unerwartet sehr schnell!)</p>	

HINWEIS

Werden gleichzeitig Drehrichtung des Motors und des Positionsmesssystems umgekehrt, kann aus einem „linksdrehenden“ Modul ein „rechtsdrehendes“, bzw. aus einem „positiv“ öffnenden Greifer ein „positiv“ schließender Greifer konfiguriert werden.

DATEN: (Bool) JA / NEIN

3.2.5.6 Positionsrampe

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Rampentyp für Positionsfahrt wird eingetragen.

- Trapez (0x00)

Für die Berechnung des Verfahrprofils wird ein Trapez zu Grunde gelegt. Dieses Profil wird, falls erforderlich, dynamisch bei „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)) (Kurvenbahnen) zugeschaltet. Eine Berechnung der Verfahrzeit ist möglich. (Umschaltpunkte werden nach Zeiten gesteuert)

- Ruckbegrenzt (0x01)

Für Positionsfahrten „MOVE POS“ ([☞ 2.2.3, Seite 35](#)), „MOVE POS REL“ ([☞ 2.2.4, Seite 37](#)) und „MOVE POS LOOP“ ([☞ 2.2.7, Seite 43](#)) wird eine Bahn mit Ruckbegrenzung berechnet. Nur in diesem Rampentyp wird der Parameter „Ruckbegrenzt“ („SET TARGET JERK“ ([☞ 2.2.13, Seite 48](#))) verwendet. Bei Kurvenbahnen „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)), „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)) wird dieses Profil deaktiviert (Dieses Vorgehen ist aus Rechenzeitgründen notwendig).

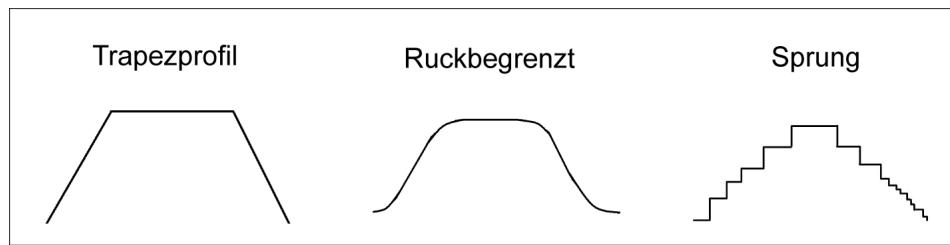
- Trapez SRU (0x02)

Für die Berechnung des Verfahrprofils wird ein Trapez zu Grunde gelegt. Eine Berechnung der Verfahrzeit erfolgt nicht. (Umschaltpunkte werden nach Positionen gesteuert)

- Sprung (0x03)

Hier wird kein Bahnprofil berechnet sondern direkt der Positionssprung vorgegeben. Die interne Bahnplanung ist abgeschalten. Je nach Interpolationstakt des externen Interpolators kann es notwendig sein die *Reglerparameter* ([☞ 3.2.4, Seite 133](#)) anzupassen. Dieses Verfahrprofil wird nur von „MOVE POS TIME“ ([☞ 2.2.5, Seite 39](#)), „MOVE POS TIME REL“ ([☞ 2.2.6, Seite 41](#)) unterstützt. „MOVE POS“ ([☞ 2.2.3, Seite 35](#)), „MOVE POS REL“ ([☞ 2.2.4, Seite 37](#)) und „MOVE POS LOOP“ ([☞ 2.2.7, Seite 43](#)) verwenden in diesem Fall das ruckbegrenzte Bahnprofil.

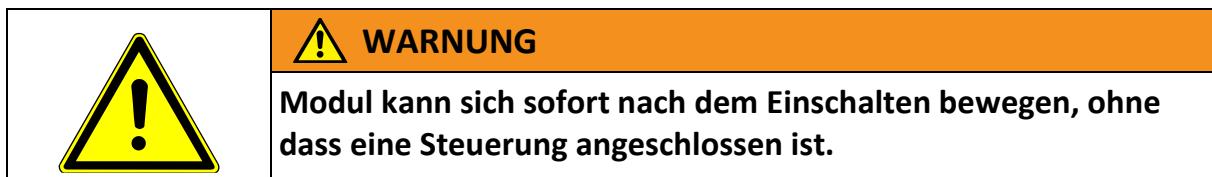
DATEN: (3Bit) 0-7



3.2.5.7 Programm nach Neustart

ZUGRIFFSRECHTE: Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Direkt nach dem Einschalten des Moduls wird das im EEPROM hinterlegte Programm ab Zeile „0“ gestartet.



HINWEIS

Das Programm wird erst dann gestartet, wenn das Modul eine Kommunikationsschnittstelle findet, daher sollte die Serielle Schnittstelle als Kommunikationsart eingestellt ([☞ 2.4.1, Seite 59](#)) worden sein. Alle anderen Bussysteme lassen einen Start erst zu, wenn ein aktiver Master gefunden wurde.

DATEN: (Bool) JA / NEIN

3.2.5.8 Endlos

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Die Achse kann endlos drehen.

DATEN: (Bool) JA / NEIN

HINWEIS

Für Greifer ist diese Option nicht empfehlenswert!

3.2.5.9 Digitale Eingänge

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Verwendung der digitalen Eingänge.

- Normal (0x00)

Digitale Eingänge können von außen beschaltet werden ohne irgendwelche Aktionen im Modul auszulösen. Diese können jederzeit über „CMD DIO“ ([☞ 2.6.4, Seite 91](#)) abgefragt werden.

- Programm (0x01)

HINWEIS

Der Programmabetrieb über digitale Eingänge ist nur bei eingestellter Seriellen Kommunikation ([☞ 3.2.5, Seite 136](#)) in jedem Betriebszustand des Moduls möglich. Alle anderen Kommunikationsarten lassen einen Start erst zu, wenn ein aktiver Master gefunden wurde.

Vorher einprogrammierte „Sätze“ können ausgeführt werden. Bevor der allererste Satz ausgeführt werden kann, müssen einmalig alle Eingänge auf „low“ gesetzt werden. Anschließend können bis zu 7 Sätze ([☞ 2.5.5, Seite 73](#)) ausgewählt werden. Der Eingang 1 dient als „Freigabe“. Die Stellung der Eingänge 2 - 4 werden nur bei einer steigenden Flanke am Eingang 1 übernommen. Wird nur der Eingang 1 gesetzt, wird eine Referenzfahrt ([☞ 2.2.1, Seite 32](#)) ausgeführt.

Schaltvorgang:

- 1 Alle Eingänge auf „low“ setzen.
- 2 Eingänge 2, 3, 4 beliebig setzen (siehe Tabelle unten).
- 3 Eingang 1 auf „high“ setzen.

	Eing. 1	Eing. 2	Eing. 3	Eing. 4	Ausführung
FSK	IN 0	IN 1	IN 2	IN 3	
	low	low	low	low	Ausgangszustand
	low -> high	low	low	low	Referenzfahrt
	low -> high	high	low	low	Programmsatz 0
	low -> high	low	high	low	Programmsatz 1
	low -> high	high	high	low	Programmsatz 2
	low -> high	low	low	high	Programmsatz 3
	low -> high	high	low	high	Programmsatz 4
	low -> high	low	high	high	Programmsatz 5
	low -> high	high	high	high	Programmsatz 6

HINWEIS

Bei aktivierten externen Schalter ([☞ 3.2.3, Seite 125](#)) ist die Betriebsart „Programm“ nicht mehr möglich.

DATEN: (3Bit) 0 .. 7

3.2.5.10 Digitale Ausgänge

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Verwendung der digitalen Ausgänge.

- Normal (0x00)

Digitale Ausgänge können über „CMD DIO“ ([☞ 2.6.4, Seite 91](#)) gesetzt werden.

- Status + Bewegung OUT2 (0x01)

Der Modulzustand wird an die digitalen Ausgänge gemeldet.
Out2 meldet, dass das Modul in Bewegung ist.

	FSK	Modulzustand bei „low“
Ausgang 1	OUT0	Referenziert
Ausgang 2	OUT1	Fehlermeldung
Ausgang 3	OUT2	Bewegung läuft
Ausgang 4	OUT3	Bewegung beendet

- Status + Position erreicht OUT2 (0x02)

Der Modulzustand wird an die digitalen Ausgänge gemeldet.
Out2 meldet, ob das Modul die Zielposition erreicht hat.

	FSK	Modulzustand bei „low“
Ausgang 1	OUT0	Referenziert
Ausgang 2	OUT1	Fehlermeldung
Ausgang 3	OUT2	Zielposition erreicht
Ausgang 4	OUT3	Bewegung beendet

- Status + Bremse OUT2 (0x03)

Der Modulzustand wird an die digitalen Ausgänge gemeldet.
Out2 meldet den Zustand der Bremse.

	FSK	Modulzustand bei „low“
Ausgang 1	OUT0	Referenziert
Ausgang 2	OUT1	Fehlermeldung
Ausgang 3	OUT2	Zustand der Bremse
Ausgang 4	OUT3	Bewegung beendet

- Status + Warnung OUT2 (0x04)

Der Modulzustand wird an die digitalen Ausgänge gemeldet.
Out2 meldet, ob eine Warnung anliegt.

	FSK	Modulzustand bei „low“
Ausgang 1	OUT0	Referenziert
Ausgang 2	OUT1	Fehlermeldung
Ausgang 3	OUT2	Warnung vorhanden
Ausgang 4	OUT3	Bewegung beendet

- Status + Programmablauf OUT2 (0x05)

Der Modulzustand wird an die digitalen Ausgänge gemeldet.
Out2 zeigt an, ob sich das Modul im "Programmbetrieb" befindet.

	FSK	Modulzustand bei „low“
Ausgang 1	OUT0	Referenziert
Ausgang 2	OUT1	Fehlermeldung
Ausgang 3	OUT2	Programmablauf
Ausgang 4	OUT3	Bewegung beendet

DATEN: (3Bit) 0 .. 7

	ACHTUNG
	Beim Einschalten des Moduls haben die Ausgänge kurzfristig undefinierte Zustände. Dies kann unter Umständen zu Zerstörung angeschlossener Hardware führen.

3.2.5.11 Analoger Ausgang

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Verwendung des analogen Ausgangs (0 - 10V).

- Keine (0x00)
wird nicht genutzt => 0V
- Position (0x01)
Position wird in analogen Wert umgesetzt.
Position 0.0 .. 5V
Maximum der absolut Werte von Minimal bzw. Maximal
Position werden auf 0V bzw. 10V abgebildet. z.B. Minimal
Position ist -5.0, maximal Position ist +10.0 => Position = 0[V],
+10.0 = 10.0[V], -5.0 = 2.5[V]
- Geschwindigkeit (0x02)
Geschwindigkeit wird in analogen Wert umgesetzt:
 - Negative maximale Geschwindigkeit = 0V
 - Stillstand = 5V
 - Positive maximale Geschwindigkeit = 10V
- Strom (0x03)
0A => 0V
Maximal Strom = 10V, hier ([☞ 3.2.1, Seite 119](#)) konfiguriert.
- Maximum (0x04)
Analog Ausgang = 10V

DATEN: (3Bit) 0 .. 7

3.2.5.12 ID

ZUGRIFFSRECHTE: Nutzer ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Eindeutige individuelle Modulkennung. Siehe auch *SET CONFIG* ([☞ 2.4.1, Seite 59](#)).

DATEN: (UInt8) 0 .. 255

3.2.5.13 Gruppe

ZUGRIFFSRECHTE: Nutzer ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Eindeutige Gruppenkennung des Moduls. Siehe auch *SET CONFIG* ([☞ 2.4.1, Seite 59](#)). Bei Profibus wird hier „Real No Add Change“ gespeichert.

DATEN: (UInt8) 0 .. 255

3.2.5.14 CAN Baudrate

ZUGRIFFSRECHTE: Nutzer ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Übertragungsrate für CAN. Gültige Werte: 50, 100, 125, 250, 500, 1000. Siehe auch *SET CONFIG* ([☞ 2.4.1, Seite 59](#)).

DATEN: (UInt16) 0 .. 65535

3.2.5.15 Min. Position

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Minimal erlaubte Position (Softwareendanschlag). Wird bei gesetztem „Endlos“ ([☞ 3.2.5, Seite 136](#)) ignoriert. Wird beim Referenzieren ([☞ 3.2.3, Seite 125](#)) mit „Hubkontrolle“ verwendet.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.5.16 Max. Position

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Maximal erlaubte Position (Softwareendanschlag) Wird bei gesetztem „Endlos“ ([☞ 3.2.5, Seite 136](#)) ignoriert. Wird beim Referenzieren ([☞ 3.2.3, Seite 125](#)) mit „Hubkontrolle“ verwendet.

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.5.17 Schleppfehler

ZUGRIFFSRECHTE: Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Schleppfehler. Dieser Wert darf während einer Positionsfahrt nicht überschritten werden. Bei Überschreitung kommt es zu einer Fehlermeldung („*ERROR TOW*“ ([☞ 2.8.2, Seite 109](#))).

DATEN: (4 Byte) Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert.

3.2.5.18 Min. Temperatur

ZUGRIFFSRECHTE:

Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Minimal zulässige Arbeitstemperatur. Bei Unterschreitung kommt es zu einer Fehlermeldung.

(„*ERROR TEMP LOW*“ ([☞ 2.8.2, Seite 109](#)))

DATEN: (Float) [°C]

3.2.5.19 Max. Temperatur

ZUGRIFFSRECHTE: *Root* ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Maximal zulässige Arbeitstemperatur. Bei Überschreitung kommt es zu einer Warnung. Sollte die Temperatur nicht innerhalb 1[min] sinken, kommt es zu einer Fehlermeldung.

(„*ERROR TEMP HIGH*“ ([☞ 2.8.2, Seite 109](#)))

DATEN: (Float) [°C]

3.2.5.20 Serielle Kommunikation Baudrate

ZUGRIFFSRECHTE: *Nutzer* ([☞ 1.6.1, Seite 25](#))

BESCHREIBUNG: Übertragungsrate für Serielle Kommunikation.
Gültige Werte: 1200, 2400, 4800, 9600, 19200, 38400. Siehe auch
SET CONFIG ([☞ 2.4.1, Seite 59](#)).

DATEN: (UInt16) 0 .. 65535

3.2.6 Positionierung

3.2.6.1 Serien Nummer

ZUGRIFFSRECHTE:

Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Seriennummer des Positionsmesssystems

DATEN: (UInt32)

3.2.6.2 Typ

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Der Typ des Messsystems wird festgelegt.

- Encoder (0x00)
Encoder Messsystem ohne Indexspur. *Parameter 1* ([☞ 3.2.6, Seite 146](#)) beachten.
- Encoder Index (0x01)
Encoder mit Indexspur. Bei allen *Referenzfahrten* ([☞ 3.2.3, Seite 125](#)) wird die Indexspur ausgewertet. Um sicherzugehen, dass eine Indexspur am richtigen Ort gefunden werden kann, kann es bei einigen Referenzierarten vorkommen, dass sich der Antrieb einige Male mit kurzen Bewegungen hin und her bewegt, bzw. kleine Bewegungen in die „falsche“ Richtung macht. *Parameter 1* ([☞ 3.2.6, Seite 146](#)) beachten.
- Resolver (0x02)
Resolversystem bei dem der Erregerstrom einstellbar ist. *Parameter 1* ([☞ 3.2.6, Seite 146](#)) und *Parameter 2* ([☞ 3.2.6, Seite 146](#)) beachten.

HINWEIS

Bei einem Motor mit Resolver ([☞ 3.2.6, Seite 146](#)) kann es beim "Positioning ramp type" ([☞ 3.2.5, Seite 136](#)) "Ruckbegrenzt" zum "Rattern" kommen. In diesem Fall muss ein anderer "Positioning ramp type" ([☞ 3.2.5, Seite 136](#)) gewählt werden.

