*phy*LOGIC®
Befehlssatz für die
*phy*MOTION® Steuerung

ORIGINAL MANUAL

Version	Änderung
7	Neue Befehle für Touchpanel, BT5 AM Terminal, AIOM und AIM
8	Ergänzung Achsenstatus-Abfrage (Kap. 6.17)
9	Befehl Schreibausgabe neu (Kap 6.4)
10	Neue Befehle, P45
11	Befehl Achsenstatus; Taktausgang
12	BiSS Encoder
13	PID Regler und T4K Modul

Phytron GmbH

Industriestraße 12

82194 Gröbenzell, Germany

Tel.: +49(0)8142/503-0

Fax: +49(0)8142/503-190

Im vorliegenden Manual "*phy***LOGIC**® *Befehlssatz für die phy***MOTION**® *Steuerung"* (http://www.phytron.de/*phy*MOTION) finden Sie die Befehle und die Programmierung für die *phy***MOTION**® Schrittmotor Steuerung.

Dieses Handbuch ist ein ergänzender Band zur Betriebsanleitung *phy***MOTION**® *Modulare Viel-Achsen-Steuerung für Schrittmotoren.*

Alle Angaben in diesem Handbuch erfolgen nach bestem Wissen, aber ohne Gewähr. Wir behalten uns im Interesse unserer Kunden vor, Verbesserungen und Berichtigungen an Hardware, Software und Dokumentation jeder Zeit ohne Ankündigung vorzunehmen.

Für Anregungen und Kritik sind wir dankbar.

E-Mail-Adresse: doku@phytron.de

Bei Fragen zur Nutzung des im Handbuch beschriebenen Produkts, die Sie hier nicht beantwortet finden, wenden Sie sich bitte an Ihren phytron-Ansprechpartner (http://www.phytron.de/) in den für Sie zuständigen Vertretungen.

1 Hinweise

Dieses Manual:

Lesen Sie vor Einbau, Inbetriebnahme und Betrieb des Gerätes dieses Manual, und ggf. mit diesem Manual in Zusammenhang stehende weiterführende Manuals gründlich durch.

- Beachten Sie während des Lesens insbesondere Hinweise, die wie folgt gekennzeichnet sind:

 **GEFAHR – Schwere Verletzung!**	*Weist auf die Gefahr von sehr wahrscheinlich eintretenden Personenschäden hin, die zu schweren Verletzungen bis hin zum Tod führen kann!*
 **GEFAHR – Schwere Verletzung durch elektrischen Schlag!**	*Weist auf die Gefahr von sehr wahrscheinlich eintretenden Personenschäden durch elektrischen Schlag hin, die zu schweren Verletzungen oder bis hin zum Tod führen kann!*
 **WARNUNG – Schwere Verletzung möglich!**	*Weist auf die Gefahr von möglichen Personenschäden hin, die zu schweren Verletzungen oder bis hin zum Tod führen kann!*
 **WARNUNG – Schwere Verletzung durch elektrischen Schlag!**	*Weist auf die Gefahr von sehr wahrscheinlich eintretenden Personenschäden durch elektrischen Schlag hin, die zu schweren Verletzungen oder bis hin zum Tod führen kann!*
 **VORSICHT – Verletzung möglich!**	*Weist auf die Gefahr von möglichen Personenschäden hin.*
 **ACHTUNG – Mögliche Schäden!**	*Weist auf die Gefahr einer möglichen Sachbeschädigung hin.*
 **ACHTUNG – Mögliche Schäden durch ESD!**	*Weist auf die Gefahr einer möglichen Sachbeschädigung durch elektrostatische Ableitströme hin.*
 **„beliebige Überschrift“**	*Weist auf eine wichtige Passage des Manuals hin.*

Beachten Sie folgende Sicherheitshinweise!

Sicherheitshinweise

ACHTUNG – Mögliche Schäden!

Beim Programmieren des Ablaufprogramms kann es zu Fehlfunktionen – z.B. Loslaufen/Abbremsen angeschlossener Motor etc. kommen.

- Testen Sie den Programmablauf daher schrittweise.

ACHTUNG – Mögliche Schäden!

Bei jeder Anwendung kann die Funktionszuverlässigkeit von Software-Produkten durch externe Faktoren wie z.B. Spannungsunterschiede oder Hardwarefehler etc. beeinträchtigt werden.

- Um Schäden durch Systemfehler vorzubeugen, sollte der Nutzer angemessene Sicherheitsmaßnahmen ergreifen. Hierzu gehören unter anderem Sicherungs- und Abschaltmechanismen.

ACHTUNG – Mögliche Schäden!

Da jedes Endnutzersystem den Kundenbedürfnissen angepasst ist und sich vom Testumfeld unterscheidet, ist der Nutzer oder Anwendungs-Entwickler für die Eignung für diese Anwendung verantwortlich.

- Die Eignung des Einsatzes dieses Gerätes ist konkret zu prüfen und zu validieren.

ACHTUNG – Mögliche Schäden!

Bei Auslieferung sind einzelne Module auf einen definierten Wert voreingestellt. So muss z.B. der Motorstrom auf den entsprechenden Wert angepasst werden (siehe hierzu die Motordaten des Motorherstellers). Durch falsch eingestellte Werte, z.B. Ströme, können angeschlossene Komponenten wie Motoren zerstört werden.

- Vor Inbetriebnahme muss überprüft werden, ob die Parameter zutreffend sind.

2 Inhaltsverzeichnis

3 Einführung

„Minilog“ oder „*phy***LOGIC**®“?

Nutzen Sie eine Steuerung der MCC, OMC oder TMC-Serie, dann verwenden Sie auch weiterhin die gewohnte Minilog-Syntax der entsprechenden Befehle.

*phy***LOGIC**® basiert auf dem Minilog-Befehlssatz und wurde um Befehle erweitert, die insbesondere komplexe Mehrachsanwendungen unterstützen und enthält auch einige leicht veränderte Befehle. Nutzen Sie eine *phy***MOTION**®, dann verwenden Sie die neue *phy***LOGIC**® Syntax.

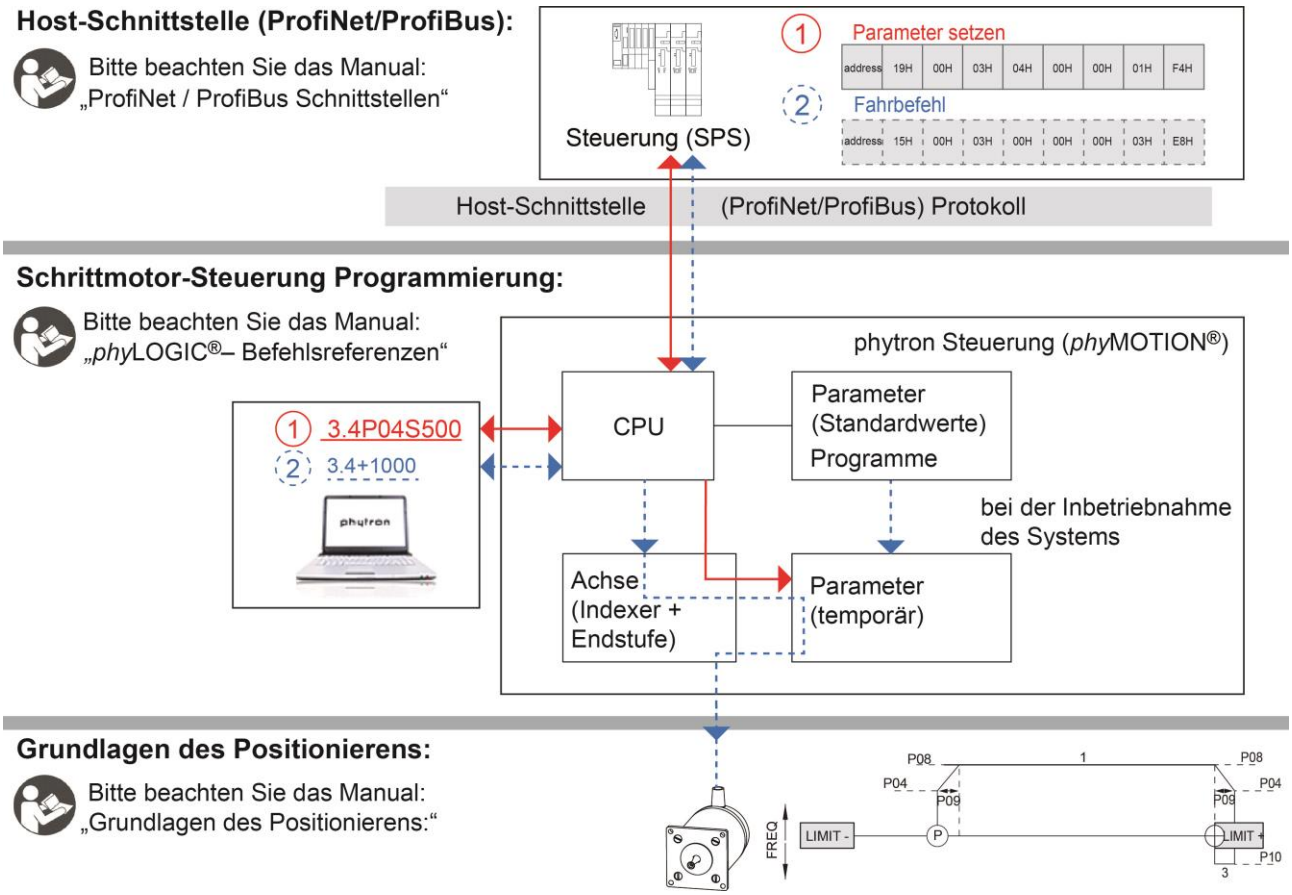
Während Minilog-Comm nur die Minilog-Syntax kennt, beherrscht die neue Software *phy***LOGIC**®-ToolBox sowohl die alte Minilog-Syntax, als auch die neue *phy***LOGIC**®-Syntax.

*phy***LOGIC**® Befehle lassen sich mit Hilfe der phytron Programmiersoftware (*phy***LOGIC**® Toolbox) über USB einfach übertragen, sowie auch in andere Protokolle wie z. B. Ethernet oder Interface Protokolle wie ProfiBus / ProfiNet implementiert werden.

Die Parametrisierung der Befehle (z.B. Fahrbefehl) je Achse kann entweder nur das erste Mal, wenn Sie ihr System einrichten, erfolgen oder Sie passen die Parameter vorübergehend an, bevor Sie einen Fahrbefehl absetzen.

Beispiel: Für “relatives Fahren” kann folgendes gesetzt werden: Schrittauflösung (P45), Laufstrom (P41), Lauffrequenz (P14), Start/Stopp-Frequenz (P04 immer “0” mit I1AM01 Modul), Rampe (P15), Beruhigungszeit (Wiederherstellungszeit?) (P16), Boost (P17), Boost-Strom (P42), Stromüberhöhungszeit (P43), etc.

Finden Sie mit Hilfe dieser Abbildung das entsprechende Manual für Ihre Programmieraufgabe:



Jeder unserer programmierbaren Steuerungen sind bereits veränderbare Parameterwerte (Auslieferungszustand) hinterlegt, die beim Starten des Gerätes in die Speicher der Achsen geladen werden. Diese Parameter können während der Programmausführung geändert werden, um sie im Programmablauf dynamisch an die nächste Fahrt anzupassen.

Wenn Sie Ihre Steuerung mit einem neuen Satz von Parametern versehen wollen, müssen Sie ihn explizit im nichtflüchtigen Speicher der Haupt-CPU-Einheit mit einem bestimmten Befehl speichern.

ACHTUNG – Mögliche Schäden!

Bei Auslieferung sind einzelne Modul auf einen definierten Wert voreingestellt. So muss z.B. der Motorstrom auf den entsprechenden Wert angepasst werden (siehe hierzu die Motordaten des Motorherstellers). Durch falsch eingestellte Werte, z.B. Ströme, können angeschlossene Komponenten wie Motoren zerstört werden.

- Vor Inbetriebnahme muss überprüft werden, ob die Parameter zutreffend sind.

Dieses Manual

*Die vollständige Befehlsreferenz zu phy***LOGIC**® *finden Sie in diesem Manual.*

Weiteres Manual

Eine Übersicht über Achs-Befehle und die damit verbundenen Parameter, sowie schematische Darstellungen der Fahrparameter finden Sie in diesem Manual:

„Grundlagen des Positionierens für Schrittmotorsteuerungen“

Weiteres Manual

*Wie Sie phy**LOGIC***®*-Befehle in Schnittstellenprotokolle einbinden, finden Sie in diesem Manual zur jeweiligen Schnittstelle:*

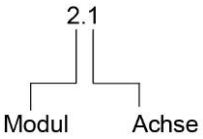
„*phy***LOGIC**™-ProfiNet/ProfiBus Interface“

Außerdem können komplette sequentielle Programme mit *phy***LOGIC**® realisiert werden: Fahrbefehle, Initialisierung der Achsen, Unterprogramme, Sprungbefehle, Lesen und setzen der Register und viele andere spezielle Befehle.

Zum Bearbeiten und Verwalten des *phy***LOGIC**® Programms wird die *phy***LOGIC**® ToolBox Kommunikationssoftware für PC mit der *phy***MOTION**® Steuerung ausgeliefert.

4 Strukturen

4.1 Struktur des Befehlcodes

2.1rzwert	**2.1**	Fettgedruckte Zeichen sind der Befehlscode und müssen unverändert eingegeben werden. In diesem Beispiel: **2.1** entspricht dem Fahrbefehl für relative Positionierung der ersten Achse des Moduls 2: 
	r	Kleingedruckte Buchstaben erfordern die Eingabe der in der Spalte *Bedeutung* beschriebenen Zeichen. In diesem Beispiel: r = Laufrichtung + oder -.
	zwert	In diesem Beispiel wird hier die Verfahrstrecke eingegeben, z.B. 1000. Die Einheit, auf die sich die Eingabe bezieht, z.B. Schritte, ist als gerätespezifischer Parameter (Kap.6) festgelegt.
Beispiel: 2.1+1000		Relativer Fahrbefehl an die Achse 1 des Moduls 2: Fahre 1000 Schritte in + Richtung.

Wichtig:

- **Alle Eingaben, die zu einem Befehl gehören, müssen ohne Leerzeichen hintereinander erfolgen.**
- **Zwischen zwei Befehlen muss ein Leerzeichen stehen!**
- **Führende Nullen eines Befehls werden ignoriert (Beispiel: der Befehl A003.1S wird als A3.1S ausgeführt).**
- **Befehle, die nicht im Programm und Direktbetrieb gleichzeitig einsetzbar sind, sind wie folgt gekennzeichnet:**
  1. **Befehl nur im Programm einsetzbar (NUR PROG)**
  2. **Befehl nur im PC-Direktbetrieb einsetzbar (NUR PC)**

Ausnahme: Bei der Befehlsgruppe „Programm- und Dateiverwaltung (NUR PC)“, Kap. 6.16, muss der erste alphanumerische Programmname durch eine Leerstelle oder Strichpunkt (;) vom nachfolgenden alphanumerischen Teil des Befehlscodes getrennt werden.

4.2 Daten und Telegrammformat

Datenformat: No Parity

1 Stopbit

8 Bit ASCII-Code

115 200 Baud (bei Auslieferung)

Adresse: ist immer 0
Ausnahme: mit RS485: 0 bis F Drehschalter

Das **Sendetelegramm** vom PC via RS Interface ist wie folgt definiert:

<STX> | Adresse | Befehl | Separator | Prüfsumme | <ETX>

Das **Antworttelegramm** (immer bei Adresse 0-9, A-F) ist wie folgt definiert:

<STX> | ACK | Antwort | Separator | Prüfsumme | <ETX> oder

<STX> | ACK | Separator | Prüfsumme | <ETX> oder

<STX> | NAK | Separator | Prüfsumme |<ETX>

	Bedeutung
<STX>	<STX> (Start of Text, 02_H): als Kennzeichen für den Start eines neuen Telegramms.
Adresse	Adresse der Steuerung mit den Werten „0“ bis „9“ und „A“ bis „F“ (30_H..39_H bzw. 41_H..46_H). Außerdem die Broadcast[1] Adresse @ (40_H).
Befehl	*phy***LOGIC**® Befehlscode
Separator	: Doppelpunkt ($3A_H$) zur Trennung von Nutzdaten und Prüfsumme.
Prüfsumme	Höherwertiges Byte der Prüfsumme (Berechnung s.u.)
	Niederwertiges Byte der Prüfsumme (Berechnung s.u.)
<ETX>	(End of Text, 03_H) als Telegrammende-Kennung.
ACK	(**Ack**nowledge 06_H), der Befehl wurde quittiert.
NAK	(**N**egative **A**c**k**nowledge 15_H), der Befehl wurde negativ quittiert.
Antwort	Antwort als Zahl oder String, z.B. E oder N

Die Prüfsumme CS wird berechnet, indem – beginnend beim Adressbyte – alle Bytes einschließlich des Separators (:) mit einer Exklusiv-Oder-Verknüpfung (⊕) aufsummiert werden:

```
CS = Adresse ⊕ Datenbyte 1 ⊕ Datenbyte 2 . . . ⊕ Datenbyte n ⊕
     Separator
```

[1] Broadcast: Alle Achsen empfangen den String und werten ihn aus.
Da alle Achsen auch fast zeitgleich antworten würden und somit unweigerlich ein Buskonflikt entstünde, wird die Antwort der Steuerung bei Adressierung per „@“ unterdrückt, keine Achse antwortet.

Die Prüfsumme CS wird als binärer Byte-Wert berechnet, das Ergebnis ist ein Byte im Wertebereich 00_H bis FF_H. Dieses Byte wird in zwei Hälften (Nibbles) zerlegt, jeweils mit dem Wertebereich 0_H bis F_H. Dann werden die den Nibbles entsprechenden lesbaren ASCII Zeichen ins Telegramm geschrieben, „0“ bis „9“ statt 0_H bis 9_H und „A“ bis „F“ für A_H bis F_H (rechnerisch wird auf den Nibble Wert 30_H bzw. 37_H addiert).

Die Steuerung berechnet beim Empfang eines Telegramms die Prüfsumme über die empfangenen Bytes und vergleicht sie mit der empfangenen Prüfsumme. Bei einer Abweichung wird das empfangene Telegramm verworfen, und der Fehler mit der Antwort NAK quittiert.

Falls auf die Absicherung des Telegramminhaltes durch die Prüfsummenüberwachung kein Wert gelegt wird, kann diese auch ausgeschaltet werden. Anstelle der Prüfsummenbytes werden auch **zwei X** akzeptiert, im Beispiel also:

<STX>| 0 | 1 | . | 1 | + | 1 | 0 | 0 | : | X | X |<ETX>

5 Programmierung mit *phy*LOGIC®

5.1 Aufbau von *phy*LOGIC® Programmen

- *phy***LOGIC**® Programme bestehen aus bis zu n Programmzeilen, die im Editor angezeigt werden.
- Die einzelnen Befehle in der Programmzeile müssen durch Leerzeichen voneinander getrennt werden.
- Zwischen die zu einem Befehl gehörenden Zeichen keine zusätzlichen Leerzeichen einfügen.
- Die Befehle werden seriell abgearbeitet.
- Mit Hilfe von Label können Sprungbefehle und Unterprogramme definiert werden.
- Parameter und Registerwerte sollten am Programmanfang definiert sein.
- Parameter- und Registernummern können mit oder ohne führende Nullen eingegeben werden.
  Beispiel: R0001 oder R1

5.2 Adressierungsarten

Für Befehle bei denen mindestens ein Operand ein Register ist, sind grundsätzlich zwei Adressierungsarten verfügbar: Die **direkte** Adressierung und die **indirekte** Adressierung. In den nachfolgenden Beschreibungen wird immer der Grundbefehl in **direkter** Adressierungsart erklärt. Die Varianten der **indirekten** Adressierung sind der Vollständigkeit halber aufgeführt. Das Zielregister steht im Befehlscode immer an erster Stelle.