- Encoder differenziell (0x06)
Differentieller Encoder ohne Indexspur. *Parameter 1* ([☞ 3.2.6, Seite 146](#)) beachten!
- Encoder Index differenziell (0x07)
differentieller Encoder mit Indexspur. Bei allen *Referenzfahrten* ([☞ 3.2.3, Seite 125](#)) wird die Indexspur ausgewertet. Um sicherzugehen, dass eine Indexspur am richtigen Ort gefunden werden kann, kann es bei einigen Referenzierarten vorkommen, dass sich der Antrieb einige Male mit kurzen Bewegungen hin und her bewegt, bzw. kleine Bewegungen in die „falsche“ Richtung macht. Die Module bewegen sich auch bei Referenzierart „Keine“, da nach dem nächsten Indexpuls gesucht wird. *Parameter 1* ([☞ 3.2.6, Seite 146](#)) beachten!

DATEN: (UInt8)

3.2.6.3 Einbaulage

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Die Einbaulage des Positionsmesssystems.

- Antriebsseitig: Das Positionsmesssystem ist direkt auf dem Antrieb montiert.
- Abtriebsseitig: Das Positionsmesssystem ist direkt auf dem Abtrieb montiert.
- Zwischen Getriebeübersetzungen: Das Positionsmesssystem ist mitten im Getriebe montiert.

DATEN: (2Bit) 0 .. 3

3.2.6.4 Parameter 1

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Parameter 1 für das Positionsmesssystem

- Encoder: Ticks pro Umdrehung (4fach Auswertung)
- Resolver: Frequenz der Spannung an der Erregerspule in [kHz]. Erlaubte Werte: 8 [kHz], 4 [kHz], 2 [kHz], 1 [kHz].

DATEN: (UInt16)

3.2.6.5 Parameter 2

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Parameter 2 für das Positionsmesssystem

- Encoder: nicht benötigt
- Resolver: Amplitude der Eingangsspannung an der Erregerspule [%]. Muss messtechnisch ermittelt werden. Die Ausgangsspannung an den Empfängerspulen darf nicht in die Sättigung gehen. **Daten:** (UInt16)

3.2.6.6 Offset Messsystem

ZUGRIFFSRECHTE: *Advanced* ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Verdrehung des Positionsmesssystems gegenüber den Motorphasen. Wird im voreingestellten *Einheitensystem* ([☞ 1.5, Seite 23](#)) konfiguriert. Dieser Wert wird unter Umständen automatisch ermittelt. Siehe hierzu *Stillstandskommutierung* ([☞ 1.8, Seite 29](#)). Wird dieser Wert zu „0“ gesetzt wird der „*Sinus-Zeiger*“ ([☞ 1.8, Seite 29](#)) neu gesucht. Die *Referenzierung* ([☞ 1.7, Seite 27](#)) wird hierbei gelöscht.

DATEN: (Float)

HINWEIS

Zur Zeigersuche sollte das Modul in alle Richtungen frei beweglich sein. Modul wird ruckartig bis zu zwei Motorumdrehungen bewegt. Eine Kommunikation mit dem Modul ist in dieser Zeit nicht möglich. Dieser Vorgang kann bis zu 30sek. benötigen.

3.2.6.7 Bewegungsschwelle

ZUGRIFFSRECHTE:

Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Der Wert in [%] von der *Höchstgeschwindigkeit* ([☞ 3.2.1, Seite 119](#)). Bei Unterschreitung wird das Modul als stehend behandelt. Siehe *Motion Tool Schunk Statusanzeige* ([☞ 4.4.1, Seite 166](#))

DATEN: (Float)

3.2.6.8 ADC Offset

ZUGRIFFSRECHTE: *Root* ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: ADC Offset um das Eingangssignal zu „zentrieren“. Wird nur für Cos/Sin Messsysteme benötigt und Resolver.

DATEN: (Int16)

3.2.7 Bremse

3.2.7.1 Serien Nummer

ZUGRIFFSRECHTE: Root ([☞ 1.6.5, Seite 26](#))

BESCHREIBUNG: Seriennummer der Bremse.

DATEN: (UInt32)

3.2.7.2 Typ

ZUGRIFFSRECHTE: Advanced ([☞ 1.6.4, Seite 26](#))

BESCHREIBUNG: Typ der Bremse. Über die konfigurierte Motorspannung und den Bremsentyp wird die Bremsenspannung per Software angepasst.

DATEN: (UInt16)

	ACHTUNG
Eine Fehlkonfiguration führt zu einer nicht funktionstüchtigen oder zur Zerstörung der Bremse!	

3.2.7.3 Bremsen Nutzung

ZUGRIFFSRECHTE Profi ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Benutzung der Bremse.

- Wird nicht genutzt (0x00)
Bremse wird nicht benutzt. Fällt nur bei Spannungsausfall ein.
Wird sofort nach dem Starten des Moduls einmalig geöffnet.
- Nur im Fehlerfall (0x01)
Bremse fällt nur im Fehlerfall ein. Ansonsten wird der Antrieb dauerhaft geregelt.
- Normal (0x02)
Bremse fällt im Fehlerfall und am Ende der Bewegung ein. Ist eine Bremse konfiguriert, ist der *Pseudoabsolutwertgeber* ([☞ 1.7, Seite 27](#)) möglicherweise aktiv, falls weitere Bedingungen erfüllt sind.

DATEN: (UInt16) 0 .. 65535

3.2.7.4 Timeout

ZUGRIFFSRECHTE: *Profi* ([☞ 1.6.3, Seite 26](#))

BESCHREIBUNG: Zeit, die die Bremse benötigt das Magnetfeld auf- bzw. abzubauen. Je nach *Einheitensystem* ([☞ 1.5, Seite 23](#)) [s] bzw. [ms].

DATEN: (Float)

HINWEIS

Eine Änderung des Wertes hat keinen Einfluss auf die Zyklengeschwindigkeit von Bewegungsbefehlen. Eine falsche Einstellung kann zu vorzeitigen Bremsenverschleiß führen.

4 Motion Tool Schunk

Die Software dient der Inbetriebnahme und dem Testen von SCHUNK-Modulen mit SCHUNK Motion Protokoll.

4.1 Voraussetzungen

- Betriebssystem: Windows, 2000, XP, Vista, 7
- Grafik: 1024x768 mit 16-bit Farbtiefe

4.2 Erste Schritte

Alle Module müssen über zumindest ein Kommunikationssystem mit dem PC verbunden werden. Derzeit werden von der Software folgende Kommunikationsinterfaces unterstützt:

- Serielle Kommunikation (COM-Ports)
- CAN: Karten von *Vector Informatik GmbH*
- CAN: Karten von *esd electronic system design gmbh*
- CAN: Konverter PCAN-USB von *PEAK-System Technik GmbH*
- CAN: Karten von *IXXAT Automation GmbH* (VCI-Treiber V3)
- CAN: Karten von *Softing AG*
- Profibus DPV0: Karten von *Hilscher* (Vorkonfigurierung notwendig!)
- Profibus DPV0: Kontrollerreihe CP56xx *Siemens AG* (Vorkonfigurierung notwendig!)

Hinweise zur Installation finden Sie bei den jeweiligen Herstellern. Eine Liste der unterstützten Hardware mit weiteren Details ist im *Anhang* ([☞ 6, Seite 185](#)) zu finden.

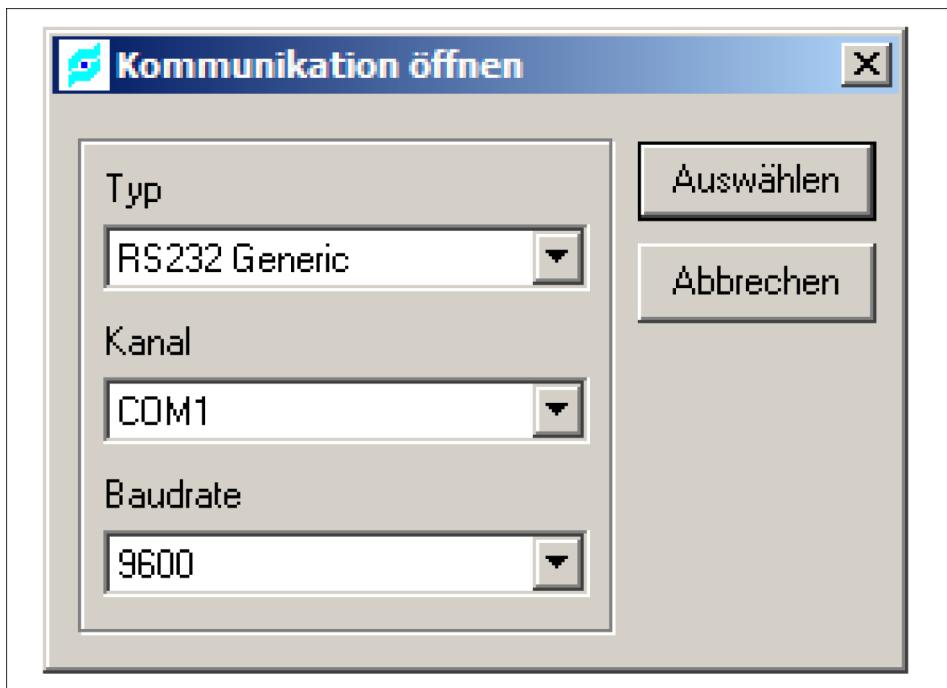


Abb. 13

Nach der Auswahl des Kommunikationsinterfaces wird dieses automatisch geöffnet. Mit Hilfe der Bussuche werden alle angeschlossenen und eingeschalteten Module nun automatisch erkannt. Falls unter *Grundeinstellungen* ([4.3.4, Seite 160](#)) die Option „Keine automatische Initialisierung von Modulen“ aktiviert ist, muss eine Modulinitialisierung (Modulfenster öffnen) nur manuell erfolgen.

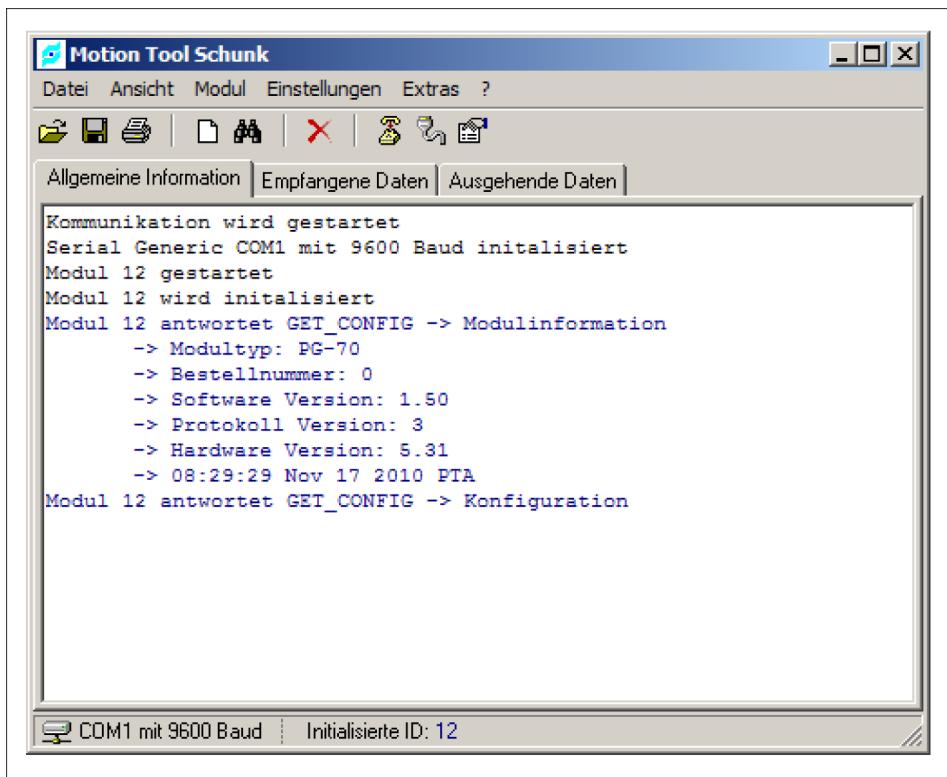


Abb. 14

Neben dem Hauptfenster gibt es für jedes gefundene Modul ein Modulfenster, mit dem alle Funktionen des jeweiligen Moduls getestet werden können.

Mit *F1* lässt sich jederzeit das Hilfesystem starten, um weitere Informationen zu bekommen.

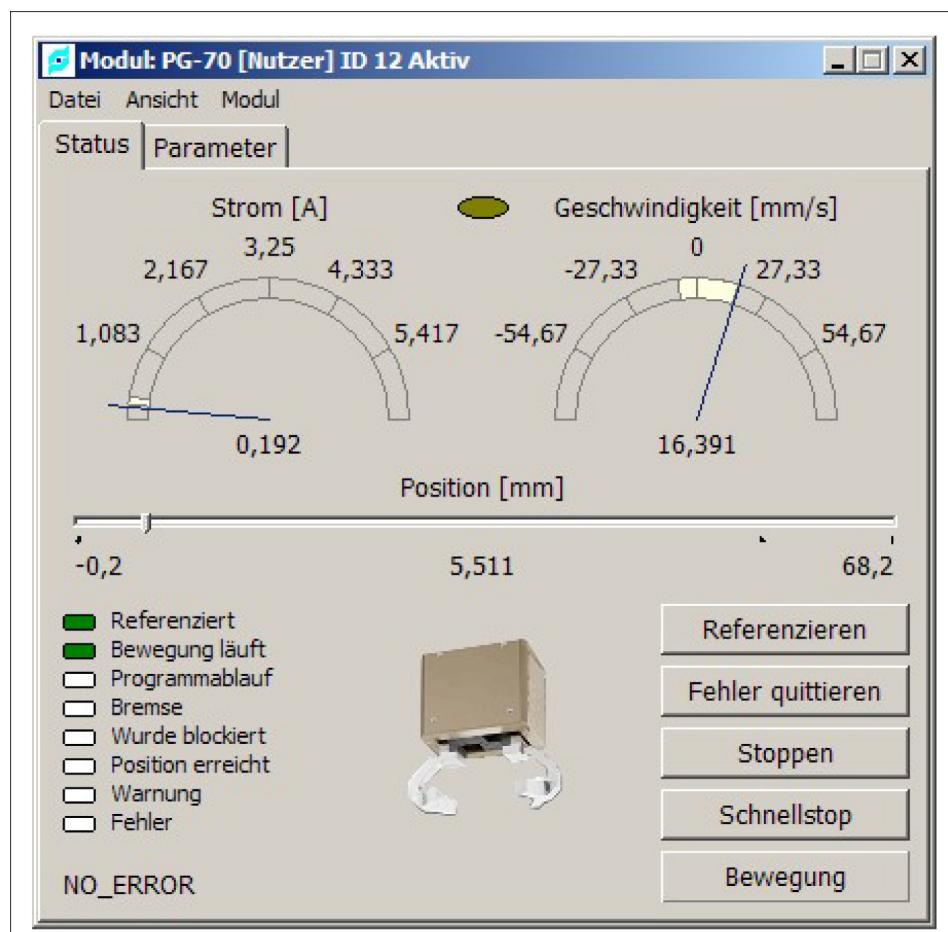


Abb. 15

4.3 Hauptfenster

Im *Hauptfenster* (Abb. ([4.2, Seite 152](#))) werden Verbindungen verwaltet, die Kommunikation auf dem aktiven Bussystem protokolliert und Anwendungseinstellungen bearbeitet.

Das Hauptfenster besteht aus:

- *Menüleiste*
- *Symbolleiste*
- *Ausgabefenster*
- *Statusleiste*

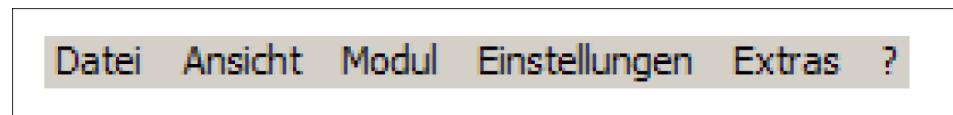


Abb. 16 Menüleiste



Abb. 17 Symbolleiste

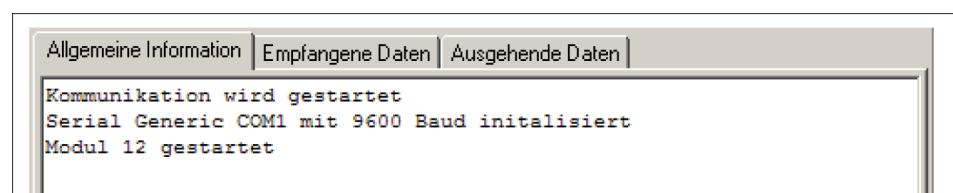


Abb. 18 Ausgabefenster



Abb. 19 Statusleiste

4.3.1 Symbolleiste

Mit Hilfe der Symbolleiste können die wichtigsten Funktionen schnell angesprochen werden.

	Der EEPROM Inhalt aller initialisierten Module wird abgespeichert.
	Der EEPROM Inhalt aller initialisierten Module wird ausgedruckt.
	Ein Modul mit bekannter ID kann initialisiert werden.
	Das gesamte Kommunikationsinterface wird nach aktiven Modulen durchsucht.
	An alle Module wird ein „CMD FAST STOP“ (☞ 2.2.19, Seite 53) gesendet.
	Das eingestellte Kommunikationsinterface wird geöffnet bzw. geschlossen.
	Es wird nach vorhandenen Kommunikationsinterfaces gesucht.
	Verschiedene <i>Grundeinstellungen</i> (☞ 4.3.4, Seite 160) können geändert werden.

4.3.2 Menü

- Datei
 - Laden: EEPROM-Konfiguration aus vorhandener Datei laden.
 - Speichern: Der EEPROM Inhalt aller initialisierten Module wird abgespeichert.
 - Drucken: Der EEPROM Inhalt aller initialisierten Module wird ausgedruckt.
 - Beenden: Die Anwendung wird geschlossen.
- Ansicht
 - Bus Details anzeigen: Detaillierte Meldung über den Zustand des Bussystems wird unter „Allgemeine Information“ angezeigt.
 - Lesezeichen setzen: In den Ausgabefenstern wird eine Markierung gesetzt.
 - Ausgabedaten löschen: Der Inhalt aller Ausgabefenster wird gelöscht.
 - Modul ... : Die Ansicht des ausgewählten Moduls wird aktiviert (Modulfenster).
- Modul
 - Initialisierung mit ID: Ein Modul mit bekannter ID wird initialisiert.
 - Deinitialisierung über ID: Ein bereits initialisiertes Modul wird deinitialisiert.
 - Bus durchsuchen: Der gesamte Bus wird nach aktiven Modulen durchsucht.
 - Diagnose finden: Nach Diagnose-Modulen suchen
 - Alle Schnellstop: An alle initialisierten Module wird ein *Schnellstop Kommando* ([☞ 2.2.19, Seite 53](#)) gesendet.
- Einstellungen
 - Verbindung herstellen/schließen: Das eingestellte Kommunikationsinterface wird geöffnet bzw. geschlossen.
 - Kommunikation öffnen: Es wird automatisch nach installierten Kommunikationsinterfaces gesucht, und eins davon manuell ausgewählt.
 - Sprache: Die aktuelle Sprache von "Motion Tool Schunk" kann umgestellt werden. Die Sprachumstellung gilt nach Neustart der Anwendung.
 - Einstellungen: Verschiedene *Grundeinstellungen* ([☞ 4.3.4, Seite 160](#)) können geändert werden.

- Extras
 - CRC: Ein *Tool* ([☞ 4.3.5, Seite 162](#)) zum Berechnen der CRC16-Prüfsumme.
 - Zahlenkonverter: Ein *Tool* ([☞ 4.3.5, Seite 162](#)), um verschiedene Zahlenformate ineinander umrechnen zu können. (*Daten Format* ([☞ 1.4.1, Seite 12](#)))
 - Programm Editor: Ein *Tool* ([☞ 4.3.5, Seite 162](#)) zur Erstellung von *Modul-Programmen* ([☞ 2.4.4, Seite 68](#)).
- ?
 - Motion Control: Diese Dokumentation.
 - Hilfe: "Motion Tool Schunk" Hilfe.
 - About: Hier stehen nähere Informationen über das Programm.

4.3.3 Ausgabefenster

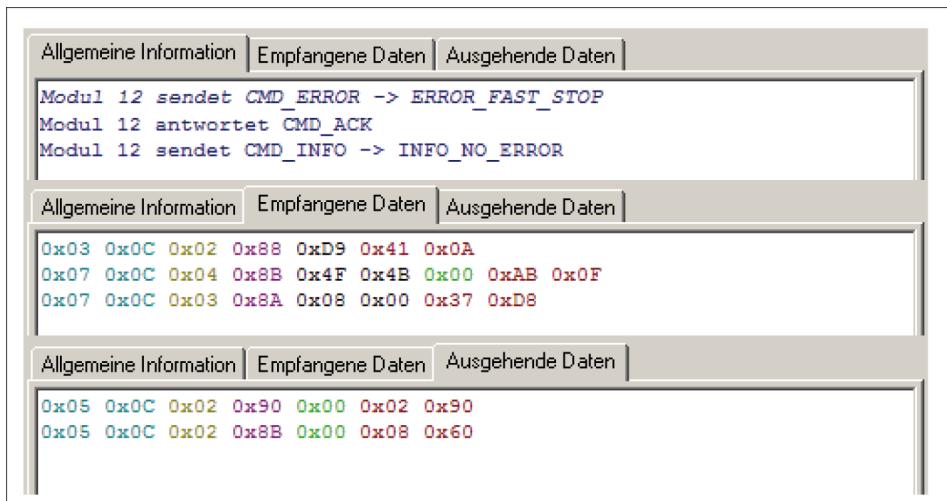


Abb. 20 Ausgabefenster

Im *Ausgabefenster* werden unter dem Reiter „*Allgemeine Information*“ alle Informationen bezüglich der einzelnen angeschlossenen Module im Klartext dargestellt. Jedes einzelne Modul wird hierbei farblich hervorgehoben. Unter dem Reiter „*Empfangene Daten*“ werden die auf dem jeweiligen Bussystem eingehenden Daten in Ihrer Rohform angezeigt. Der Reiter „*Ausgehende Daten*“ zeigt die versendeten Rohdaten an.

HINWEIS

„Empfangene“ bzw. „ausgehende“ Daten können eine große Hilfe beim Erstellen eigener Applikationen sein. Die Bytefolgen, welche auf dem jeweiligen Bussystem geschickt werden müssen, und die dazugehörige Antwort vom Modul, sind einfach abzulesen

Alle angezeigten Daten können in andere Anwendungen kopiert ("Strg-C, Strg-V") werden. Die einzelnen Protokollelemente ([☞ 1.4.2, Seite 15](#)) sind dabei farblich hervorgehoben.

Statusleiste

In der *Statusleiste* ([☞ 4.3, Seite 155](#)) werden die aktuellen Kommunikationseigenschaften angezeigt. Zustand und Einstellungen des Kommunikationsinterface, Anzahl und ID der initialisierten Module.

4.3.4 Grundeinstellungen

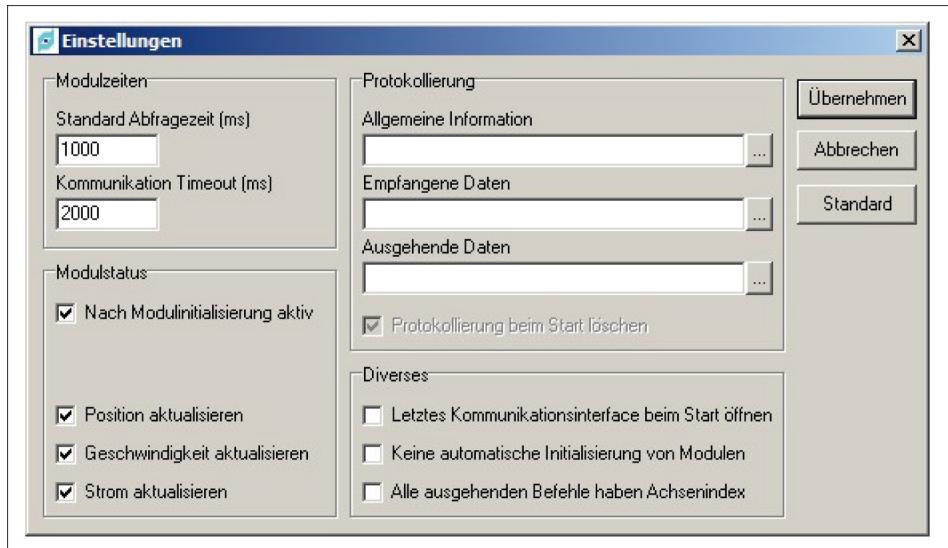


Abb. 21

- Modulzeiten
 - Standard Abfragezeit (ms): Beim Initialisieren eines Moduls wird bei gesetztem Hacken ein *GET STATE* ([☞ 2.6.1, Seite 84](#)) mit eingestellter Zyklus-Zeit abgesendet.
 - Kommunikation Timeout (ms): Zeit, die die jeweilige Schnittstelle mindestens auf eine Antwort vom Modul wartet. Beim nicht Empfangen der Antwort und nach Ablauf dieser Zeit wird ein „Timeout“ erzeugt.
- Modulstatus
 - Nach Modulinitialisierung aktiv: Bei gesetztem Hacken wird automatisch nach der Initialisierung eines Moduls ein *GET STATE* ([☞ 2.6.1, Seite 84](#)) abgesendet.
 - Position aktualisieren: Position wird mit *GET STATE* ([☞ 2.6.1, Seite 84](#)) angefordert.
 - Geschwindigkeit aktualisieren: Geschwindigkeit wird mit *GET STATE* ([☞ 2.6.1, Seite 84](#)) angefordert.
 - Strom aktualisieren: Strom wird mit *GET STATE* ([☞ 2.6.1, Seite 84](#)) angefordert.

- Protokollierung
 - Allgemeine Information: Name der Datei zum Speichern der Ausgaben unter „Allgemeine Information“.
 - Empfangene Daten: Name der Datei zum Speichern der Ausgaben unter „Empfangene Daten“.
 - Ausgehende Daten: Name der Datei zum Speichern der Ausgaben unter „Ausgehende Daten“.
 - Protokollierung beim Start löschen: Alle vorherigen Protokollierungen werden beim Neustart der Anwendung gelöscht.
- Diverses
 - Letztes Kommunikationsinterface beim Start öffnen: Zuletzt benutzte Kommunikationseinstellung wird beim Neustart eingestellt, und die Kommunikation wird gestartet.
 - Keine automatische Initialisierung von Modulen: Wenn ein Modul über die Bussuche gefunden wird, wird kein Modulfenster initialisiert. Eine Modulinitialisierung erfolgt ausschließlich manuell □.
 - Alle ausgehenden Befehle haben Achsenindex: Alle Befehle, die zum Modul versendet werden, haben immer den Achsenindex in der Nachricht. D.h., wenn kein Achsenindex definiert ist, wird „0“ mit versendet.

4.3.5 Tools

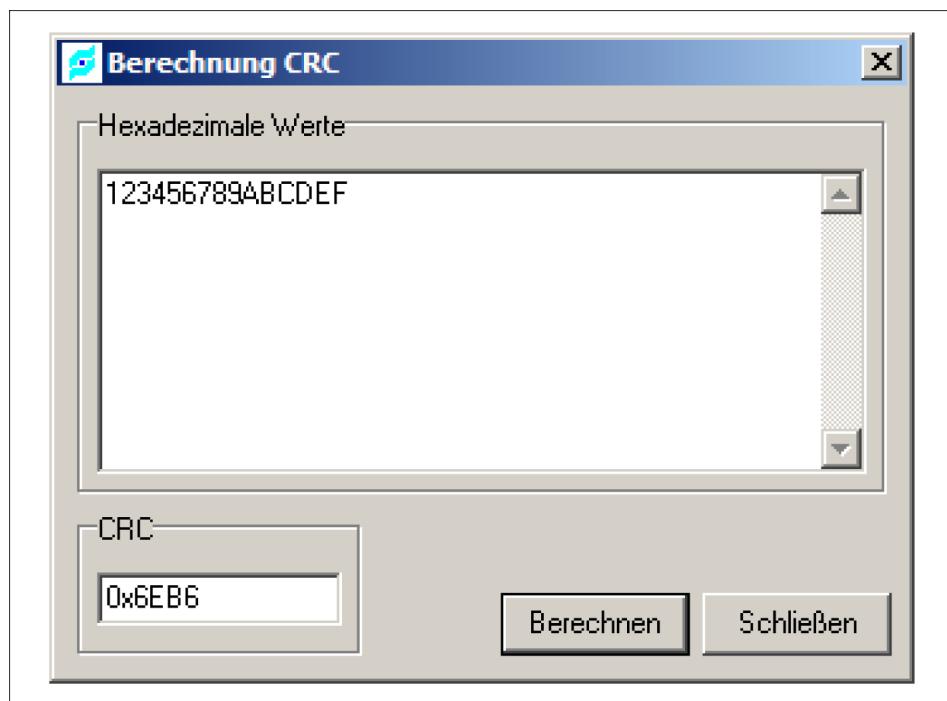


Abb. 22 Berechnung CRC



Abb. 23 Zahlenkonverter

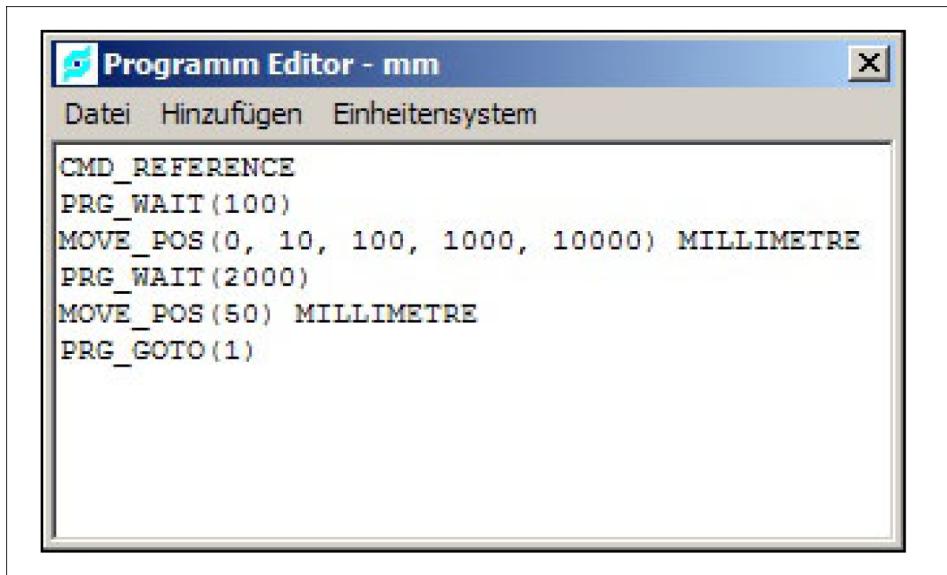


Abb. 24 Programm Editor

Mit dem Tool Berechnung CRC ([☞ 4.3.5, Seite 162](#)) wird die CRC16 ([☞ 6.2, Seite 196](#)) über zuvor eingegebene Hexadezimalzahlen berechnet.

Der Zahlenkonverter ([☞ 4.3.5, Seite 162](#)) gestattet es Zahlen in verschiedene Formate umzurechnen.