Beispiel **Direkte Adressierung:**

Befehl	***Bedeutung***
Rnn**BE**m.n**–**m.y	Das Register nn wird mit dem Status der Eingänge m.n bis m.y binär beschrieben. **Beispiel: R1BE1.1–1.8** Die Eingänge 1 bis 8 des Modul 1 haben z. B. den Zustand: **1010 0101.** Das Register 1 wird nun mit dem Binärwert der Eingänge beschrieben. Nach dem Befehl hat das Register 1 den Wert 165.

Beispiel für Indirekte Adressierung:

Befehl	***Bedeutung***
R[Rnn**]BE**m.n**–**m.y	**Indirekte Adressierung:** Das Register, das durch das Register **[Rnn]** adressiert wird, wird mit dem Status der Eingänge m.n bis m.y binär beschrieben Beispiel: **R1S10 R[R1]BE1.1–1.8** Das Register **[R1]** wird auf den Wert 10 gesetzt. Der Status der Eingänge 1 bis 8 des Moduls 1 ist z. B. 1010 0101. Das Register 10, das durch das Register 1 ([R1]) adressiert wird, wird nun mit dem Binärwert der Eingänge beschrieben. Nach dem Befehl hat das Register 10 den Wert 165.

Adressierung mit Label:

Bei Sprungbefehlen (siehe Kap. 6.13) und Unterprogrammaufrufen (siehe Kap. 6.29) kann die Ziel- bzw. Startzeile im Befehlscode mit einem Label (*la*), das dieser Programmzeile zugeordnet wurde, angeben werden. Ein Label steht zwischen zwei * und kann bis zu 18 alphanumerische Zeichen haben. Innerhalb eines Programms können bis maximal 512 Labels eingesetzt werden.

Beispiel: *Labelname*

Programmname:

Programmnamen [name] können bis zu 16 alphanumerische Zeichen haben.

5.3 Bedingte Befehle

Die Ausführung einiger Befehle (z. B. Sprungbefehle, Unterprogrammaufruf) kann mit einer Bedingung verknüpft sein. Bevor bedingte Sprünge usw. eingesetzt werden können, muss das Bedingungsbyte vorher z. B. durch eine Eingangsabfrage (siehe Kap. 6.6) oder durch einen Register Vergleich (siehe Kap. 6.12) gesetzt werden.

Mögliche Zustände des Bedingungsbytes:

E = Bedingung erfüllt **N** = Bedingung nicht erfüllt

Der Zustand des Bedingungsbytes wird mit dem nächsten Befehlscode geändert.

Alle Befehle, die keine Bedingung setzen, löschen die Bedingungsabfrage.

6 *phy*LOGIC® Befehle

6.1 AD Wandler

Befehl	***Bedeutung***
ADm**.**n	Lesen von Kanal n des AD Wandlers des Moduls m. **Antwort:** **<STX><ACK> wert :CS<ETX>** wert=0 bis 16383 oder ±8191
ADm**.**n**B**z	Block z von Kanal n des Moduls m (des AD Wandlers) lesen. z = 1 bis 64 **Antwort:** **<STX><ACK> wert;wert;wert;….. :CS<ETX>** 128 „wert“ pro Block
ADm**.**n**R**	Messwertaufnahme des Kanals n des AD Wandler-Moduls m starten. **Antwort:** **<STX><ACK>:CS<ETX>**
ADm**.**n**S**	Messwertaufnahme des Kanals n des AD Wandler-Moduls m stoppen. **Antwort:** **<STX><ACK> :CS<ETX>**
ADm**.**n**T**	Funktion des Kanals n des AD Wandler-Moduls m lesen. **Antwort:** **<STX><ACK> wert :CS<ETX>** wert=0,1 oder 2
ADm**.**n**T**wert	Kanal n des AD Wandler-Moduls m auf Funktion setzen wert=0 → Spannung unipolar wert=1 → Spannung bipolar wert=2 → Strom unipolar
i	Die Messwerte werden kartenbezogen gelesen, ohne Angabe des Kanals:
ADm**V**x	Messwert x aus aufgezeichneten Block des Moduls m lesen.
Ryy**=AD**m**V**x	Messwert x aus aufgezeichneten Block des Moduls in das Register y schreiben.
ADm**W**	Anzahl der aufgezeichneten Messwerte des Moduls m lesen.
Ryy**=AD**m**W**	Anzahl der aufgezeichneten Messwerte des Moduls m in das Register y schreiben.

Befehl	***Bedeutung***
ADm**Z**wert	Messzyklus am Modul m einstellen wert → Messzyklus von 1 bis 3000 1=33 µs 2=66 µs…. 3000=100 ms
ADm**ZR**	Zeitwert des Messzyklus am Modul m lesen **Antwort:** **<STX><ACK> wert :CS<ETX>** wert=1 bis 3000

6.2 PID Regler

ADMm**.**n	Messwert des PID Moduls m am Messeingang n lesen
ADMm**.**n**D**	d-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm**.**n**DA**	Aktuellen d-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm**.**n**D=**xxxx	d-Anteil für PID Regelung am Modul m für Reglerausgang n setzen xxxx→ 0 bis 65535
ADMm**.**n**F**	Frequenz des PID Moduls m am Reglerausgang n lesen
ADMm**.**n**F=**xxxx	Frequenz für PID Regelung am Modul m für alle Reglerausgänge setzen xxxx→ 1 bis 1000 Hz
ADMm**.**n**I**	i-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm**.**n**IA**	Aktuellen i-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm**.**n**I=**xxxx	i-Anteil für PID Regelung am Modul m für Reglerausgang n setzen xxxx→ 0 bis 65535
ADMm.n**OFF**	Regelung am Modul m, Reglerausgang n ausschalten
ADMm.n**ON**	Regelung am Modul m, Reglerausgang n anschalten
ADMm.n**P**	p-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm.n**PA**	Aktuellen p-Anteil des PID Moduls m am Reglerausgang n lesen
ADMm.n**P=**xxxx	p-Anteil für PID Regelung am Modul m für Reglerausgang n setzen xxxx→ 0 bis 65535
ADMm.n**R**	Reglerstartpunkt R des PID Moduls m am Reglerausgang n lesen

Befehl	***Bedeutung***
ADMm.n**R=**xxxx	Reglerausgang n starten, wenn Messeingang n den Wert xxxx erreicht hat xxxx= Sollwert minus Wert R xxxx→ 0 bis max. Sollwert
ADMm.n**S**	Sollwert des PID Moduls m am Messeingang n setzen
ADMm.n**S=**xxxx	Sollwert des PID Moduls m am Messeingang n setzen xxxx→ 0 bis 65535 (in 1/10 Grad)

6.3 Digitale Ausgänge

Befehl	***Bedeutung***
	Ausgänge schalten
Am**.**nz	Setzen eines Ausgangs. z→ Zustand des Ausgangs z = S → setzen z = R → rücksetzen m→ Nummer (ID) des Moduls n, y, x → Nummer (ID) des Ausgangs
Am**.**n**==**z	Ausgangsstatus des Ausgangs n des Moduls m lesen und Bedingungsbyte setzen. Wenn z=S, Ausgang ist ON. Wenn z=R, Ausgang ist OFF.
Am**.**nzm**.**yzm**.**xz	Mehrere Ausgänge setzen/lesen. **Beispiel: A1.1S1.2R1.3S** Ausgang **1** und **3** ON, Ausgang **2** OFF
	Ausgangsgruppe lesen
AGm**R**	Die Ausgangsgruppe der Modulnummer (m) lesen. (NUR PC) **Beispiel: AG2R** Die 2. Ausgangsgruppe wird gelesen **Antwort:** **<STX><ACK> wert :CS<ETX>** (NUR PC) wert = 0 bis 255
	Ausgangsgruppe Ausgänge setzen
AGm**S**zwert **AG**m**=**zwert **AG**m**=R**nn **AG**m**=R**[**R**nn]	Ausgangsgruppe setzen m= Modulnummer wert: 0 bis 255 **Beispiel: AG1S170** Die 1. Ausgangsgruppe wird mit der Information '10101010' gesetzt

Befehl	***Bedeutung***
	Ausgangsimpuls erzeugen (DIOM)
AIm.n;zeit	Ausgangsimpuls an Ausgang n des Moduls m mit zeit (in ms) setzen**.** Beispiel: AI3.1;100
Am.n**D=**wert	Am Ausgang n des Moduls m die periodische Einschaltdauer definieren. wert=1 bis 100 % Beispiel: A1.1D=50
Am.n**F=**wert	Ausgang n des Moduls m mit Frequenz wert pulsen wert=1 bis 1000 Hz Beispiel: **A**2.1**F**=100 Bei Frequenzwert 0 wird die Einschaltdauer auf 100% gesetzt. Damit wird die Funktion Pulsen ausgeschaltet
	PID Messmodul
AMm**.**n	Ausgang n des Moduls m schalten
AMIm.n**;**wert	Ausgangsimpuls an Ausgang n des Messmoduls m mit wert (in ms) setzen**.** Beispiel: AMI2.1;1000
AMm.n**D=**wert	Am Ausgang n des Messmoduls m die Einschaltdauer definieren. wert=1 bis 100 % Beispiel: AM1.1D=50
AMm.n**F=**wert	Ausgang n des Messmoduls m mit Frequenz wert pulsen wert=1 bis 1000 Hz Beispiel: **AM**3.4**F**=0 Bei wert=0 (Frequenz ist gleich Null) wird die Funktion Pulsen ausgeschaltet.

Befehl	***Bedeutung***
	Ausgangszustand lesen
AMZm.n;m.y;m.x	Der Zustand der Ausgänge (des Messmoduls m) n, y, x wird gelesen. (NUR PC)
	Beispiel: AMZ1.1;1.2;1.3
	Antwort : <STX><ACK>rrr:CS<ETX>
	r = 0 Ausgang OFF
	r = 1 Ausgang ON
	Wichtig: Zwischen den Ausgangsnummern ein **;** setzen.
AZm.n;m.y;m.x	Der Zustand der Ausgänge (des Digitalmoduls m) n, y, x wird gelesen. (NUR PC)
	Beispiel: AZ1.1;1.2;1.3
	Antwort : <STX><ACK>rrr:CS<ETX>
	r = 0 Ausgang OFF
	r = 1 Ausgang ON
	Wichtig: Zwischen den Ausgangsnummern ein ; setzen.