Mit dem Programm Editor gibt es eine Unterstützung zur internen Programmierung der Module.

4.4 Modulfenster

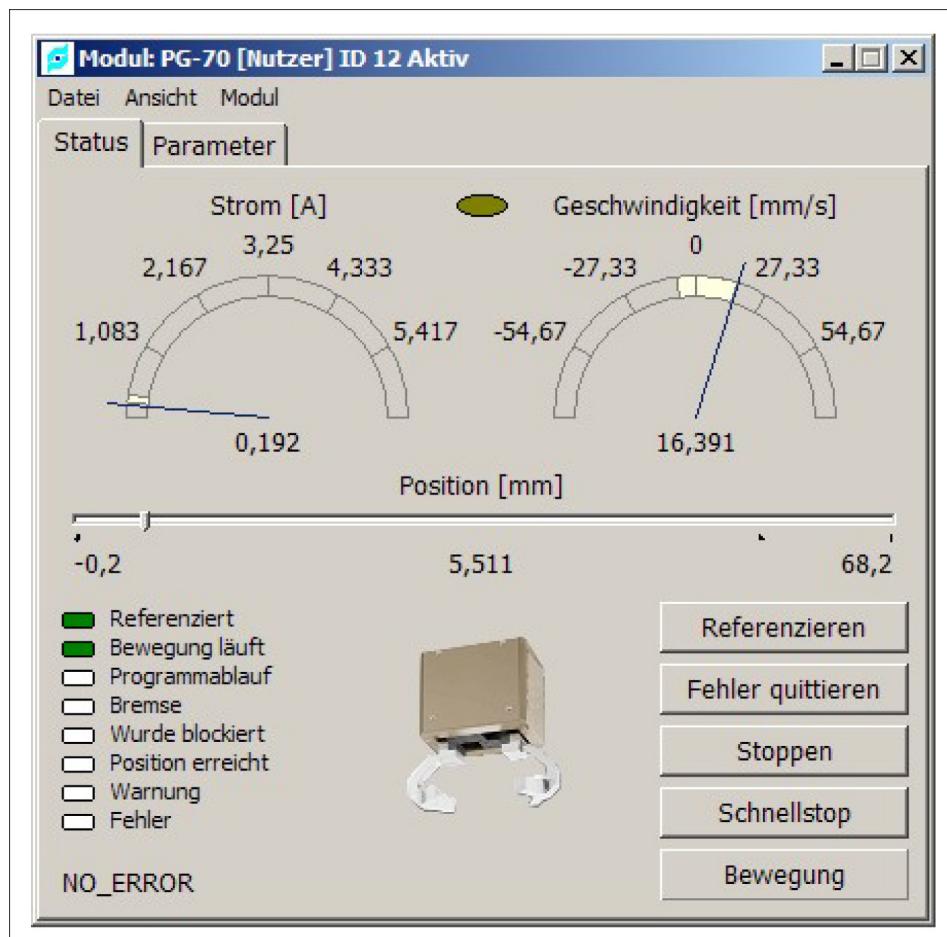


Abb. 25 Modulfenster

Im Modulfenster können alle modulrelevanten Daten verwaltet werden, sowie einzelne Befehle zu dem jeweiligen Modul gesendet werden. Unter dem Reiter „Status“ ist der aktuelle Status des Moduls zu sehen. Die Statusanzeige wird bei jeder eingehenden „*GET STATE*“ ([☞ 2.6.1, Seite 84](#)) Nachricht aktualisiert.

HINWEIS

Wird Statusabfrage abgeschaltet, bzw. ist unter Grundeinstellungen ([☞ 4.3.4, Seite 160](#)) dieser Befehl abgeschaltet, wird die Anzeige nicht aktualisiert. Bei der Kommunikation über Profibus werden die acht Indikatoren an der linken Seite aber mit jeder vom Modul angekommenen Nachricht aktualisiert.

Die Titelzeile wird in der Form „Modul: <Modultyp> [<Benutzerrechte>] ID <Modul-ID> <Status (aktiv/inaktiv)>“ angezeigt. Wenn ein zum Modul gesandter Befehl nicht beantwortet wird, ändert sich die Statusanzeige von „aktiv“ auf „inaktiv“.

Mit den Schaltern auf der rechten Seite können die wichtigsten Befehle direkt an das Modul versendet werden. Nach Aktivierung des Bereiches „Bewegung“ wird das Fenster um die notwendige *Parametrierung von Bewegungsbefehlen* erweitert.

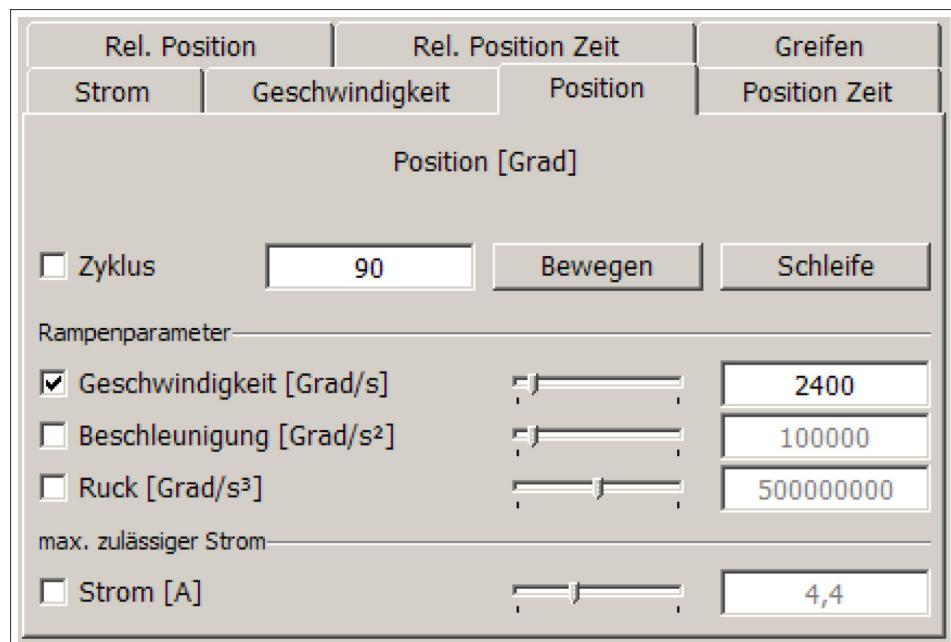


Abb. 26 Bewegungsbefehle

4.4.1 Modulzustände

Im Status können folgende Zustände angezeigt werden:

- Referenziert: Das Modul ist referenziert.
- Bewegung läuft: Das Modul bewegt sich mit einer Geschwindigkeit größer als die parametrierte *Bewegungsschwelle* ([☞ 3.2.6, Seite 146](#)).
- Programmablauf: Das interne *Programm wird ausgeführt* ([☞ 2.4.4, Seite 68](#)).
- Bremse: Die Bremse ist aktiviert.
- Wurde blockiert:
Der Zustand *wird aktiviert* ([☞ 2.3.2, Seite 55](#)), falls:
 - Das Modul steht oder sich mit einer Geschwindigkeit unterhalb der *Bewegungsschwelle* ([☞ 3.2.6, Seite 146](#)) bewegt, und
 - der Zielstrom erreicht worden ist ($\pm 15\%$), und
 - der *Bremsen-Timeout* ([☞ 3.2.7, Seite 150](#)) abgelaufen ist.
- Der Zustand wird deaktiviert, falls:
 - Ein neuer Verfahrbefehl ausgelöst wird, oder
 - der Zielstrom mit dem Befehl „*SET TARGET CURRENT*“ ([☞ 2.2.14, Seite 49](#)) neu gesetzt wird, oder
 - das Modul neu startet.
- Position erreicht: Die Zielposition ist während einer Positionsfahrt *erreicht* ([☞ 2.3.3, Seite 56](#)).
- Warnung: Es liegt eine *Warnung* ([☞ 2.8.1, Seite 106](#)) vor.
 - Fehler: Es liegt ein *Fehler* ([☞ 2.8.1, Seite 106](#)) vor.

4.4.2 Schalterelemente

Folgende Befehle stehen zur Verfügung:

- Referenzieren / Handreferenz: Eine *Referenzfahrt* ([☞ 2.2.1, Seite 32](#)) wird ausgeführt, bzw. manuelle Referenzierung wird gestartet.
- Fehler quittieren: Ein „CMD ACK“ ([☞ 2.8.1, Seite 106](#)) wird ausgeführt.
- Stoppen: Das Kommando *Stop* ([☞ 2.2.18, Seite 52](#)) wird ans Modul gesendet.
- Schnellstop: Ein *Schnellstop* ([☞ 2.2.19, Seite 53](#)) wird ausgelöst.
- Bewegung: *Das Fenster wird erweitert um weitere Bedienelemente* (Abb. ([☞ 4.4, Seite 164](#))) .

Bewegungsbefehle:

- Geschwindigkeit: Eine *Geschwindigkeitsfahrt* ([☞ 2.2.9, Seite 45](#)) wird durchgeführt.
- Position: Eine *Positionsfahrt* ([☞ 2.2.3, Seite 35](#)) wird durchgeführt. Wird *Zyklus* oder *Schleife* aktiviert, kann zwischen zwei frei wählbaren Positionen „hin und her“ gefahren werden.
- Position Zeit: Eine zeitgeregelte *Positionsfahrt* ([☞ 2.2.5, Seite 39](#)) (Kurvenbahnen möglich) wird durchgeführt. Wird *Zyklus* oder *Schleife* aktiviert, kann zwischen zwei frei wählbaren Positionen „hin und her“ gefahren werden.
- Rel. Position: Eine relative *Positionsfahrt* ([☞ 2.2.4, Seite 37](#)) wird durchgeführt. Wird *Zyklus* oder *Schleife* aktiviert, werden zwei relative Positionsfahrten wechselnd ausgeführt.
- Rel. Position Zeit: Eine zeitgeregelte relative *Positionsfahrt* ([☞ 2.2.6, Seite 41](#)) (Kurvenbahnen möglich) wird durchgeführt. Wird *Zyklus* oder *Schleife* aktiviert, werden zwei relative Positionsfahrten wechselnd ausgeführt.
- Greifen: Ein *Greifvorgang* ([☞ 2.2.10, Seite 46](#)) wird durchgeführt.

4.4.3 Modulparameter

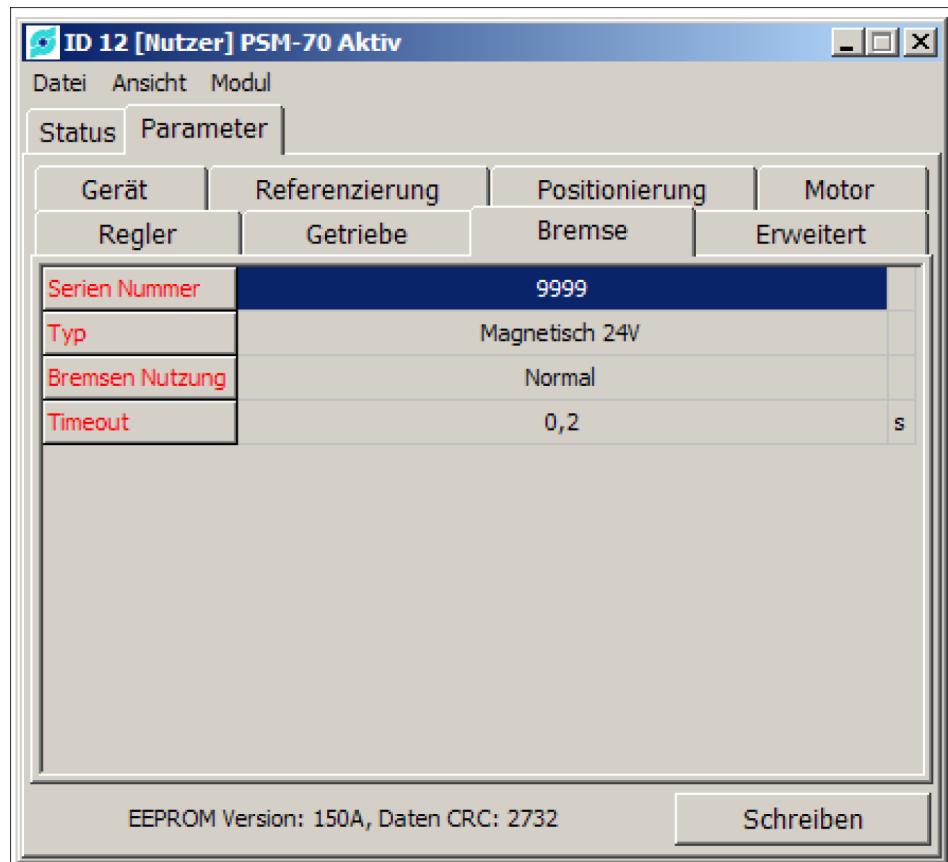


Abb. 27 EEPROM Bereich Bremse

Die Ansicht „Parameter“ ist vorhanden, sobald die Modulkonfiguration empfangen wird.

Unter den einzelnen „Reitern“ (Abb. [\(☞ 4.4.3, Seite 168\)](#)) der Ansicht „Parameter“ können die EEPROM Einträge ausgelesen und abgeändert werden. Je nach Nutzerrechten ([\(☞ 1.6, Seite 25\)](#)) sind einige Einträge nur zum Lesen freigegeben.

Folgende Reiter stehen zur Auswahl:

- Gerät ([3.2.5, Seite 136](#))
- Referenzierung ([3.2.3, Seite 125](#))
- Positionierung ([3.2.6, Seite 146](#))
- Motor ([3.2.1, Seite 119](#))
- Regler ([3.2.4, Seite 133](#))
- Getriebe ([3.2.2, Seite 124](#))
- Bremse ([3.2.7, Seite 150](#))
- Erweitert (Ab Firmware V1.50)

Im Reiter „Erweitert“ kann man erweiterte Parameter gezielt abfragen und unter Berücksichtigung der vorhandenen Rechte ändern:

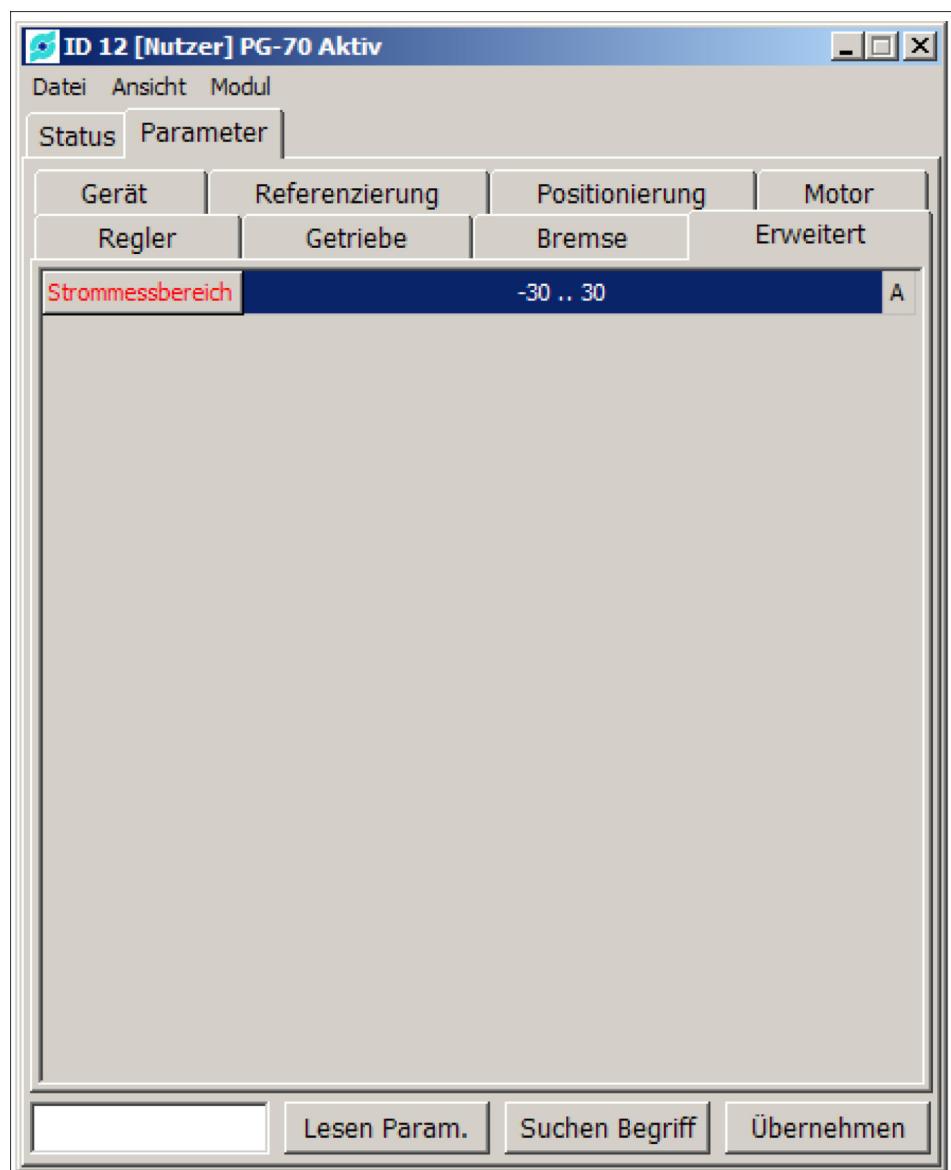


Abb. 28 Erweiterte Parametrierung

- *Lesen Param.* ([☞ 2.4.4, Seite 68](#)): Laden von Parametern anhand ihrer Parameter-ID.
- *Speichern Alle* ([☞ 2.4.4, Seite 68](#)): Alle angezeigten Parameter auf das Modul übertragen.
- *Änderungen* ([☞ 2.4.3, Seite 65](#)): Speichern der geänderten Parameter. Alle grün dargestellten Parameter sind editierbar. Alle rot dargestellten Parameter sind dagegen nicht editierbar.

4.4.4 Menü

Über das Menü kann man alle im Modul verfügbaren Befehle testen und einige weitere Einstellungen vornehmen. Hier nun die Beschreibung der einzelnen Menüeinträge:

- Datei
 - EEPROM laden: Konfigurationsparameter aus einer Datei laden.
 - EEPROM speichern: Die angezeigten Konfigurationsparameter werden in eine Datei gespeichert.
 - EEPROM ausdrucken: Die angezeigten Konfigurationsparameter werden gedruckt.
 - Änderungen ausdrucken: Nur geänderte (farblich hervorgehoben) Konfigurationsparameter werden gedruckt.
 - Deinitialisieren: Das Modulfenster wird geschlossen. An das entsprechende Modul wird dabei der Befehl „Schnellstop“ gesendet.
- Ansicht
 - Aktuelle Werte: Die zuletzt empfangene Information über den Modulzustand wird angezeigt.
 - Stromdiagramm: Der Stromverlauf wird dargestellt.
 - Geschwindigkeitsdiagramm: Der Geschwindigkeitsverlauf wird dargestellt.
 - Positionsdiagramm: Die Dynamik der Positionsänderung wird dargestellt.
 - Kombiniertes Diagramm: Die Dynamik von Strom, Geschwindigkeit und Position wird dargestellt.
 - Pause/Aktualisiere Diagramm: Aktualisierung des Diagramms kann angehalten bzw. wieder gestartet werden.
- Modul
 - Bewegung:
 - Stromfahrt: *Eine Stromfahrt wird ausgelöst* ([☞ 2.2.8, Seite 43](#)).
 - Geschwindigkeitsfahrt: *Eine Geschwindigkeitsfahrt wird ausgelöst* ([☞ 2.2.9, Seite 45](#)).
 - Positionsfahrt: *Eine Positionsfahrt wird ausgelöst* ([☞ 2.2.3, Seite 35](#)).
 - Positionsfahrt (Zeit): *Eine zeitgeregelte Positionsfahrt (Kurvenbahn möglich) wird ausgelöst* ([☞ 2.2.5, Seite 39](#)).

- Rel. Positionsfahrt: *Eine relative Positionsfahrt wird ausgelöst* ([☞ 2.2.4, Seite 37](#)).
 - Rel. Positionsfahrt (Zeit): *Eine zeitgeregelte relative Positionsfahrt (Kurvenbahn möglich) wird ausgelöst* ([☞ 2.2.6, Seite 41](#)).
 - Greifbewegung: *Eine Greiffahrt wird ausgelöst* ([☞ 2.2.10, Seite 46](#))
 - Sollstrom: *Der Sollstrom für die nächste(n) Bewegung(en) wird eingestellt* ([☞ 2.2.14, Seite 49](#)).
 - Sollbeschleunigung: *Die Sollbeschleunigung für die nächste(n) Bewegung(en) wird eingestellt* ([☞ 2.2.12, Seite 48](#)).
 - Sollgeschwindigkeit: *Die Sollgeschwindigkeit für die nächste(n) Bewegung(en) wird eingestellt* ([☞ 2.2.11, Seite 47](#)).
 - Sollruck: *Der „Sollruck“ für die nächste(n) Bewegung(en) wird eingestellt* ([☞ 2.2.13, Seite 48](#)).
 - Sollzeit: *Die Zeit für den nächsten Positionierbefehl wird eingestellt* ([☞ 2.2.15, Seite 50](#)).
 - Sollposition: *Die Position für den nächsten Positionierbefehl wird eingestellt* ([☞ 2.2.16, Seite 50](#)).
 - Rel. Sollposition: *Die relative Position für den nächsten Positionierbefehl wird eingestellt* ([☞ 2.2.17, Seite 51](#)).
- Programmierung
 - Programm zum Modul: *Ein Programm kann von einer Datei auf ein Modul übertragen werden* ([☞ 2.5.1, Seite 71](#)).
 - Programm vom Modul: *Ein im Modul gespeichertes Programm kann ausgelesen und in eine Datei gespeichert werden* ([☞ 2.5.2, Seite 72](#)).
 - Programm löschen: Der Programmspeicher wird geleert.
 - Programm anzeigen: *Ein im Modul vorhandenes Programm wird angezeigt* ([☞ 2.5.2, Seite 72](#)).
 - Programm ausführen: *Ein Programm wird ausgeführt* ([☞ 2.5.3, Seite 72](#)).
 - Satzauswahl: *Ein Programmsatz wird ausgeführt* ([☞ 2.5.4, Seite 73](#)).

- Satz 0 ausführen: *Programmsatz Nummer "0" wird ausgeführt* ([☞ 2.5.5, Seite 73](#)).
 - Satz 1 ausführen: *Programmsatz Nummer „1“ wird ausgeführt* ([☞ 2.5.6, Seite 74](#)).
 - Satz 2 ausführen: *Programmsatz Nummer „2“ wird ausgeführt* ([☞ 2.5.7, Seite 75](#)).
 - Satz 3 ausführen: *Programmsatz Nummer „3“ wird ausgeführt* ([☞ 2.5.8, Seite 75](#)).
 - Satz 4 ausführen: *Programmsatz Nummer „4“ wird ausgeführt* ([☞ 2.5.9, Seite 76](#)).
 - Satz 5 ausführen: *Programmsatz Nummer „5“ wird ausgeführt* ([☞ 2.5.10, Seite 77](#)).
 - Satz 6 ausführen: *Programmsatz Nummer „6“ wird ausgeführt* ([☞ 2.5.11, Seite 77](#)).
 - Satz 7 ausführen: *Programmsatz Nummer „7“ wird ausgeführt* ([☞ 2.5.12, Seite 78](#)).
 - Satz 8 ausführen: *Programmsatz Nummer „8“ wird ausgeführt* ([☞ 2.5.13, Seite 79](#)).
 - Satz 9 ausführen: *Programmsatz Nummer „9“ wird ausgeführt* ([☞ 2.5.14, Seite 79](#)).
 - Satz 10 ausführen: *Programmsatz Nummer „10“ wird ausgeführt* ([☞ 2.5.15, Seite 80](#)).
 - Satz 11 ausführen: *Programmsatz Nummer „11“ wird ausgeführt* ([☞ 2.5.16, Seite 80](#)).
 - Satz 12 ausführen: *Programmsatz Nummer „12“ wird ausgeführt* ([☞ 2.5.17, Seite 81](#)).
 - Satz 13 ausführen: *Programmsatz Nummer „13“ wird ausgeführt* ([☞ 2.5.18, Seite 81](#)).
 - Satz 14 ausführen: *Programmsatz Nummer „14“ wird ausgeführt* ([☞ 2.5.19, Seite 82](#)).
 - Satz 15 ausführen: *Programmsatz Nummer „15“ wird ausgeführt* ([☞ 2.5.20, Seite 82](#)).
- Kommunikationstests
 - Vom Modul: Alle Werte: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Vom Modul: -1.2345: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Vom Modul: 47.11: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).

- Vom Modul: 287454020: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Vom Modul: -1122868: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Vom Modul: 512: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Vom Modul: -20482: *Testdaten vom Modul zur Steuerung übertragen* ([☞ 2.6.11, Seite 100](#)).
 - Zum Modul: Alle Werte: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: -1.2345: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: 47.11: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: 287454020: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: -1122868: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: 512: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
 - Zum Modul: -20482: *Testdaten von der Steuerung zum Modul übertragen* ([☞ 2.6.12, Seite 102](#)).
- EEPROM lesen
 - Lesen EEPROM: *Die komplette Konfiguration wird vom Modul gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen ID: *Die Modul ID wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen Gruppe: *Die Gruppenkennung wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen Kommunikation: *Der aktuell eingestellte Kommunikationsmodus wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen Einheitensystem: *Das aktuell eingestellte Einheitensystem wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen Baudrate Seriell: *Die aktuell eingestellte Serielle Baudrate wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen Baudrate CAN: *Die aktuell eingestellte CAN Baudrate wird gelesen* ([☞ 2.4.2, Seite 62](#)).
 - Lesen max. Softwareendanschlag: *Der aktuell eingestellte obere Softwareendanschlag wird gelesen* ([☞ 2.4.2, Seite 62](#)).

- Lesen min. Softwareendanschlag: *Der aktuell eingestellte untere Softwareendanschlag wird gelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen nom. Strom: *Der eingestellte nominal Strom wird gelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen max. Strom: *Der eingestellte maximal Strom wird gelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen max. Geschwindigkeit: *Die maximal mögliche Geschwindigkeit wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen max. Beschleunigung: *Die maximal mögliche Beschleunigung wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen max. Ruck: *Der maximal mögliche Ruck wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Offset Phase A: *Der Offset der Stromphase A wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Offset Phase B: *Der Offset der Stromphase B wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Referenz Offset: *Der Offset der Referenzierung wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Serien Nummer: *Die Seriennummer des Geräts wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Daten CRC: *Die Prüfsumme der Konfigurationsparameter wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Bestellnummer: *Die Bestellnummer wird ausgelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Platine Version: *Die Version der Modulplatine wird gelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen Platine Datum: *Das Herstellungsdatum der Modulplatine wird gelesen* ([☞ 2.4.2, Seite 62](#)).
- Lesen erweiterten Parameter: *Erweiterte Parameter werden einzeln gelesen* ([☞ 2.4.4, Seite 68](#)).
- EEPROM schreiben
 - Schreiben EEPROM: Die komplette Konfiguration wird geschrieben ([☞ 2.4.1, Seite 59](#)). Es werden nur die für den aktiven Nutzer ([☞ 1.6, Seite 25](#)) freigegebenen Daten geschrieben.
 - Schreiben ID: Die Modul ID wird geändert ([☞ 2.4.1, Seite 59](#)). Ein Neustart ist erforderlich.
 - Schreiben Gruppe: Die Gruppen Nummer wird geändert ([☞ 2.4.1, Seite 59](#)). Ein Neustart ist erforderlich.

- Schreiben Kommunikation: Die Kommunikationsschnittstelle wird geändert ([☞ 2.4.1, Seite 59](#)). Ein Neustart ist erforderlich.
 - Schreiben Einheitensystem: Das Einheitensystem wird umgestellt ([☞ 2.4.1, Seite 59](#)). Ein Neustart ist erforderlich.
 - Schreiben Baudrate Seriell: Die Baudrate für die serielle Kommunikation wird umgestellt. ([☞ 2.4.1, Seite 59](#)) Ein Neustart ist erforderlich.
 - Schreiben Baudrate CAN: Die CAN Baudrate wird umgestellt ([☞ 2.4.1, Seite 59](#)). Ein Neustart ist erforderlich.
 - Schreiben erweiterten Parameter: Erweiterte Parameter werden einzeln geschrieben ([☞ 2.4.3, Seite 65](#)).
- Referenzieren / Nonius anfahren: Eine Referenzierfahrt wird ausgelöst, bzw. Nonius wird angefahren, wenn das Modul manuell referenziert ist. ([☞ 2.2.1, Seite 32](#))
 - Fehler quittieren: Ein Fehler wird quittiert ([☞ 2.8.1, Seite 106](#)).
 - Stoppen: Eine Bewegung wird angehalten ([☞ 2.2.18, Seite 52](#)).
 - Schnellstop: Ein Schnellstop wird ausgelöst ([☞ 2.2.19, Seite 53](#)).
 - Benutzer ändern: Der aktuelle Nutzer wird gewechselt ([☞ 2.6.10, Seite 99](#)). Kennwort erforderlich.
 - Status abfragen: Der aktuelle Status des Moduls wird angefordert ([☞ 2.6.1, Seite 84](#)).
 - Fehlerinformation: Erweiterte Information über den aktiven Fehler werden ausgelesen ([☞ 2.8.1, Seite 106](#)).
 - Modulinformation: Informationen (Best. Nr., Software-, Hardwareversion, Gerätetyp) werden ausgelesen ([☞ 2.4.2, Seite 62](#)).
 - Handreferenz: Das Modul wird manuell referenziert ([☞ 4.4.5, Seite 177](#)).
 - Digitale Ein-/Ausgänge: Die digitalen Ein-/Ausgänge werden ansteuert bzw. gelesen ([☞ 2.6.4, Seite 91](#)).
 - Neustart: Das Modul wird neu gestartet ([☞ 2.6.3, Seite 90](#)).
 - Abschalten: Das Modul wird vom Bussystem getrennt ([☞ 2.6.9, Seite 98](#)). Ein Neustart des Moduls ist erforderlich um das Modul weiter betreiben zu können.
 - Spontanmeldung ein/aus: Spontanmeldungen werden eingeschaltet bzw. ausgeschaltet ([☞ 2.3.7, Seite 58](#)).
 - Inbetriebnahme Assistent: Das Modul wird in Betrieb genommen ([☞ 4.4.6, Seite 177](#)).
 - Firmware aktualisieren: Eine neue Firmware kann auf das Modul geladen werden. Kennwort erforderlich.