6.4 Reset

Befehl	***Bedeutung***
CR	Die Steuerung wird über die Schnittstelle zurück gesetzt.
CT	Die gesamte Bedienterminalanzeige wird über die Schnittstelle gelöscht.
CTn	Eine einzelne Zeile löschen. n--> Zeilennummer n=1 bis 4 (BT5 AM Terminal)
CTn;m	Ausgewählte Zeilen löschen. n oder m--> Zeilennummer n=1 bis 4; m=1 bis 4 (BT5 AM Terminal)
	Antwort : <STX><ACK><ETX><CR><LF> (NUR PC)

6.5 Schreibausgabe über serielle Schnittstelle

Befehl	*Bedeutung*
	Es können über 3 serielle Schnittstellen Informationen ausgegeben werden. Die Schreibausgabe erfolgt ohne Formatierung. s = 1,2 oder 3 → Schnittstellenbezeichnung 1 → Schnittstelle USB 2 → Terminalschnittstelle 3 → Schnittstelle Feldbus, RS und Ethernet
Ds= **<**Text**>**	Den Text, der in Klammern steht, ausgeben.
Ds=**R**nn	Den Inhalt des Registers nn ausgeben.
Ds=**R[R**nn**]**	Den Inhalt des Registers, das durch das Register nn addressiert wird, ausgeben.
Ds=m**.**n**P**mm	Den Parameter mm der Achse n des Moduls m ausgeben. mm = 1 bis max. Parameternummer **Beispiel: D1=5.2P10** Hier wird der Parameter 10 der Achse 2 des Moduls 5 über die USB Schnittstelle ausgegeben.

6.6 DA Wandler

Befehl	***Bedeutung***
DAm**.**n=wert	An Kanal n des DA Wandler-Moduls m wert ausgeben. wert=0 bis 65535 oder ±32767
DAm**.**n	Lesen von Kanal n des DA Wandler-Moduls m. **Antwort:** **<STX><ACK> wert :CS<ETX>** wert=0 bis 65535 oder ±32767
DAm**.**nTwert	Kanal n des DA Wandler-Moduls m auf Funktion setzen wert=0 → Spannung unipolar wert=1 → Spannung bipolar wert=2 → Strom unipolar
DAm**.**nT	Funktion des Kanals n des DA Wandler-Moduls m lesen. **Antwort:** **<STX><ACK> wert :CS<ETX>** wert=0,1 oder 2

6.7 Eingangsabfragen

Befehl ***Bedeutung***

Wichtig: Die Befehle zur Abfrage digitaler Eingänge für das DIOM-Modul sind auch verwendbar für andere Module mit digitalen Eingängen (z.B. PIDM-Modul,..).

UND-Verknüpfung

E^m**.**nzm**.**yzm**.**xz Es werden die Eingänge n, y, x als UND-Verknüpfung abgefragt. Wenn die UND-Bedingung erfüllt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt.

m → Modulnummer
n. y. x → Nummer des Eingangs
z = S → Eingang ON
z = R → Eingang OFF

Beispiel: E^1.1S1.2R1.3S
Hier werden die Zustände der Eingänge 1, 2 und 3 abgefragt. Wenn Eingang 1 gesetzt, 2 rückgesetzt und 3 gesetzt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt.
Nun kann ein bedingter Sprung oder ein bedingter Unterprogrammaufruf ausgeführt werden.

Antwort: <STX><ACK> E :CS<ETX> oder
<STX><ACK> N :CS<ETX> (NUR PC)

ODER Verknüpfung

Evm**.**nzm**.**yzm**.**xz Es werden die Eingänge n, y, x als ODER Verknüpfung abgefragt. Wenn die ODER-Bedingung erfüllt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt.

m → Modulnummer
n/y/x → Nummer des Eingangs
z = S → Eingang ON
z = R → Eingang OFF

Beispiel: Ev1.1S1.2R1.3S

Hier werden die Zustände der Eingänge **1, 2 und 3** abgefragt. Wenn der Eingang 1 gesetzt oder 2 rückgesetzt oder 3 gesetzt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt.
Nun kann ein bedingter Sprung oder ein bedingter Unterprogrammaufruf ausgeführt werden.

Antwort: <STX><ACK> E :CS<ETX> oder
<STX><ACK> N :CS<ETX> (NUR PC)

Befehl	***Bedeutung***
	Warten bis Zustand erfüllt
Em**.**nz	Warten auf den vorgegebenen Eingangszustand. Das Programm wartet hier, bis der Zustand erreicht wird. Das Bedingungsbyte wird nicht beeinflusst. (NUR PROG)
Em**.**nzm**.**yz	Bei Eingangsabfragen mit mehreren Eingängen werden die Zustände der Eingänge nacheinander abgefragt (keine UND-Verknüpfung). Das Bedingungsbyte wird nicht beeinflusst. (NUR PROG)
	Beispiel: E1.1S1.2R1.3S
	Hier werden die Zustände der Eingänge 1, 2 und 3 (von Modul 1) abgefragt. Wenn der Eingang 1 gesetzt ist, wird der Eingang 2 abgefragt. Wenn der Eingang 2 rückgesetzt ist, wird der Eingang 3 abgefragt. Ist der Eingang 3 gesetzt, dann ist der Befehl abgearbeitet und das Programm wird fortgesetzt. Bei Befehlsende können die Eingänge 1 und 2 schon wieder einen anderen Zustand haben.
Em**.**n**==**z	Eingangszustand des Eingangs n des Moduls m lesen und Bedingungsbyte setzen. Wenn z=S, Eingang ist ON. Wenn z=R, Eingang ist OFF.
Em**.**n**=**z**;**Befehl	Dem Eingang n des Moduls m wird der Befehl zugeordnet. z=S, Befehl wird bei Wechsel von AUS auf EIN ausgeführt. z=R, Befehl wird bei Wechsel von EIN auf AUS ausgeführt. z=C, setzt den Befehl zurück, der dem Eingang zugeordnet war.
	Beispiel 1: E1.2=S;R1+10 Bei Wechsel des Eingangs von AUS auf EIN wird der Befehl R1+10 ausgeführt bis der Befehl rückgesetzt wird.
	Beispiel 2: E1.2=S;U*test* Im Programmablauf wird bei jeden Wechsel des Eingangs von AUS auf EIN das Unterprogramm *test* ausgeführt.
	Wichtig: Befehl→Register oder Unterprogramme
ECm**=**n	Eingang 1 des Moduls m als Zähler bei mit Flanke n setzen m→Modulnummer n→Flanke n=0 steigende Flanke n=1 fallende Flanke

Befehl	***Bedeutung***
ECm**C**	Eingangszähler 1 des Moduls m Null setzen m=Modulnummer **Antwort: <STX><ACK> :CS<ETX>**
ECm**R**	Eingangszähler 1 des Moduls m lesen m=Modulnummer **Antwort: <STX><ACK> wert:CS<ETX>**
	Eingangsgruppe lesen
EGm**R**	Die Eingangsgruppe der Modulnummer m lesen. **Antwort: <STX><ACK> wert :CS<ETX>** (NUR PC) wert = 0 bis 255
EMm**.**n	Eingang n des Messmoduls m lesen.
EZm**.**n**;**m**.**y**;**m**.**x	Der Zustand der Eingänge n, y, x wird gelesen. (NUR PC). **Antwort: <STX><ACK>nnn:CS<ETX>** n = 0 Eingang nicht gesetzt n = 1 Eingang gesetzt **Wichtig:** Zwischen den Eingangsnummern ein **;** setzen.

6.8 Programmbeeinflussung bei Nothalt (NUR PROG)

Befehl	***Bedeutung***
FN* la****	Es wird die Zeile, an der das Programm bei Nothalt fortfahren soll, durch ein Label bestimmt.

6.9 Steuerung auf Werkseinstellung setzen

Befehl	***Bedeutung***
GW*.*	Steuerungswerte und Achsenparameter auf Grundwert setzen und alle Register und Programme löschen

6.10 Programmunterbrechung

Befehl	***Bedeutung***
H	Das Programm wartet hier, bis alle Achsen stehen. (NUR PROG)

6.11 Systemanpassung im Programmablauf

Befehl	***Bedeutung***
	Achsenanzahl
IAR	Die Anzahl der vorhandenen Achsenmodule auslesen. (NUR PC) **Antwort: <STX><ACK>x:CS <ETX>** x=1 bis n AchsenModul
IAn	Die Anzahl der vorhandenen Achsen pro Modul auslesen.(NUR PC). **Antwort: <STX><ACK>a:CS <ETX>** a = 1 bis 4 pro Modul
IACn	Den Encodertyp lesen (NUR PC) n=1 bis Anzahl der Achsenmodule **Antwort: <STX><ACK>a:CS <ETX>** a → Encoder Modul vom Typ 4 Byte: Beispiel: 000A a=0 → kein Modul a=A → Encoder ECAS01 a=B → Encoder ECBS01 a=E → Encoder ECES01 a=M→ Encoder ECMS01
IAEn	Den Endstufentyp lesen (NUR PC) n=1 bis Anzahl der AchsenModul **Antwort: <STX><ACK>a:CS <ETX>** a → Endstufen Modul vom Typ 4 Byte: Beispiel: 00aa a=0 → Endstufe unbekannt a=A → APS 5A intern a=B → APS 9A intern a=I → I1AM01 a=L → LPS 9A a=M → MSX 102-120 a=S → MSX 52-120 a=T → I1AM03 a=X → MSX 152-120 a=Z → ZMX+