4.4.5 Manuelle Referenzierung

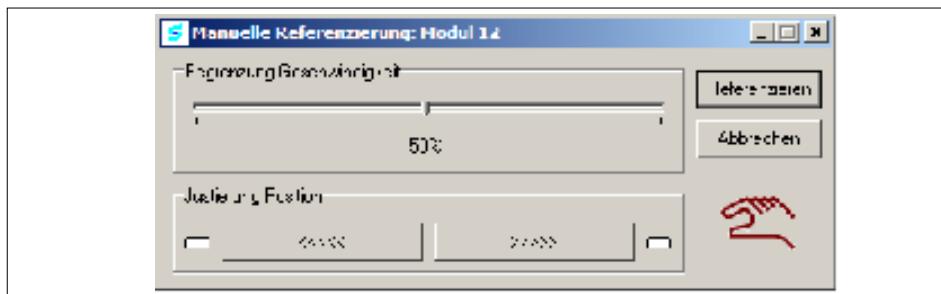


Abb. 29 Manuelle Referenzierung

Wenn das Modul in den manuellen Referenziermodus („*CMD REFERENCE HAND*“ ([☞ 2.2.2, Seite 33](#)) versetzt ist, ist es möglich, das Modul auf die gewünschte *Referenzposition* ([☞ 3.2.3, Seite 125](#)) zu justieren. Die höchste Geschwindigkeit während der Justagefahrt (100%) ist über den Parameter *maximale Referenzgeschwindigkeit* ([☞ 3.2.3, Seite 125](#)) einstellbar.

Bei erneutem Referenzieren eines Moduls mit Positionssystem ([☞ 3.2.6, Seite 146](#) „Encoder mit Indexspur“, das bereits manuell referenziert war, startet das Modul nach dem Verlassen dieser Funktion neu.

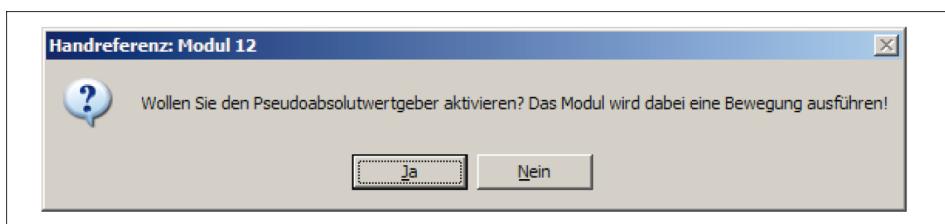


Abb. 30 Aktivierung Pseudoabsolutwertgeber

HINWEIS

Wurde der Pseudoabsolutwertgeber ([☞ 1.7, Seite 27](#)) nicht aktiviert, so ist das Modul nach dem Neustart nicht referenziert! Escheint die Dialogbox nicht, ist der Pseudoabsolutwertgeber bereits aktiviert, oder eine Aktivierung ist nicht möglich.

4.4.6 Inbetriebnahme Assistent

Beim Assistenten werden wichtige Konfigurationsparameter Schritt für Schritt angepasst und schließlich auf das Modul übertragen. Das Tool ist hauptsächlich für Inbetriebnahme von PSM und PDU Modulen konzipiert, die an einer Linearachse hängen.

4.5 Tipps

4.5.1 Unterstützte Sprachen

Derzeit werden Englisch und Deutsch unterstützt. Eine Sprachumstellung erfolgt über das Menü „Einstellungen/Sprache“. Die ausgewählte Sprache gilt nach Neustart der Anwendung.

4.5.2 Treiber Vector CAN

Falls die Anwendung die benötigte Bibliothekdatei „vcand32.dll“ nicht findet, ist zu beachten:

- Entweder die oben genannte DLL-Datei aus dem Installationsverzeichnis von Vector CAN ins Installationsverzeichnis von "Motion Tool Schunk" kopieren,
- oder das Verzeichnis, wo die DLL-Datei sich befindet, in die PATH-Variable von Windows hinzufügen,
- oder die DLL-Datei ins Windows-Systemverzeichnis „[Windows-Installation]/system32“ kopieren.

4.5.3 Treiber Peak CAN

Falls die Anwendung die benötigte Bibliothekdatei „pcan-usb.dll“ nicht findet, ist zu beachten:

- Entweder die oben genannte DLL-Datei von der mit dem Gerät mitgelieferten CD ins Installationsverzeichnis von "Motion Tool Schunk" kopieren,
- oder das Verzeichnis, wo die DLL-Datei sich befindet, in die PATH-Variable von Windows hinzufügen,
- oder die DLL-Datei ins Windows-Systemverzeichnis „[Windows-Installation]/system32“ kopieren.

4.5.4 Treiber Softing CAN

Falls die Anwendung die benötigte Bibliothekdatei „canL2.dll“ nicht findet, ist zu beachten:

- Entweder die oben genannte DLL-Datei von der mit dem Gerät mitgelieferten CD ins Installationsverzeichnis von "Motion Tool Schunk" kopieren,
- oder das Verzeichnis, wo die DLL-Datei sich befindet, in die PATH-Variable von Windows hinzufügen,
- oder die DLL-Datei ins Windows-Systemverzeichnis „[Windows-Installation]/system32“ kopieren.

4.5.5 Schnittstelle ESD CAN

Beim Durchsuchen der Schnittstelle auf aktive Module ohne dabei mindestens ein Modul eingeschaltet zu haben wechselt die Karte in den Bus-Off-Zustand und bleibt gewisse Zeit drin.

- Mindestens ein Modul einschalten, das für die CAN-Kommunikation bereit ist.
- Wenige Sekunden abwarten.
- Wenn kein Modul sich gemeldet hat, dann den Bus durchsuchen.

4.5.6 Schnittstelle Siemens Profibus

Die Kommunikation über Profibus-Karten von Siemens in "Motion Tool Schunk" ist relativ langsam.

Falls "Motion Tool Schunk" keine SMP-Module am Bus findet, obwohl welche existieren, müssen diese neu gestartet werden.

4.5.7 Modulstatus automatisch anzeigen lassen

Im Fenster „Einstellungen“ die Option „Nach Modulinitialisierung aktiv“ aktivieren und die Option „Standard Abfragezeit“ auf eine positive Zahl setzen (standardmäßig ca. 1000 ms).

4.5.8 Kommunikationsschnittstelle beim Start öffnen

Bei einer permanent vorhandenen Kommunikationsschnittstelle ist es empfehlenswert, die zuletzt verwendete Schnittstelle beim Starten der Anwendung automatisch öffnen lassen. Dazu im Fenster „Einstellungen“ die Option „Letztes Kommunikationsinterface verwenden“ aktivieren.

4.5.9 Datendurchsatz mit CAN

Das kürzeste Zeitintervall beim Kommando *GET STATE* ([☞ 2.6.1, Seite 84](#)) wird mit der CAN-Schnittstelle erreicht. Dieses ist bis zu 1 ms Zykluszeit möglich. Das heißt, die Anwendung kann jede Millisekunde ankommende Nachrichten anzeigen ohne einen Überlauf des internen Nachrichtenpuffers zu verursachen.

Das kürzeste Zeitintervall beim Kommando *GET STATE* GET STATE wird mit der CAN-Schnittstelle erreicht. Dieses ist bis zu 1 ms Zykluszeit möglich. Das heißt, die Anwendung kann jede Millisekunde ankommende Nachrichten anzeigen ohne einen Überlauf des internen Nachrichtenpuffers zu verursachen.

4.5.10 Konfigurierte Module unter Profibus

Um eine Liste der für Profibus konfigurierten Module anzusehen kann man im Hauptfenster im Menü „Ansicht“ den Eintrag „Bus Details anzeigen“ ausführen.

4.5.11 Ötere Timeouts bei der Seriellen Kommunikation

Da die Serielle Schnittstelle kein Bussystem ist, kann es sporadisch zu Datenkollisionen kommen, falls gleichzeitig mehrere Module kommunizieren. Die Anwendung versucht stets diese Datenkollisionen zu behandeln (Befehle werden wiederholt), damit die Kommunikation ohne Unterbrechung weiter läuft.

Die Wahrscheinlichkeit, dass Timeouts auftreten (also, keine Antwort vom Modul), kann man verringern durch:

- Das Zeitintervall des Befehls *GET STATE* ([☞ 2.6.1, Seite 84](#)) vergrößern oder bei Nichtgebrauch der Statusinformationen diese abzuschalten,
- oder/und die Baudrate der Seriellen Schnittstelle erhöhen,
- oder/und das Kommunikation-Timeout unter "Einstellungen" im Hauptfenster etwas erhöhen,
- oder/und möglichst weniger Module initialisieren bzw. gleichzeitig ansteuern.
- Das Zeitintervall des Befehls *GET STATE* *GET STATE* vergrößern oder bei Nichtgebrauch der Statusinformationen diese abzuschalten,
- oder/und die Baudrate der Seriellen Schnittstelle erhöhen,
- oder/und das Kommunikation-Timeout unter "Einstellungen" im Hauptfenster etwas erhöhen,
- oder/und möglichst weniger Module initialisieren bzw. gleichzeitig ansteuern.

4.5.12 Einzelne EEPROM-Parameter ändern

Falls nur einzelne EEPROM-Parameter wie die Modul-ID oder die Kommunikationsschnittstelle zu ändern sind, ist es empfehlenswert die entsprechenden *SET CONFIG* ([☞ 2.4.1, Seite 59](#)) Befehle auszuführen. Diese sind im Modulfenster unter dem Menü „Modul/EEPROM schreiben/Schreiben ...“ zu finden.

4.5.13 Hauptfenster nicht maximieren

Das Hauptfenster sollte nicht maximiert werden, da bereits initialisierte Modulfenster verdeckt bleiben. Ein verdecktes Modulfenster kann man jedoch mit dem entsprechenden Menüpunkt „Ansicht/Modul ...“ wieder finden.

4.5.14 Kommunikationsart „Auto“ beim Modul

Wenn bei einem Modul die Kommunikationsart auf „Auto“ eingestellt ist, sollte man folgende Punkte beachten:

- Nach dem Einschalten meldet sich das Modul erst dann, wenn es eine Kommunikationsschnittstelle als aktiv erkennt.
- Profibus wird unter Umständen sofort als aktiv erkannt (ein aktiver Host vorausgesetzt).
- Sicherstellen, dass nur eine einzige physikalische Schnittstelle für die Kommunikation vorhanden ist, sonst würde das Modul unter Umständen eine „falsche“ Kommunikation wählen.

4.5.15 Keine automatische Initialisierung

Die Option „Keine automatische Initialisierung von Modulen“ unter *Grundeinstellungen* ([☞ 4.3.4, Seite 160](#)) der Anwendung dient dazu, damit gefundene Module nicht automatisch initialisiert werden. Dabei werden keine Modulfenster automatisch geöffnet.

Die Einstellung ist besonders dann nützlich, wenn viele mit der seriellen Schnittstelle kommunizierenden Module zu gleicher Zeit zwangsweise initialisiert werden, was dazu führen kann, dass nicht alle Module vollständig initialisiert werden können.

5 Bemerkungen

5.1 Module

Anschlussbeschreibung zum Modul	Da verschiedenste Schunk Module mit verschiedenen Anschlussbelegungen existieren, bitte hierfür in der mitgelieferten Dokumentation zu Ihrem Modul nachsehen.
Modul kann nicht aus jeder Lage referenzieren	Der maximal zugelassene Referenzstrom reicht nicht aus um das Modul zu bewegen. Den <i>max. Referenzstrom</i> (☞ 3.2.3, Seite 125) erhöhen.

5.2 Protokoll

Fragmentierung ist nicht möglich	Das komplette Modul kann vollständig bedient werden, ohne das eine Fragmentierung der Daten notwendig ist. Dies gilt für alle unterstützten Kommunikationsschnittstellen.
---	---

5.3 Serielle Kommunikation

Es kommt zu Datenkollisionen	<ul style="list-style-type: none"> • Weniger Module an den „seriellen Bus“ anschließen • Spontanmeldungen abschalten <i>CMD TOGGLE IMPULSE MESSAGE</i> (☞ 2.3.7, Seite 58) • Schnittstelle „Serielle Kommunikation ohne Spontanmeldung“ nutzen.
Beim Zusammenschließen mehrerer Module kommt es zu Problemen	Die serielle Schnittstelle ist nicht als Bussystem konzipiert. Wenn Sie mehrere Module mit einer Schnittstelle betreiben wollen, sollten Sie auf ein „echtes“ Bussystem (CAN, Profibus) wechseln. Sind zu viele Teilnehmer angeschlossen kann es gehäuft zu <i>Datenkollisionen</i> (☞ 5.3, Seite 183) kommen. Wird die serielle Schnittstelle als Bussystem genutzt werden, nur Baudaten bis 19200 [kBaud] unterstützt.
Welche Baudaten werden vom Modul bei der seriellen Kommunikation unterstützt?	1200, 2400, 4800, 9600, 19200, 38400 [kBaud].

5.4 CAN

Welche CAN-Baudaten werden vom Modul unterstützt?	50, 100, 125, 250, 500 und 1000 [kBit/s]
--	--

5.5 Profibus

Wird SSA (Set-Slave-Address) unterstützt?	Ja
Daten können nicht konsistent übertragen werden	<ol style="list-style-type: none">1 D-Len (erstes Byte) auf 0x00 setzen, alle Daten auffüllen und zuletzt D-Len auf gewünschten Wert setzen.2 SYNC, FREEZE Mechanismus verwenden.

6 Anhang

6.1 Ausgesuchte Beispiele

6.1.1 Serielle Kommunikation

6.1.1.1 CHECK MC PC COMMUNICATION (Float)

Kommunikation Modul zur Steuerung mit Testdaten prüfen.

- 1 *Testdaten* ([2.6.11, Seite 100](#)) vom Modul anfordern
(Float Wert -1.2345).
- 2 Testdaten werden vom Modul gesendet.

	Id	D-Len	Cmd	Daten	CRC16
M->S	0x05 0x01	0x03	0xE4	0x01 0x01	0xBD 0xB6
S->M	0x07 0x01	0x07	0xE4	0x19 0x04 0x9E 0xBF 0x01 0x01	0x74 0x37

6.1.1.2 CHECK PC MC COMMUNICATION

Kommunikation Steuerung zum Modul mit Testdaten prüfen.

- 1 *Testdaten* ([2.6.11, Seite 100](#)) zum Modul senden.
- 2 Modul meldet Testdaten empfangen und Information, welche Daten fehlerhaft interpretiert wurden. (alle Daten in Ordnung)

	Id	D-Len	Cmd	Daten	CRC16
M->S	0x05 0x01	0x15	0xE5	0x19 0x04 0x9E 0xBF 0xA4 0x70 0x3C 0x42 0x44 0x33 0x22 0x11 0xCC 0xDD 0xEE 0xFF 0x00 0x02 0xFE 0xAF	0x89 0xD7
S->M	0x07 0x01	0x04	0xE5	0x4F 0x4B 0x00	0xB6 0xFA

6.1.1.3 Fehler Behebung

Ein Fehler („*ERROR MOTOR VOLTAGE LOW*“) ([☞ 2.8.2, Seite 109](#)) ist aufgetreten.

- 1 Fehler interpretieren und beheben. Spannungsversorgung zum Motor abgeschaltet => Spannungsversorgung wieder einschalten
- 2 Fehler quittieren „*CMD ACK*“ ([☞ 2.8.1, Seite 106](#))
- 3 Kommando „*CMD ACK*“ wird bestätigt
- 4 Info Meldung erscheint „Keine weiteren Fehler vorhanden.“

	Id	D-Len	Cmd	Daten	CRC16
alle 15 [s]	0x03 0x01	0x02	0x88	0x74	0x82 0x1B
M->S	0x05 0x01	0x01	0x8B		0x10 0xFB
S->M	0x07 0x01	0x03	0x8B	0x4F 0x4B	0x38 0x1E
Spon- tan	0x07 0x01	0x03	0x8A	0x08 0x00	0x1A 0x19

6.1.1.4 GET STATE 1 [s]

Jede Sekunde soll der aktuelle Positionswert übertragen werden.