0	0	0	A
4.Achse	3.Achse	2.Achse	1.Achse

0	0	a	a
4.Achse	3.Achse	2.Achse	1.Achse

Befehl	***Bedeutung***
IATn	Den Temperaturmodultyp lesen(NUR PC). n=1 bis Anzahl der AchsenModul **Antwort: <STX><ACK>a:CS <ETX>** a → Temperaturmodul Typ 4 Byte Beispiel: 00PP

0	0	P	P
4.Achse	3.Achse	2.Achse	1.Achse

a=0 → kein Modul
a=P → PT 100 Sensor
a=K → K Typ

Automatikstart (nur PC)

IBC	Programmname aus Autostartregister löschen.
IBR	Der Programmname für den Autostart wird ausgelesen (NUR PC). **Antwort: <STX><ACK>name:CS<ETX>**
IBSname	Der Programmname wird in das Autostartregister geschrieben. Mit diesem Programm wird gestartet, wenn der REMOTE/LOCAL-Schalter auf LOCAL steht. **Antwort: <STX><ACK>:CS<ETX>** oder **<STX><NAK>:CS<ETX>** (NUR PC)

ProfiBus Ethercat oder CAN Daten (für Servicezwecke)

IBER	Ethercat und CAN empfangene Daten lesen
IBES	Ethercat und CAN Daten geschriebene Daten lesen
IBPRx**;**y	ProfiBus und ProfiNet empfangene Daten lesen x→Start y→Ende
IBPSx**;**y	ProfiBus und ProfiNet gesendete Daten werden gelesen

Baudrate einstellen/lesen (NUR PC)

ICn**S**baud	Die Baudrate für die Schnittstellen (bei RS/USB, CAN und Bluetooth) setzen. n = 1 → COM 1 für RS und USB: baud = Baudrate (9600, 19200, 38400, 57600, 115200 oder 230400 Baud)

Befehl	***Bedeutung***
ICn**R**	Baudraten-Einstellung der Schnittstellen wird ausgelesen. n = 1 bis 3 1→ USB 2→ Debug 3→ Feldbus **Antwort, wenn n=3 und Feldbustyp>=3 und <6: <STX><NAK>:CS<ETX>**
	Feldbus lesen
IC3**T**	Nur für Feldbus! Schnittstellentyp für Feldbus lesen **Antwort: <STX><ACK>n:CS<ETX>** n = 0→kein Modul n= 1→ RS 232 oder RS 485 sind implementiert n = 3→ ProfiBus n = 4→ ProfiNet n = 5→ Ethernet
IC3**HVS**x	Nur für RS Bus! (wenn Feldbus=1) Setze RS Bus x=0 → RS 485 4-Draht x=1 → RS 485 2-Draht
IC3**HVR**	RS Bus lesen! (wenn Feldbus=1) **Antwort: <STX><ACK>n:CS<ETX>** x=0 → RS 485 4-Draht x=1 → RS 485 2-Draht
	ProfiBus einstellen/setzen (wenn Feldbus=3)
IDR	Steuerungsadresse für ProfiBus lesen Wenn **Antwort: <STX><ACK>(1…125):CS<ETX>**, dann ist Grundeinstellung =1
IDSn	Steuerungsadresse für ProfiBus setzen n = 1 bis 125
	Remote/Local Umschaltung (NUR PC)
IFR	Die Steuerung wird auf Remote-Funktion umgeschaltet. Wenn ein Programm läuft, wird es abgebrochen. Bei Schalterstellung Local wird die Stellung Remote simuliert. **Antwort: <STX><ACK>:CS<ETX>**

Befehl	***Bedeutung***
IFL	Die Steuerung wird auf Local-Funktion umgeschaltet, wenn der Remote/Local Schalter auf Local steht. Steht der Schalter auf Remote, wird nicht umgeschaltet.
	Antwort: <STX><ACK>:CS<ETX>
	Modul Identifikation (NUR PC)
IIPR	IP Adresse des Ethernet Moduls lesen.
IMA	Anzahl der Achsen-Module lesen.
IMAn	Die Anzahl der vorhandenen Achsen pro Modul auslesen.(NUR PC). **Antwort: <STX><ACK>a:CS <ETX>**
	a = 1 bis 4 pro Modul
IMAI	Anzahl der Analog-Eingangsmodule lesen.
IMAIO	Anzahl der Analog-Ein-und AusgangsModule lesen.
IMAO	Anzahl der Analog-Ausgangsmodule lesen.
IMDI	Anzahl der Digital-Eingangsmodule lesen.
IMDIO	Anzahl der Digital-Ein-und Ausgangsmodule lesen.
IMDO	Anzahl der Digital-Ausgangsmodule lesen.
IMn	Modulsteckplatz nach Modultyp abfragen.
	n =0 bis max. Anzahl aller Module
	n=0→MCM Modul
	Antwort: <STX><ACK>a:CS<ETX>
	a=I1AM01 → Einachs-Schrittmotor-Ansteuerungsmodul (3,5 A) a=I1AM02 → → Einachs-Schrittmotor-Ansteuerungsmodul (5 A) a=IDXM01 → 4-Achsen-Indexer Modul a=DIOM01 → Digitales I/O Modul a=MCM01 → Main Controller Modul a=AIOM01 → Analoges I/O Modul a=AIM01 → Analoges Eingangs-Modul a=AOM01 → Analoges Ausgangs-Modul a=T4KM01 → Temperatur-Auswertungs-Modul für Temperaturfühler Typ K
IMt4K	Anzahl der T4K Module lesen
IMR	MAC Adresse lesen.
	PID Modul
IPIDEm	Messeinheit des Messmoduls lesen
	Antwort: $D_{[4]};D_{[3]};D_{[2]};D_{[1]}$

Befehl	***Bedeutung***
	D→Degree (Grad); $D_{[1]}$→ am Messeingang1… $D_{[4]}$→ am Messeingang4 V→Volt 0→kein Modul gesteckt
IPIDSm	Die Skalierung der Messeinheit des Messmoduls lesen 1→1/1 10→1/10
IPIDTm	Den Typ des Messmoduls lesen Antwort: K→KTS oder P→PTS
	Statische IP Adresse
IPR	IP Adresse lesen (gilt nicht für ProfiNet)
IPS192.168.0.10	IP Adresse statisch setzen auf 192.168.0.10
IPS0.0.0.0	Statische IP Adresse löschen
	Betriebsart
IPT=x	Befehls-Betriebsart definieren x=0: *phy***LOGIC**® Modus x=1: G-Code Modus x=2: *phy***LOGIC**® und IXE Befehlssatz Modus x=3: *phy***LOGIC**® und OMC/TMC Befehlssatz Modus x=4; G-Code Minilog IXE
IPTR	Befehls-Betriebsart lesen Antwort: phyLOGIC, G-Code, phyLOGIC/IXE oder phyLOGIC/OMC
	Inhaltsverzeichnis Flash
IP	Anzahl der Programme auf der Steuerung abfragen.
IPM=n	Nur für ProfiBus und ProfiNet: Grundeinstellung: 1 mal 8-Byte für MCM Modul (Master) Anzahl der 8-Byte Blöcke anlegen, die nach einem Reset aktiv bleiben n=1 bis 4 →1 bis 4 mal 8 Byte
IPMR	Die Anzahl der 8 Byte Blöcke, die für das MCM Modul angelegt sind, auslesen
IR	Anzahl der gespeicherten Registersätze auf der Steuerung abfragen.
IRn	n-ten Registernamen der Programmliste vom Flash auslesen. (NUR PC). **Antwort: <STX><ACK>name:CS <ETX>** **Antwort: <STX><NAK>:CS <ETX>** wenn kein Name vorhanden
ITR	Anzahl der gespeicherten Textregister auf der Steuerung abfragen.

Befehl	***Bedeutung***
ITRn	n-ten Textregisternamen der Programmliste vom Flash auslesen (NUR PC). **Antwort: <STX><ACK>name:CS <ETX>** **Antwort: <STX><NAK>:CS <ETX>** wenn kein Name vorhanden
ITAIOn	Abfrage des Analog-Modultyps **Antwort: <STX><ACK>typ:CS <ETX>** n=1 bis maximale Anzahl der Analog-Module typ=AIOM01
ITIOn	Abfrage des Digital-Modultyps **Antwort: <STX><ACK>typ:CS <ETX>** n=1 bis maximale Anzahl der Digital-Module typ=DIOM01
ITIDXn	Abfrage des Indexer-Modultyps **Antwort: <STX><ACK>typ:CS <ETX>** n=1 bis maximale Anzahl der Indexer-Module typ=ID4XM01, I1AM01 oder I1AM02
ITT4kn	Abfrage des Temperatur-Auswertungs-Moduls **Antwort: <STX><ACK>typ:CS <ETX>** typ=T4KM01
	Systemname
ISN=name	*phy***MOTION**® wird ein Name zugewiesen, der auch wieder gelesen werden kann
ISN	Der Systemname wird zurückgelesen
	Versionsabfrage
IV	Die Anzahl der Module auslesen.
IVAIOn	System Softwarestand des n-ten AIOM Moduls lesen.
IVIOn	System Softwarestand des n-ten DIOM Moduls lesen.
IVIDXn	System Softwarestand des n-ten I1AM oder IDX Moduls lesen.
IVM	System Softwarestand des MCM Moduls lesen.
IV0	Die Softwareversion des MCM Moduls lesen (Loader, System-Lib, MiniLog System)
IVm	Die Softwareversion des Moduls m (Loader, System-Lib, MiniLog System) wird gelesen (NUR PC). m=1 bis max. Module **Antwort: <STX><ACK>Software Version:CS<ETX>**

Befehl	***Bedeutung***
	Feldbusschnittstelle (Deutschmann) abfragen
IVB	Scriptversion lesen
IVPIDn	Softwareversion des PID Moduls lesen
IVT4Kn	Softwareversion des T4K Moduls lesen

6.12 Interpolationsbefehle (NUR für I4XM01)

Linearinterpolation

Strecke oder Position in P18 der Achse eintragen (z.B.1.1P18S5000 1.2P18S1000)

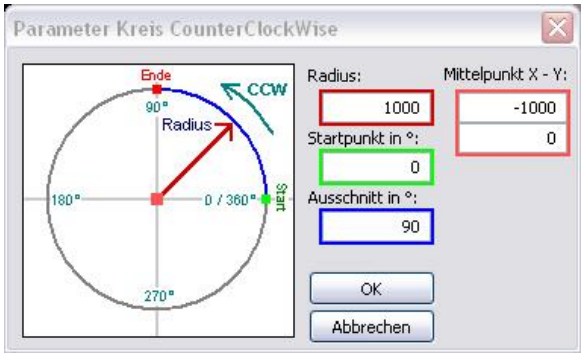
m**I**aw**;**bw**;**cw**;**dw Start der Interpolation

m → Modulnummer
a,b,c,d → Nummer der Achse
w=A → absolut fahren
w=R → relativ fahren

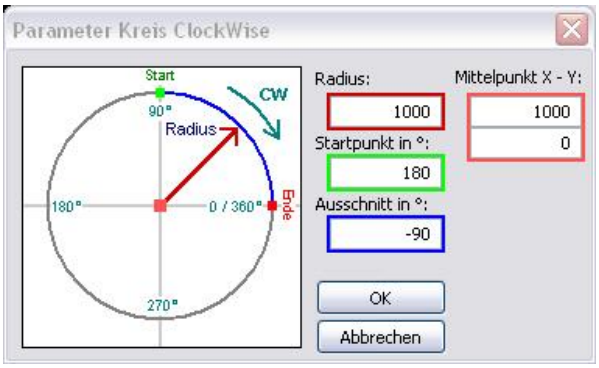
Beispiel: 1I1A;2R → Modul 1, Achse 1 absolut, Achse 2 relativ

Antwort: <STX><ACK>:CS <ETX>(NUR PC)

Zirkularinterpolation



gegen Uhrzeigersinn (CCW)



im Uhrzeigersinn (CW)

x**KR**n — Den Radius n des Kreisbogens für Indexermodul x setzen,
Einheit und Faktor von n sind in P2 und P3 definiert (siehe Kap. 9)

x**KS**n — Den Startpunkt n auf der Kreisbahn für Indexermodul x in Grad (°) setzen
n =0 bis 360°

x**KW**n — Den Weg (Ausschnitt) n in Grad (°) vom Startpunkt für Indexermodul x setzen
n =0 bis 360° (CCW)
n =0 bis -360° (CW)

Wichtig: Die 3 Befehle in **1** Programmzeile schreiben!

Beispiel: 1KR100 1KS90 1KW180

x**KG**a;b — Achsenzuordnung der Indexerkarte x setzen,

a= Masterachse (1,2 oder 3)

b=Slaveachse (1,2,3 oder 4)

x**KT**a:b — Die Teiler von Achse a und Achse b des Indexermoduls x setzen (für Ellipsenlauf)

a: Teiler für Achse 1
b: Teiler für Achse 2

6.13 Registersatz

6.13.1 Laden

Befehl	***Bedeutung***
LRname	Registersatz name wird ins Register-RAM geladen.
LTRname	Textregistersatz name wird ins Textregister-RAM geladen.
LR=Rnn	Registername vom Speicher laden nn→Registernummer von 1 bis 1000
LTR=Rnn	Textregistername vom Speicher laden nn→Registernummer von 1 bis 1000
LR=TDx	Den Namen des Registernsatzes zum Laden aus der Drop Down Box x holen
LTR=TDx	Den Namen des Textregistersatzes zum Laden aus der Drop Down Box x holen

6.13.2 Speichern

Befehl	***Bedeutung***
SRname**;**x**-**y	Registersatz wird auf dem Flashspeicher unter *name* gespeichert. x = Startregister y = Endregister y,x von 1 bis 1000 **Beispiel: SRtest;1-100** Die Register 1 bis 100 werden unter dem Namen *test* auf dem Flash gespeichert.
STRname**;**x**-**y	Textregistersatz wird auf dem Flashspeicher unter *name* gespeichert. x = erstes Textregister y = letztes Textregister y,x von 1 bis 100 à 40 Zeichen
SR=Rnn**;**a**-**e	Registersatz wird auf dem Flashspeicher unter Namen aus Register nn gespeichert. a = Startregister e = Endregister nn → von 1 bis 1000 **Beispiel: SR=R500;1-100** Die Register 1 bis 100 werden unter dem Namen aus Register 500 auf dem Flash gespeichert.

Befehl	***Bedeutung***
STR=Rnn**;**a**-**e	Textregistersatz wird auf dem Flashspeicher unter Namen aus Register nn gespeichert. a = erstes Register e = letztes Register nn → von 1 bis 1000

6.14 Sprungbefehle (NUR PROG)

Befehl	***Bedeutung***
	Sprungbefehle absolut
N* la****	Absoluter Sprung zur Zeile, die durch das Label *la* gekennzeichnet ist.
goto * la****	
	Bedingter Sprung absolut E = Bedingung erfüllt
NE* la****	Absoluter Sprung zur Zeile, die durch das Label *la* gekennzeichnet ist.
	Bedingter Sprung absolut N = Bedingung nicht erfüllt
NN* la****	Absoluter Sprung zur Zeile, die durch das Label *la**** gekennzeichnet ist.

6.15 Passwort (nur PC)

Befehl	***Bedeutung***
PA␣	Die Steuerung wird freigeschaltet, wenn kein Passwort vergeben ist. Dann ist eine Sperrung der Steuerung auch nicht möglich.
PAname	Schaltet die passwortgeschützte Steuerung frei.
PSname	Dieser Befehl vergibt ein Passwort für die Steuerung. Es besteht aus maximal 8 Stellen.
	Freigabezustand für Programme, Parameter und Register
PWSp	Freigabezustand setzen Bei einer passwortgeschützten Steuerung können deren Programme, Parameter und Register freigegeben oder gesperrt werden. p → Freigabezustand für Programm, Parameter und Register p = 0 alle freigegeben p = 1 Programm R/W gesperrt p = 2 Parameter R/W gesperrt p = 4 Register R/W gesperrt p kann zwischen 0 und 7 liegen Beispiel: PWS**5** Programm und Register gesperrt (1+4=5)

Befehl	***Bedeutung***
PWR	Freigabezustand lesen Die Antwort ist eine zweistellige Zahlenkombination. **<ACK> sp:CS <ETX>** s → Freigabezustand der Steuerung s = 0 Steuerung gesperrt s = 1 Steuerung frei p → Freigabezustand für Programm, Parameter und Register siehe oben **Beispiel:** **PWR** **<ACK> 15:CS <ETX>** s = 1 → Steuerung frei p = 5 → Programm und Register gesperrt (1+4=5)

6.16 Programmaufruf beenden oder unterbrechen (NUR PROG)

Befehl	***Bedeutung***
PE	Bei diesem Befehl im Programm wird das Programm beendet und auf einen neuen Wechsel des Schalters **REMOTE/LOCAL** gewartet. Beim Programmstart über Rechner wird in den **Rechnerbetrieb** zurück gesprungen.

6.17 Programm- und Dateiverwaltung (NUR PC)

Befehl	***Bedeutung***
	Programme und Dateien löschen
QDA*.*	Es werden alle Achsenparameter auf Defaultwerte gesetzt.
QDP*.*	Es werden alle Programme, Register- und Textregistersätze im Flash Speicher gelöscht.
QDPname	Es wird das Programm *name* im Flash Speicher gelöscht.
QDR oder **QDR*.***	Es werden alle Register im RAM auf null gesetzt.
QDRname	Es wird der Registersatz *name* im Flash Speicher gelöscht.
QDTRname	Es wird der Textregistersatz *name* im Flash Speicher gelöscht.
QDTR*.*	Es werden alle Textregister im RAM gelöscht.
	Programmstart
QPname **A**	Programm *name* wird gestartet.

Befehl	***Bedeutung***
	Programmabbruch
QPE	Wird der QPE Befehl vom Rechner gesendet, dann wird das aktuell laufende Programm beendet.
	Programmübertragung mit Abfrage
QPname **S**byte	Das Programm mit dem Namen *name* kann nur blockweise übertragen werden. Bei der Übertragung des Programms muss die gesamte Steuersequenz eingehalten werden. name → maximal 16 Zeichen, byte → Anzahl der zu übertragenden Bytes 1. vom Rechner: **<STX>SteuerungsadresseQPname S**byte**:CS <ETX>** Die Programmübertragungssequenz wird gestartet. 2. Antwort der Steuerung: **STX><ACK>O:CS<ETX>** wenn das Programm nicht auf der Steuerung vorhanden ist und das Programm in den Flash-Speicher der Steuerung passt. **<STX><ACK>V:CS<ETX>** K = Flash voll V = existent wenn das Programm auf der Steuerung vorhanden ist und überschrieben werden muss: Wie wird überschrieben: Die Antwort zu “V” ist: **<STX>Adresse J:CS<ETX>** für Überschreiben oder **<STX>N:CS<ETX>** für Stopp (Ende) 3. Programmübertragung: • Start: **<STX> Adresse block 1:CS<ETX>** Block 1 = 256 Byte lang und beginnt mit <ETB>Programmname, wobei Programmname 16 Zeichen lang sein muss! • Weitere Blöcke (Block 1+x) müssen immer 256 Byte lang sein und sind in <STX><ETX> eingeschlossen. **<STX>Adresse block 1+x:CS<ETX>** Antwort der Steuerung nach jedem Block: **<STX><ACK>:CS<ETX>**

Befehl	***Bedeutung***
	Programm lesen mit Abfrage
QPname **R**	Das Programm *name* soll aus der Steuerung ausgelesen werden. Das Programm wird blockweise gelesen.
	Abfrage und senden
	1.vom Rechner:
	<STX>Adresse QPname **R:CS<ETX>** Das Programm *name* soll gelesen werden.
	2. von der Steuerung:
	<STX><ACK>O:CS<ETX> Ist das Programm verfügbar, meldet die Steuerung ein O.
	3. vom Rechner:
	<STX> Adresse J:CS<ETX> Der Rechner empfängt den ersten Block von der Steuerung.
	4. von der Steuerung:
	<STX>Daten Programm Block x:CS<ETX>
	Punkt 3. und 4. wird so lange wiederholt bis alle Zeilen empfangen wurden. An die letzte Zeile sind 0x04(EOT) angehängt.
	Beispiel letzte Zeile:
	<STX> Daten letzter Block:CS<EXT>