- 1 Kommando *Get State* ([☞ 2.6.1, Seite 84](#)) 1[s] nur Position absenden.
- 2 Eine Meldung der Position mit aktuellem Status.
(Position 1.011[mm], in Bewegung, kein Fehler)
- 3 Zyklisch jede Sekunde weitere Meldungen.
- 4 Meldung der Position mit aktuellem Status.
(Position 5.054[mm], in Bewegung, kein Fehler)

	Id	D-Len	Cmd	Daten	CRC16
M->S	0x05 0x01	0x06	0x95	0x00 0x00 0x80 0x3F 0x01	0x54 0x41
S->M	0x07 0x01	0x07	0x95	0x36 0x89 0x81 0x3F 0x02 0x00	0xF9 0xBC
S->M	0x07 0x01	0x07	0x95
S->M	0x07 0x01	0x07	0x95	0x76 0xBE 0xA1 0x40 0x02 0x00	0x38 0xA0

6.1.1.5 MOVE POS 10 [mm]

Der Standard Befehl zum Positionieren.

- 1 Kommando fahre an *Position* ([2.2.3, Seite 35](#)) 10 [mm] absenden
- 2 Meldung „Werde Position in 3.358 [s] erreichen“. Bewegung wird gestartet.
- 3 Nach einiger Zeit „Habe Position 9.9969 [mm] erreicht“

	Id	D-Len	Cmd	Daten	CRC16
M->S	0x05 0x01	0x05	0xB0	0x00 0x00 0x20 0x41	0x48 0x80
S->M	0x07 0x01	0x05	0xB0	0xEE 0xEE 0x56 0x40	0x7B 0xE4
Spon- tan	0x07 0x01	0x05	0x94	0xB6 0xF3 0x1F 0x41	0x7E 0xD5

6.1.1.6 Referenzieren

- 1 Kommando *Referenzieren* ([2.2.1, Seite 32](#)) absenden
- 2 Habe Referenzierkommando verstanden
- 3 Nach einiger Zeit „Stehe an Position 5.792 [mm]“

	Id	D-Len	Cmd	Daten	CRC16
M->S	0x05 0x01	0x01	0x92		0xD1 0x31
S->M	0x07 0x01	0x03	0x92	0x4F 0x4B	0xE9 0xD9
Spon- tan	0x07 0x01	0x05	0x93	0x21 0x56 0xB9 0x40	0x4D 0x22

6.1.2 CAN

6.1.2.1 CHECK MC PC COMMUNICATION (Float)

Kommunikation Modul zur Steuerung mit *Testdaten* ([☞ 2.6.11, Seite 100](#)) prüfen.

- 1 *Testdaten* ([☞ 2.6.11, Seite 100](#)) vom Modul anfordern (Float Wert -1.2345).

⇒ Testdaten werden vom Modul gesendet.

	Id	D-Len	Cmd	Daten	
M->S	0x501	0x04	0x03	0xE4	0x01 0x01
S->M	0x701	0x08	0x07	0xE4	0x19 0x04 0x9E 0xBF 0x01 0x01

6.1.2.2 CHECK PC MC COMMUNICATION

Kommunikation Steuerung zum Modul mit *Testdaten* ([☞ 2.6.11, Seite 100](#)) prüfen. Eine Fragmentierung ist notwendig.

HINWEIS

Eine Fragmentierung der Daten ist nicht zwingend notwendig um das Modul zu betreiben und/oder zu prüfen

- 1 Erstes Fragment der Testdaten zum Modul senden.
 - 2 Zweites Fragment der Testdaten zum Modul senden.
 - 3 Drittes Fragment der Testdaten zum Modul senden.
 - 4 Letztes Fragment der Testdaten zum Modul senden.
- ⇒ Modul meldet Testdaten empfangen und Information, welche Daten fehlerhaft interpretiert wurden (alle Daten in Ordnung).

	Id	DLC	D-Len	Cmd	Daten
M->S	0x501	0x08	0x15	0x84 0xE5	0x19 0x04 0x9B 0xBF 0xA4
M->S	0x501	0x08	0x0F	0x85	0x70 0x3C 0x42 0x44 0x33 0x22
M->S	0x501	0x08	0x09	0x85	0x11 0xCC 0xDD 0xEE 0xFF 0x00
M->S	0x501	0x05	0x03	0x86	0x02 0xFE 0xAF
S->M	0x701	0x05	0xE4	0x4F 0x4B 0x00	

6.1.2.3 Fehler Behebung

Ein Fehler („*ERROR MOTOR VOLTAGE LOW*“) ([☞ 2.8.2, Seite 109](#)) ist aufgetreten.

Fehler interpretieren und beheben. Spannungsversorgung zum Motor abgeschaltet => Spannungsversorgung wieder einschalten.

- 1 Fehler quittieren „CMD ACK“ ([☞ 2.8.1, Seite 106](#))
- 2 Kommando „CMD ACK“ wird bestätigt
⇒ Info Meldung erscheint „Keine weiteren Fehler vorhanden.“

	Id	DLC	D-Len	Cmd	Daten
alle 15[s]	0x301	0x03	0x02	0x88	0x74
M->S	0x501	0x02	0x01	0x8B	
S->M	0x701	0x04	0x03	0x8B	0x4F 0x4B
Spontan	0x701	0x04	0x03	0x8A	0x08 0x00

6.1.2.4 GET STATE 1 [s]

Jede Sekunde soll der aktuelle Positionswert übertragen werden.

- 1 Kommando Get State ([☞ 2.6.1, Seite 84](#)) 1[s] nur Position absenden.
- 2 Eine Meldung der Position mit aktuellem Status. (Position 1.011[mm], in Bewegung, kein Fehler)
- 3 Zyklisch jede Sekunde weitere Meldungen.
- 4 Meldung der Position mit aktuellem Status. (Position 5.054[mm], in Bewegung, kein Fehler)

	Id	DLC	D-Len	Cmd	Daten
M->S	0x501	0x07	0x06	0x95	0x00 0x00 0x80 0x3F 0x01
S->M	0x701	0x08	0x07	0x95	0x36 0x89 0x81 0x3F 0x02 0x00
S->M	0x701	0x08	0x07	0x95
S->M	0x701	0x08	0x05	0x94	0x76 0xBE 0xA1 0x40 0x02 0x00

6.1.2.5 MOVE POS 10 [mm]

Der Standard Befehl zum *Positionieren* ([☞ 2.2.3, Seite 35](#)).

- 1 Kommando fahre an Position 10[mm] absenden
- 2 Meldung „Werde Position in 3.358[s] erreichen“. Bewegung wird gestartet.
- 3 Nach einiger Zeit „Habe Position 9.9969[mm] erreicht“

	Id	DLC	D-Len	Cmd	Daten
M->S	0x501	0x06	0x05	0xB0	0x00 0x00 0x20 0x41
S->M	0x701	0x06	0x05	0xB0	0xEE 0xEE 0x56 0x40
Spontan	0x701	0x06	0x05	0x94	0xB6 0xF3 0x1F 0x41

6.1.2.6 Referenzieren

- 1 Kommando *Referenzieren* ([☞ 2.2.1, Seite 32](#)) absenden
- 2 Habe Referenzierkommando verstanden
- 3 Nach einiger Zeit „Stehe an Position 5.792[mm]“

	Id	DLC	D-Len	Cmd	Daten
M->S	0x501	0x02	0x01	0x92	
S->M	0x701	0x04	0x03	0x92	0x4F 0x4B
Spontan	0x701	0x06	0x05	0x93	0x21 0x56 0xB9 0x40

6.1.3 Profibus

6.1.3.1 CHECK MC PC COMMUNICATION (Float)

Kommunikation Modul zur Steuerung mit *Testdaten* ([☞ 2.6.11, Seite 100](#)) prüfen.

- 1 *Testdaten* ([☞ 2.6.11, Seite 100](#)) vom Modul anfordern (Float Wert -1.2345).
- 2 Testdaten werden vom Modul gesendet. Der „*MsgCount*“ ([☞ 1.4.5, Seite 19](#)) wird um eins erhöht.

	D-Len	Cmd	Daten	State/ MsgCount
M->S	0x03	0xE4	0x01 0x01 0x?? 0x?? 0x?? 0x??	
S->M	0x07	0xE4	0x19 0x04 0x9E 0xBF 0x01 0x01 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x05

6.1.3.2 CHECK PC MC COMMUNICATION

Kommunikation Steuerung zum Modul mit *Testdaten* ([☞ 2.6.11, Seite 100](#)) prüfen. Eine Fragmentierung ist notwendig.

HINWEIS

Eine Fragmentierung der Daten ist nicht zwingend notwendig um das Modul zu betreiben und/oder zu prüfen.

- 1 Erstes Fragment der Testdaten zum Modul senden.
 - 2 Auf *Bestätigung* ([☞ 2.7.1, Seite 104](#)) warten
 - 3 Zweites Fragment der Testdaten zum Modul senden
 - 4 Auf *Bestätigung* ([☞ 2.7.1, Seite 104](#)) warten
 - 5 Drittes Fragment der Testdaten zum Modul senden
 - 6 Auf *Bestätigung* ([☞ 2.7.1, Seite 104](#)) warten.
 - 7 Letztes Fragment der Testdaten zum Modul senden.
- ⇒ Modul meldet Testdaten empfangen und Information, welche Daten fehlerhaft interpretiert wurden (alle Daten in Ordnung). Der „*MsgCount*“ ([☞ 1.4.5, Seite 19](#)) wird um eins erhöht.

	D-Len	Cmd	Daten	State/ MsgCount
M->S	0x15	0x84	0xE5 0x19 0x04 0x9E 0xBF 0xA4	
S->M	0x02	0x87	0x15 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x06
M->S	0x0F	0x85	0x70 0x3C 0x42 0x44 0x33 0x22	
S->M	0x02	0x87	0x0F 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x06
M->S	0x09	0x85	0x11 0xCC 0xDD 0xEE 0xFF 0x00	
S->M	0x02	0x87	0x09 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x06
M->S	0x03	0x86	0x02 0xFE 0xAF 0x?? 0x?? 0x??	
S->M	0x04	0xE5	0x4F 0x4B 0x00 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x06

6.1.3.3 Fehler Behebung

Ein Fehler („*ERROR MOTOR VOLTAGE LOW*“) ([☞ 2.8.2, Seite 109](#)) ist aufgetreten.

- 1 Fehler interpretieren (erweiterte Diagnose wird unterstützt) und beheben. Spannungsversorgung zum Motor abschalten => Spannungsversorgung wieder einschalten.
 - 2 Fehler quittieren „CMD ACK“ ([☞ 2.8.1, Seite 106](#)).
 - 3 Kommando „CMD ACK“ wird bestätigt. Der „MsgCount“ ([☞ 1.4.5, Seite 19](#)) wird um eins erhöht.
- ⇒ Info Meldung erscheint „Keine weiteren Fehler vorhanden.“

	D-Len	Cmd	Daten	State/ MsgCount
S->M	0x02	0x88	0x74 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x30 0x03
M->S	0x01	0x8B	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	
S->M	0x03	0x8B	0x4F 0x4B 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x04
Spon- tan	0x03	0x8A	0x08 0x00 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x20 0x04

6.1.3.4 GET STATE 1 [s]

Jede Sekunde soll der aktuelle Positions Wert übertragen werden.

- 1 Kommando *Get State* ([☞ 2.6.1, Seite 84](#)) 1 [s] nur Position absenden.
- 2 Eine Meldung der Position mit aktuellem Status. (Position 1.011 [mm], in Bewegung, kein Fehler) Der „*MsgCount*“ ([☞ 1.4.5, Seite 19](#)) wird um eins erhöht.
- 3 Zyklisch jede Sekunde weitere Meldungen.
- 4 Meldung der Position mit aktuellem Status. (Position 5.054 [mm], in Bewegung, kein Fehler)

	D-Len	Cmd	Daten	State/ MsgCount
M->S	0x07	0x95	0x00 0x00 0x80 0x3F 0x01 0x??	
S->M	0x08	0x95	0x36 0x89 0x81 0x3F 0x02 0x00 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x02 0x03
S->M	0x08	0x95	0x02 0x03
S->M	0x08	0x95	0x76 0xBE 0xA1 0x40 0x02 0x00 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x02 0x03

6.1.3.5 MOVE POS 10 [mm]

Der Standard Befehl zum *Positionieren* ([☞ 2.2.3, Seite 35](#)).

- 1 Kommando fahre an Position 10[mm] absenden
- 2 Meldung „Werde Position in 3.358 [s] erreichen“. Bewegung wird gestartet. Der „*MsgCount*“ ([☞ 1.4.5, Seite 19](#)) wird um eins erhöht.
- 3 'Nach einiger Zeit „Habe Position 9.9969 [mm]“ erreicht'

	D-Len	Cmd	Daten	State/ MsgCount
M->S	0x05	0xB0	0x00 0x00 0x20 0x41 0x?? 0x??	
S->M	0x05	0xB0	0xEE 0xEE 0x56 0x40 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x01 0x02
Spon- tan	0x05	0x94	0xB6 0xF3 0x1F 0x41 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x61 0x02

6.1.3.6 Referenzieren

- 1 Kommando *Referenzieren* ([☞ 2.2.1, Seite 32](#)) absenden
- 2 Habe Referenzierkommando verstanden. Der „*MsgCount*“ 15 wird um eins erhöht.
- 3 Nach einiger Zeit „Stehe an Position 5.792[mm]“

	D-Len	Cmd	Daten	State/ MsgCount
M->S	0x01	0x92	0x?? 0x?? 0x?? 0x?? 0x?? 0x??	
S->M	0x03	0x92	0x4F 0x4B 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x?? 0x??	0x00 0x01
Spon- tan	0x05	0x93	0x21 0x56 0xB9 0x40	0x61 0x01

6.2 CRC16 Berechnung für Serielle Kommunikation

Der hier gezeigte CRC16 Algorithmus ist so in der Firmware vorhanden. Im Netz lassen sich weiter Algorithmen finden, welche ebenfalls eine CRC16 Berechnung durchführen und ohne Tabelle auskommen.

```
UInt16 CRC16(UInt16 crc, UInt8 data)
{
const UInt16 tbl[256] = {
0x0000, 0xC0C1, 0xC181, 0x0140, 0xC301, 0x03C0, 0x0280, 0xC241,
0xC601, 0x06C0, 0x0780, 0xC741, 0x0500, 0xC5C1, 0xC481, 0x0440,
0xCC01, 0x0CC0, 0x0D80, 0xCD41, 0x0F00, 0xCFC1, 0xCE81, 0x0E40,
0x0A00, 0xCAC1, 0xCB81, 0x0B40, 0xC901, 0x09C0, 0x0880, 0xC841,
0xD801, 0x18C0, 0x1980, 0xD941, 0x1B00, 0DBC1, 0xDA81, 0x1A40,
0x1E00, 0xDEC1, 0xDF81, 0x1F40, 0xDD01, 0x1DC0, 0x1C80, 0xDC41,
0x1400, 0xD4C1, 0xD581, 0x1540, 0xD701, 0x17C0, 0x1680, 0xD641,
0xD201, 0x12C0, 0x1380, 0xD341, 0x1100, 0xD1C1, 0xD081, 0x1040,
0xF001, 0x30C0, 0x3180, 0xF141, 0x3300, 0xF3C1, 0xF281, 0x3240,
0x3600, 0xF6C1, 0xF781, 0x3740, 0xF501, 0x35C0, 0x3480, 0xF441,
0x3C00, 0xFCC1, 0xFD81, 0x3D40, 0xFF01, 0x3FC0, 0x3E80, 0xFE41,
0xFA01, 0x3AC0, 0x3B80, 0xFB41, 0x3900, 0xF9C1, 0xF881, 0x3840,
0x2800, 0xE8C1, 0xE981, 0x2940, 0xEB01, 0x2BC0, 0x2A80, 0xEA41,
0xEE01, 0x2EC0, 0x2F80, 0xEF41, 0x2D00, 0xEDC1, 0xEC81, 0x2C40,
0xE401, 0x24C0, 0x2580, 0xE541, 0x2700, 0xE7C1, 0xE681, 0x2640,
0x2200, 0xE2C1, 0xE381, 0x2340, 0xE101, 0x21C0, 0x2080, 0xE041,
0xA001, 0x60C0, 0x6180, 0xA141, 0x6300, 0xA3C1, 0xA281, 0x6240,
0x6600, 0xA6C1, 0xA781, 0x6740, 0xA501, 0x65C0, 0x6480, 0xA441,
0x6C00, 0xACC1, 0xAD81, 0x6D40, 0xAF01, 0x6FC0, 0x6E80, 0xAE41,
0xAA01, 0x6AC0, 0x6B80, 0xAB41, 0x6900, 0xA9C1, 0xA881, 0x6840,
0x7800, 0xB8C1, 0xB981, 0x7940, 0xBB01, 0x7BC0, 0x7A80, 0xBA41,
0xBE01, 0x7EC0, 0x7F80, 0xBF41, 0x7D00, 0xBDC1, 0xBC81, 0x7C40,
0xB401, 0x74C0, 0x7580, 0xB541, 0x7700, 0xB7C1, 0xB681, 0x7640,
0x7200, 0xB2C1, 0xB381, 0x7340, 0xB101, 0x71C0, 0x7080, 0xB041,
0x5000, 0x90C1, 0x9181, 0x5140, 0x9301, 0x53C0, 0x5280, 0x9241,
0x9601, 0x56C0, 0x5780, 0x9741, 0x5500, 0x95C1, 0x9481, 0x5440,
0x9C01, 0x5CC0, 0x5D80, 0x9D41, 0x5F00, 0x9FC1, 0x9E81, 0x5E40,
0x5A00, 0x9AC1, 0x9B81, 0x5B40, 0x9901, 0x59C0, 0x5880, 0x9841,
0x8801, 0x48C0, 0x4980, 0x8941, 0x4B00, 0x8BC1, 0x8A81, 0x4A40,
0x4E00, 0x8EC1, 0x8F81, 0x4F40, 0x8D01, 0x4DC0, 0x4C80, 0x8C41,
0x4400, 0x84C1, 0x8581, 0x4540, 0x8701, 0x47C0, 0x4680, 0x8641,
0x8201, 0x42C0, 0x4380, 0x8341, 0x4100, 0x81C1, 0x8081, 0x4040};

return ((crc & 0xFF00) >> 8) ^ tbl[(crc & 0x00FF) ^ (data &
0x00FF)];
}
```