6.18 Register

- Die *phy***MOTION**® Steuerungen enthalten 1100 Speicherplätze zur Eingabe von Variablen, die in *phy***LOGIC**® Programmen **Register** genannt werden.
- Die Register werden mit R1 bis R1100 bezeichnet.
  In jedes Register können Zahlen vom Typ double geschrieben werden.
- Register beschreiben: **R**nn**=**zz
  Register auslesen: **R**nn**R**

Erklärungen:		
	R	Kennbuchstabe Register
	nn	Registernummer
	S oder **=**	Schreiben
	zz	double oder int

  Im Programm können Register zur indirekten Eingabe von Positionen verwendet werden. In Verbindung mit Rechenoperationen sind die Register für Zählerfunktionen während des Programmablaufs einsetzbar.
- Der Inhalt von Register R1 bis R1000 bleibt nach einem Reset gespeichert. Der Inhalt von R1001 bis R1100 wird nach einem Reset auf Null gesetzt.
- R1001 bis R1100 ermöglichen einen schnelleren Zugriff nur durch den Prozessor (keine Peripherie).
- Indirekte Zuweisung
  Die Register erlauben auch eine indirekte Zuweisung. Das bedeutet, dass der Inhalt eines Registers auf die Adresse eines anderen Registers zeigt, auf die zugegriffen werden soll.
  Eine Anwendung ist z.B. das Vergleichen von Zahlen:
  Vergleiche den Inhalt des Registers R999 mit dem Inhalt des Registerinhalts von R1000:
  R[R999]==R1000.

  Für alle Verknüpfungen und Rechenoperationen mit Registern gilt:
  Der ermittelte Wert wird immer in das zuerst aufgeführte Register geschrieben.

  Beispiel: Addition der Werte aus zwei Registern
  R18+R2 Der Wert aus Register 2 wird zum Wert aus Register 18 addiert. Das Ergebnis wird in Register 18 abgelegt.

Vergleichsoperationen mit Registern

Als Ergebnis eines Vergleichs wird ein Bedingungsbyte gesetzt:
E = Bedingung erfüllt,
N = Bedingung nicht erfüllt.
Das Bedingungsbyte kann z.B. für bedingte Sprünge oder andere Aktionen ausgewertet werden.

Beispiel: R999==1 NE*eins* N*la*	Vergleich Registerwert mit Zahl und bedingter Sprung. Wenn Register 999 den Wert 1 enthält, springe zu Label*eins*, wenn nicht, springe zu Label *la*.

6.19 Registerbefehle

Rnn=Wert	Wert ins Register schreiben.
Rnn==Wert	Vergleiche den Registerwert und setze Bedingung.
Rnn!=Wert	Vergleiche den Registerwert und setze Bedingung.
Rnn>=Wert oder Rnn>Wert	Vergleiche den Registerwert und setze Bedingung.
Rnn<=Wert oder Rnn<Wert	Vergleiche den Registerwert und setze Bedingung.
Rnn++	Erhöhe den Inhalt um 1
Rnn--	Vermindere den Inhalt um 1

Befehl	***Bedeutung***
	Registerinhalt Achsen
Rnn**SMA** oder **R**nn**=MA**	Anzahl der Achsenmodule lesen.
Rnn**SMA**x oder **R**nn**=MA**x	Anzahl der Achsen pro Modul lesen. x=Modul → 1 bis n
Rnn**SMAI** oder **R**nn**=MAI**	Anzahl der Analog-Eingangsmodule lesen.
Rnn**SMAIO** oder **R**nn**=MAIO**	Anzahl der Analog-Ein-und Ausgangsmodule lesen.
Rnn**SMAO** oder **R**nn**=MAO**	Anzahl der Analog-Ausgangsmodule lesen.
Rnn**SMDI** oder **R**nn**=MDI**	Anzahl der Digital-Eingangsmodule lesen.
Rnn**SMDIO** oder **R**nn**=MDIO**	Anzahl der Digital-Ein-und Ausgangsmodule lesen.
Rnn**SMDO** oder **R**nn**=MDO**	Anzahl der Digital-Ausgangsmodule lesen.
	Registerinhalt ganzzahlig
Rnn.z	Die Nachkommastellen des Registerinhalts nn löschen ohne Rundung des Wertes. z = 0 – 6 Nachkommastellen

Befehl	***Bedeutung***
	Register binär schreiben (über Eingänge)
Rnn**BE**m**.**n**–**m**.**x	Die Eingänge n bis x des Moduls m in den Inhalt des Registers nn binär schreiben. **Beispiel: R1BE1.1–1.8 →** Eingang Status: **1010 0101** Ergebnis: **165**
	Register hexadezimal beschreiben
Rnn**BS**zwert	Den Inhalt des Registers nn binär mit dem Wert zwert beschreiben. Die Daten werden hexadezimal eingegeben. **Beispiel: R1BS1FA** Das Register 1 wird mit dem Hexadezimalwert 1FA beschrieben. Nach dem Befehl hat das Register 1 den Wert 506 dezimal.
	Registerinhalt Bit weise schieben
Rnn**BL**m	Der Inhalt des Registers nn wird binär um m Stellen nach links (MSB ←) geschoben. Es wird rechts mit 0 aufgefüllt. m = 1 bis 31 → Maximalwert des Registerinhaltes **Beispiel: R1S168 R1BL2** Das Register 1 hat den Dezimalwert 168, das entspricht dem Binärwert **10101000**. Nach dem Befehl **R1BL2** ist der Binärwert **1010100000**, das entspricht dem Dezimalwert 672. Der Inhalt des Registers 1 ist um 2 Stellen binär nach links geschoben worden. Das Register 1 hat nun den Wert 672. **Antwort: <STX><ACK>:CS<ETX>**
Rnn**BR**m	Der Inhalt des Registers nn wird binär um m Stellen nach rechts (→ LSB) geschoben. Es wird links mit 0 aufgefüllt. m = 1 bis 31 → Maximalwert des Registerinhaltes **Beispiel: R1S168 R1BR2** Das Register 1 hat den Dezimalwert 168, das entspricht dem Binärwert **10101000**. Nach dem Befehl **R1BR2** ist der Binärwert **101010**, das entspricht dem Dezimalwert 42. Der Inhalt des Registers 1 ist um 2 Stellen binär nach rechts geschoben worden. Das Register 1 hat nun den Wert 42.
	Register Bit testen
Rnn**BT**m	Der Inhalt des Registers nn wird in einen Binärwert gewandelt. Nun wird das Bit an der Stelle m im Register überprüft. Ist das Bit = 1 wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt. m = 0 bis 31 → Maximalwert des Registerinhaltes

Befehl	***Bedeutung***
	Beispiel: R1S168 R1BT4 Das Register 1 hat den Binärwert **10101000**. Mit dem Befehl **R1BT4** wird das 4. Bit des Binärwertes von rechts (m ← LSB) getestet. Nun wird das Bedingungsbyte rückgesetzt, da das 4. Bit auf 0 steht. **Antwort:** bei PC: **<STX><ACK> E:CS <ETX>** oder **<STX><ACK> N:CS <ETX>** bei PROG: Bedingungsbyte wird gesetzt oder rückgesetzt
Rnn**B**x**==**1 **R**nn**B**x**==**0	Der Inhalt des Registers nn wird auf ‚1' oder ‚0' verglichen und das Bedingungsbyte gesetzt (erfüllt oder nicht erfüllt). x = 0 bis 31 → Bitposition
Rnn**B**x**!=**1 **R**nn**B**x**!=**0	Der Inhalt des Registers nn wird auf ‚1' oder ‚0' verglichen und das Bedingungsbyte gesetzt (erfüllt oder nicht erfüllt). x = 0 bis 31 → Bitposition
	Register logische Verknüpfung
	Und Verknüpfung
Rnn**B^**zwert	Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert **UND** verknüpfen. **Beispiel: R1BS2A8 R1B^1A0** Das Register 1 wird mit dem Hexadezimalwert 2A8 beschrieben. Der Inhalt des Registers ist nun 680 dezimal. Nach dem Befehl **R1B^1A0** hat das Register 1 den Wert 160 dezimal.

	Dezimal	**Hex**	**Binär**
	680	**2A8**	**1010101000**
	416	**1A0**	**0110100000**
Ergebnis:	**160**	**0A0**	**0010100000**

Rnn**B^R**mm	Den Inhalt des Registers nn mit dem Inhalt des Registers mm **UND** verknüpfen.
	ODER Verknüpfung
Rnn**Bv**zwert	Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert **ODER** verknüpfen. **Beispiel: R1BS2A8 R1Bv1A0** Das Register 1 wird mit dem Hexadezimalwert 2A8 beschrieben. Der Inhalt des Registers ist nun 680 dezimal. Nach dem Befehl **R1Bv1A0** hat das Register 1 den Wert 936 dezimal.

Befehl	***Bedeutung***

	Dezimal	**Hex**	**Binär**
	680	**2A8**	**1010101000**
	416	**1A0**	**0110100000**
Ergebnis:	**936**	**3A8**	**1110101000**

Rnn**BvR**mm — Den Inhalt des Registers nn mit dem Inhalt des Registers mm **ODER** verknüpfen.

Antwort: <STX><ACK>:CS<ETX> (NUR PC)

XOR Verknüpfung

Rnn**BX**zwert — Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert **XOR** verknüpfen.

Beispiel: R1BS2A8 R1BX1A0
Das Register 1 wird mit dem Hexadezimalwert **2A8** beschrieben. Der Inhalt des Registers ist nun 680 dezimal. Nach dem Befehl **R1BX1A0** hat das Register 1 den Wert 776 dezimal.

	Dezimal	**Hex**	**Binär**
	680	**2A8**	**1010101000**
	416	**1A0**	**0110100000**
Ergebnis:	**776**	**308**	**1100001000**

Rnn**BXR**mm — Den Inhalt des Registers nn mit dem Inhalt des Registers mm **XOR** verknüpfen.