6.3 Kommando

Hex	Dez	Name [Parameter]	Seite
0x80	128	GET CONFIG [Parameter]	(☞ 2.4.2, Seite 62)
0x81	129	SET CONFIG [Parameter]	(☞ 2.4.1, Seite 59)
0x82	130	GET CONFIG EXT [Parameter]	(☞ 2.4.4, Seite 68)
0x83	131	SET CONFIG EXT [Parameter]	(☞ 2.4.3, Seite 65)
0x84	132	FRAG START	(☞ 2.7.2, Seite 104)
0x85	133	FRAG MIDDLE	(☞ 2.7.3, Seite 104)
0x86	134	FRAG END	(☞ 2.7.4, Seite 105)
0x87	135	FRAG ACK [D-Len]	(☞ 2.7.1, Seite 104)
0x88	136	CMD ERROR [Fehlercode]	(☞ 2.8.1, Seite 106)
0x89	137	CMD WARNING [Fehlercode]	(☞ 2.8.1, Seite 106)
0x8A	138	CMD INFO [Fehlercode]	(☞ 2.3.1, Seite 54)
0x8B	139	CMD ACK	(☞ 2.8.1, Seite 106)
0x90	144	CMD FAST STOP	(☞ 2.2.19, Seite 53)
0x91	145	CMD STOP	(☞ 2.2.18, Seite 52)
0x92	146	CMD REFERENCE	(☞ 2.2.1, Seite 32)
0x93	147	CMD MOVE BLOCKED [Aktuelle Position]	(☞ 2.3.2, Seite 55)
0x94	148	CMD POS REACHED [Aktuelle Position]	(☞ 2.3.3, Seite 56)
0x95	149	GET STATE [Zeit] [Daten]	(☞ 2.6.1, Seite 84)
0x96	150	GET DETAILED ERROR INFO	(☞ 2.8.1, Seite 106)
0x97	151	CMD REFERENCE HAND	(☞ 2.2.2, Seite 33)
0x98	152	GET STATE AXIS	(☞ 2.6.2, Seite 87)
0xA0	160	SET TARGET VEL [Geschwindigkeit]	(☞ 2.2.11, Seite 47)
0xA1	161	SET TARGET ACC [Beschleunigung]	(☞ 2.2.12, Seite 48)
0xA2	162	SET TARGET JERK [Ruck]	(☞ 2.2.13, Seite 48)
0xA3	163	SET TARGET CUR [Strom]	(☞ 2.2.14, Seite 49)
0xA4	164	SET TARGET TIME [Zeit]	(☞ 2.2.15, Seite 50)
0xA6	166	SET TARGET POS [Position]	(☞ 2.2.16, Seite 50)
0xA7	167	SET TARGET POS REL [rel. Positon]	(☞ 2.2.17, Seite 51)

Hex	Dez	Name [Parameter]	Seite
0xB0	176	MOVE POS [Position] [Geschwindigkeit] [Beschleunigung] [Strom] [Ruck]	(☞ 2.2.3, Seite 35)
0xB1	177	MOVE POS TIME [Position] [Geschwindigkeit] [Beschleunigung] [Strom] [Zeit]	(☞ 2.2.5, Seite 39)
0xB3	179	MOVE CUR [Strom]	(☞ 2.2.8, Seite 43)
0xB5	181	MOVE VEL [Geschwindigkeit]	(☞ 2.2.9, Seite 45)
0xB7	183	MOVE GRIP [Strom]	(☞ 2.2.10, Seite 46)
0xB8	184	MOVE POS REL [Positionsänderung] [Geschwindigkeit] [Beschleunigung] [Strom] [Ruck]	(☞ 2.2.4, Seite 37)
0xB9	185	MOVE POS TIME REL [Positionsänderung] [Geschwindigkeit] [Beschleunigung] [Strom] [Zeit]	(☞ 2.2.6, Seite 41)
0xBA	186	MOVE POS LOOP [Position] [Geschwindigkeit] [Beschleunigung] [Strom] [Ruck]	(☞ 2.2.7, Seite 43)
0xC0	192	SET PHRASE [Datenrahmen]	(☞ 2.5.1, Seite 71)
0xC1	193	EXE PHRASE [Satznummer]	(☞ 2.5.4, Seite 73)
0xC2	194	GET PHRASES	(☞ 2.5.2, Seite 72)
0xC3	195	PRG GOTO [Satznummer]	(☞ 2.5.21, Seite 83)
0xC4	196	PRG WAIT [Zeit]	(☞ 2.5.22, Seite 83)
0xCF	207	PRG EXE [Programm Nummer]	(☞ 2.5.3, Seite 72)

Hex	Dez	Name [Parameter]	Seite
0xD0	208	EXE PHRASE0	(☞ 2.5.5, Seite 73)
0xD1	209	EXE PHRASE1	(☞ 2.5.6, Seite 74)
0xD2	210	EXE PHRASE2	(☞ 2.5.7, Seite 75)
0xD3	211	EXE PHRASE3	(☞ 2.5.8, Seite 75)
0xD4	212	EXE PHRASE4	(☞ 2.5.9, Seite 76)
0xD5	213	EXE PHRASE5	(☞ 2.5.10, Seite 77)
0xD6	214	EXE PHRASE6	(☞ 2.5.11, Seite 77)
0xD7	215	EXE PHRASE7	(☞ 2.5.12, Seite 78)
0xD8	216	EXE PHRASE8	(☞ 2.5.13, Seite 79)
0xD9	217	EXE PHRASE9	(☞ 2.5.14, Seite 79)
0xDA	218	EXE PHRASE10	(☞ 2.5.15, Seite 80)
0xDB	219	EXE PHRASE11	(☞ 2.5.16, Seite 80)
0xDC	220	EXE PHRASE12	(☞ 2.5.17, Seite 81)
0xDD	221	EXE PHRASE13	(☞ 2.5.18, Seite 81)

Hex	Dez	Name [Parameter]	Seite
0xDE	222	EXE PHRASE14	(☞ 2.5.19, Seite 82)
0xDF	223	EXE PHRASE15	(☞ 2.5.20, Seite 82)
0xE0	224	CMD REBOOT	(☞ 2.6.3, Seite 90)
0xE1	225	CMD DIO [<i>Out</i>]	(☞ 2.6.4, Seite 91)
0xE2	226	FLASH MODE [<i>Kennwort</i>]	(☞ 2.6.8, Seite 97)
0xE3	227	CHANGE USER [<i>Kennwort</i>]	(☞ 2.6.10, Seite 99)
0xE4	228	CHECK MC PC COMMUNICATION [<i>Testdaten</i>] [<i>ParamCode</i>]	(☞ 2.6.11, Seite 100)
0xE5	229	CHECK PC MC COMMUNICATION [<i>Testdaten</i>] [<i>ParamCode</i>]	(☞ 2.6.12, Seite 102)
0xE6	230	CMD DISCONNECT [<i>Kennwort</i>]	(☞ 2.6.9, Seite 98)
0xE7	231	CMD TOGGLE IMPULSE MESSAGE	(☞ 2.3.7, Seite 58)
0xE9	233	CMD GET DIO [<i>In</i>]	(☞ 2.6.6, Seite 93)
0xEA	234	CMD SET DIO [<i>Out</i>]	(☞ 2.6.7, Seite 95)
0xEB	235	CMD DIO AXIS [<i>Out</i>]	(☞ 2.6.5, Seite 92)

6.4 Info- und Fehlercodes

Hex	Dez	Name	Seite
0x0001	1	INFO BOOT	(☞ 2.8.2, Seite 109)
0x03	3	INFO NO RIGHTS	(☞ 2.8.2, Seite 109)
0x04	4	INFO UNKNOWN COMMAND	(☞ 2.8.2, Seite 109)
0x05	5	INFO FAILED	(☞ 2.8.2, Seite 109)
0x06	6	NOT REFERENCED	(☞ 2.8.2, Seite 109)
0x0007	7	INFO SEARCH SINE VECTOR	(☞ 2.8.2, Seite 109)
0x0008	8	INFO NO ERROR	(☞ 2.8.2, Seite 109)
0x09	9	INFO COMMUNICATION ERROR	(☞ 2.8.2, Seite 109)
0x10	16	INFO TIMEOUT	(☞ 2.8.2, Seite 109)
0x11	17	INFO UNKNOWN AXIS INDEX	(☞ 2.8.2, Seite 109)
0x16	22	INFO WRONG BAUDRATE	(☞ 2.8.2, Seite 109)
0x19	25	INFO CHECKSUM	(☞ 2.8.2, Seite 109)
0x1D	29	INFO MESSAGE LENGTH	(☞ 2.8.2, Seite 109)
0x1E	30	INFO WRONG PARAMETER	(☞ 2.8.2, Seite 109)
0x70	112	ERROR TEMP LOW	(☞ 2.8.2, Seite 109)
0x71	113	ERROR TEMP HIGH	(☞ 2.8.2, Seite 109)
0x72	114	ERROR LOGIC LOW	(☞ 2.8.2, Seite 109)
0x73	115	ERROR LOGIC HIGH	(☞ 2.8.2, Seite 109)
0x74	116	ERROR MOTOR VOLTAGE LOW	(☞ 2.8.2, Seite 109)
0x75	117	ERROR MOTOR VOLTAGE HIGH	(☞ 2.8.2, Seite 109)
0x76	118	ERROR CABLE BREAK	(☞ 2.8.2, Seite 109)
0x82	130	ERROR OVERSHOOT	(☞ 2.8.2, Seite 109)
0xC8	200	ERROR WRONG RAMP TYPE	(☞ 2.8.2, Seite 109)
0xD2	210	ERROR CONFIG MEMORY	(☞ 2.8.2, Seite 109)
0xD3	211	ERROR PROGRAM MEMORY	(☞ 2.8.2, Seite 109)
0xD4	212	ERROR INVALIDE PHRASE	(☞ 2.8.2, Seite 109)
0xD5	213	ERROR SOFT LOW	(☞ 2.8.2, Seite 109)
0xD6	214	ERROR SOFT HIGH	(☞ 2.8.2, Seite 109)
0xD8	216	ERROR SERVICE	(☞ 2.8.2, Seite 109)
0xD9	217	ERROR FAST STOP	(☞ 2.8.2, Seite 109)
0xDA	218	ERROR TOW	(☞ 2.8.2, Seite 109)
0xDB	219	ERROR VPC3	(☞ 2.8.2, Seite 109)

Hex	Dez	Name	Seite
0xDC	220	ERROR FRAGMENTATION	(☞ 2.8.2, Seite 109)
0xDD	221	ERROR COMMUTATION	(☞ 2.8.2, Seite 109)
0xDE	222	ERROR CURRENT	(☞ 2.8.2, Seite 109)
0xDF	223	ERROR I2T	(☞ 2.8.2, Seite 109)
0xE0	224	ERROR INITIALIZE	(☞ 2.8.2, Seite 109)
0xE1	225	ERROR INTERNAL	(☞ 2.8.2, Seite 109)
0xE4	228	ERROR TOO FAST	(☞ 2.8.2, Seite 109)
0xEB	229	ERROR RESOLVER CHECK FAILED	(☞ 2.8.2, Seite 109)
0xEC	236	ERROR MATH	(☞ 2.8.2, Seite 109)

6.5 Unterstützte Hardware

Bus-System	Karte	Auto detect	Bemerkung
Serielle Kommunikation	PC-Intern	Ja	wenn "echte" RS232 vorhanden
Serielle Kommunikation	USB-RS232 Umsetzer	Ja/ Nein	Abhängig von PC Konfiguration und Hersteller
CAN	Vector Informatik CAN-CardXL PCMCIA	Ja	
CAN	Vector Informatik alle CAN Karten	unbekannt	
CAN	esd CAN USB mini	Ja	
CAN	esd CAN PCI/331	Ja	
CAN	esd EtherCAN/2	Ja	
CAN	esd weitere CAN Karten	unbekannt	
CAN	Peak P-CAN USB	Ja	
CAN	IXXAT CAN iPC-320/PCI (VCI-Treiber Version 3)	Ja	
CAN	IXXAT weitere CAN Karten	unbekannt	
CAN	Softing CAN-ACx-PCI	Ja	
CAN	Softing weitere CAN Karten	unbekannt	
Profibus	Hilscher PCMCIA CIF-60	Ja	
Profibus	Hilscher weiter Karten	unbekannt	
Profibus	WoodHead Applicom I/O	Nein	
Profibus	CP5611	Ja	
Profibus	Siemens S7	Ja	

7 Versionshinweise

Version	Datum	Bemerkung
1.00	03.09.2007	Erstellt
1.01	29.10.2007	Korrektur
1.16	31.01.2008	Anpassung an Firmware V1.10
1.18	18.03.2008	Tippfehler korrigiert
1.20	18.06.2008	Anpassung an Firmware V1.20
1.22	29.07.2008	Anpassung an Firmware V1.22
1.23	05.08.2008	Korrektur der Beschreibungen MD-SE Parameter
1.24	02.09.2008	Korrektur
1.25	08.10.2008	Anpassung an Firmware V1.23
1.26	19.11.2008	Anpassung an Firmware V1.24
1.30	08.05.2009	Anpassung an Firmware V1.30
1.40	10.12.2009	Anpassung an Firmware V1.40
1.41	08.03.2010	Anpassung an Firmware V1.41
1.43	31.05.2010	Achsindex Funktionalität
1.50	13.01.2011	Anpassung an Firmware V1.50
1.55	30.03.2012	Anpassung an Firmware V1.55
1.56	19.06.2012	Anpassung an Firmware V1.56
1.56	10.11.2014	Korrektur
1.59	19.01.2015	Anpassung an Firmware V1.59