Registerinhalte vergleichen (mit einem Zahlenwert)

Rnn**==**zwert — Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis gleich ist, sonst wird es rückgesetzt.

Rnn**!=**zwert — Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis ungleich ist, sonst wird es rückgesetzt.

Rnn**>**zwert
Rnn**>=**zwert — Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis größer oder gleich ist, sonst wird es rückgesetzt.

Rnn**<**zwert
Rnn**<=**zwert — Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner oder gleich ist, sonst wird es rückgesetzt.

Befehl	***Bedeutung***
	Registerinhalte miteinander vergleichen
Rnn**==R**mm	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis gleich ist, sonst wird es rückgesetzt.
Rnn**!=R**mm	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis ungleich ist, sonst wird es rückgesetzt.
Rnn**>R**mm **R**nn**>=R**mm	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis größer oder gleich ist, sonst wird es rückgesetzt.
Rnn**<R**mm **R**nn**<=R**mm	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner oder gleich ist, sonst wird es rückgesetzt.
	Antwort bei allen Vergleichen: bei PC: **<STX><ACK> E:CS <ETX>** oder **<STX><ACK> N:CS <ETX>** bei PROG: Bedingungsbyte wird gesetzt
Rnn**B**x**=1**	Das Bit x im Registers nn wird gesetzt. x = 0 bis 31 → Bitposition
Rnn**B**x**=0**	Das Bit x im Registers nn wird mit zurückgesetzt. x = 0 bis 31 → Bitposition
	Rechenoperationen mit Registern
	Addieren
Rnn**+**zwert	Zum Inhalt des Registers nn wird der Wert zwert addiert.
Rnn**+R**mm	Zum Inhalt des Registers nn wird der Inhalt des Registers mm addiert.
Rnn**++**	Der Inhalt des Registers nn wird um „1“ erhöht.
	Subtrahieren
Rnn**–**zwert	Vom Inhalt des Registers nn wird der Wert zwert subtrahiert.
Rnn**–R**mm	Vom Inhalt des Registers nn wird der Inhalt des Registers mm subtrahiert.
Rnn**– –**	Der Inhalt des Registers nn wird um „1“ vermindert.

Befehl	***Bedeutung***
	Multiplizieren
Rnn*****zwert	Der Inhalt des Registers nn wird mit dem Wert zwert multipliziert.
Rnn***R**mm	Der Inhalt des Registers nn wird mit dem Inhalt des Registers mm multipliziert.
	Dividieren
Rnn**:R**mm	Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.
Rnn**/R**mm	Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.
	Winkelfunktionen
Rnn**SIN** **R**nn**COS** **R**nn**TAN**	Vom Wert des Registers nn wird Sinus, Cosinus oder Tangens berechnet und das Ergebnis ins Register nn zurückgeschrieben.
Rnn**ASIN** **R**nn**ACOS** **R**nn**ATAN**	Vom Wert des Registers nn wird der Winkel von Sinus, Cosinus oder Tangens berechnet und das Ergebnis ins Register nn zurückgeschrieben.
	Wurzelfunktion
Rnn**QW**	Aus dem Wert des Registers nn wird die Quadratwurzel berechnet und in das Register nn zurückgeschrieben.
	Potenzfunktion
Rnn**E**zwert	Der Wert des Registers nn wird potenziert und in das Register nn zurückgeschrieben.
	Zufallszahl
Rnn**RAND**	Das Register nn wird mit Zufallszahl im Bereich 0 bis 4294967296 (2^{32}) beschrieben.
	Register auslesen
Rnn**R**	Der Inhalt des Registers nn wird ausgelesen. (NUR PC) **Antwort: <STX><ACK>zwert:CS<ETX>** **Antwort bei allen Rechenoperationen:** **<STX><ACK>:CS<ETX>** (NUR PC)
	Register beschreiben: mit Dezimalwerten
Rnn**S**zwert oder **R**nn**=**zwert	Das Register nn mit dem Zahlenwert zwert beschreiben.

Befehl	***Bedeutung***
	mit Registerinhalten
Rnn**SR**mm oder **R**nn**=R**mm	Das Register nn mit dem Inhalt des Registers mm beschreiben.
	mit Parameterwerten
Rnn**S**m.a**P**y oder **R**nn**=**m.a**P**y	Das Register nn mit dem Wert des Parameters y des Moduls m der Achse a beschreiben.
	mit dem Zahlenwert der Bedienterminaleingabe
Rnn**ST** oder **R**nn**=T**	Das Register nn mit der Werteingabe des Bedienterminals BT5 AM beschreiben.
	mit dem Temperatur-Auswertungs-Modul T4K
Rnn**STK**m**.**n oder **R**nn**=TK**m**.**n	Temperatur des Moduls m am Kanal n lesen
Rnn**STK**m**.**n**c** oder **R**nn**=TK**m**.**n**c**	Temperaturtyp des Moduls m am Kanal n lesen
Rnn**STK**m**.**n**T** oder **R**nn**=TK**m**.**n**T**	Zyklen für die Mittelwertbildung des des Moduls m am Kanal n lesen
	mit dem Impulswert
Rnn**STT** oder **R**nn**=TT**	Das Register nn mit dem Impulswert beschreiben. 1 Impulswert = 0,1 msec
	Über Eingänge
Rnn**SE**m.n-m.x oder **R**nn**=E**m.n-m.x	Das Register nn mit einem BCD Wert über die Eingänge m.n bis m.x beschreiben. Der Wert wird mit k Nachkommastellen ins Register geschrieben.
	Beispiel: R1SE1–8.1
	Die Eingänge 1 bis 8 haben z.B. den Status: **1001 0011**. Das Ergebnis ist 9.3.
Rnn**SEC**m oder **R**nn**=EC**m	Zählerwert der Eingangskarte m ins Register n schreiben.
Rnn**=TBA**	Das Register nn wird mit der Information geladen, welche Taste betätigt wurde. Bit 0 = 1→ Taste 1 Bit 31 = 1 → Taste 32
Rnn**=TCA**	Das Register nn wird mit der Information geladen, welcher Schalter betätigt wurde. Bit 0 = 1→ Schalter 1 Bit 31 = 1 → Schalter 32

6.20 Systemstatus (NUR PC)

Befehl	***Bedeutung***
	Systemstatus Allgemein
S	Achsentest mit Ausgabe der Anzahl der Achsen. **Antwort:<STX><ACK>n IO:CS <ETX>** n = Anzahl der Achsen
	Systemstatus erweitert (für I4XM01 oder I1AM01)
SEm.n	Den Status der Steuerung auslesen. m = Modulnummer; n = Achsennummer **Antwort:** **<STX><ACK>?????$d_H d_H d_H d_H d_H d_H d_H d_H$:CS<ETX>** Die Antwort kommt im **Dezimal**code. Der Achsenstatus als Hexadezimalcode konvertiert lautet: 00000001 = Achse beschäftigt 00000002 = Befehl ungültig 00000004 = Achse wartet auf Synchronisation 00000008 = Achse initialisiert 00000010 = Achse Endschalter + 00000020 = Achse Endschalter – 00000040 = Achse Endschalter Mitte 00000080 = Achse Software-Endschalter + 00000100 = Achse Software-Endschalter – 00000200 = Achse Endstufe bereit 00000400 = Achse befindet sich in der Rampe 00000800 = Achse interner Fehler 00001000 = Achse Endschalter Fehler 00002000 = Achse Endstufenfehler 00004000 = Achse SFI Fehler 00008000 = Achse ENDAT Fehler 00010000 = Achse läuft 00020000 = Achse ist in der Beruhigungszeit (s. Parameter P13 oder P16) 00040000 = Achse ist in der Stoppstromüberhöhungszeit (Parameter P43) 00080000 = Achse ist in Position 00100000 = Achse APS betriebsbereit 00200000 = Achse Positionier-Modus 00400000 = Achse Freier Lauf-Modus 00800000 = Achse Multi F Lauf 01000000 = Achse SYNC erlaubt

Befehl	***Bedeutung***
SEC	Status aller Module zurücksetzen.
SECm.a	Status des IDX bzw. I1AM Moduls zurücksetzen. m = Modulnummer a = Achsennummer
	Systemstatus Achsen
SGn	Achsenkurzstatus der Achsenkarte auslesen n → Nummer der Achsenkarte **Antwort:** **<STX><ACK>$d_H d_H d_H d_H d_H d_H d_H d_H$:CS<ETX>** Die Antwort kommt im **Dezimal**code. Den Achsenstatus in den Hexadezimal Code konvertieren. **Beispiel:** **SG1** **<STX><ACK>00000005:CS<ETX>** Achse 1 und 3 der Achsenkarte 1 (hier: I4X-Modul) sind aktiv.

Modul	Byte	3		2		1		0	
	Bit	7	6	5	4	3	2	1	0
	Status	–	–	Pre-Register	Inter-polation	End-schalter –	End-schalter +	Achse Fehler	Achse beschäftigt
I4X	Achse 1 gesetzt	0	0	1	1	1	1	1	1
	Achse 2 gesetzt	0	0	2	2	2	2	2	2
	Achse 3 gesetzt	0	0	4	4	4	4	4	4
	Achse 4 gesetzt	0	0	8	8	8	8	8	8
	Summe der 4 Achsen	0	0	F	F	F	F	F	F
I1AM	Achse 1 gesetzt	0	0	0	0	1	1	1	1

SH **SH==1** **SH!=1** **SH!=E** **SH!=N**	Achsentest mit Ausgabe, ob Achsen stehen. **Antwort:<STX><ACK> E :CS <ETX>, wenn alle Achsen stehen.** **<STX><ACK> N :CS <ETX>, wenn eine Achse läuft.**

Systemstatus dezimal

ST Den Status der Steuerung auslesen. Der Status wird in einer Dezimalzahl ausgegeben.

Antwort : <STX><ACK>zwert :CS<ETX>
zwert = Nummer zwischen 0 und FFFF

Die Antwort kommt im **Dezimal**code. Der Achsenstatus als Hexadezimalcode konvertiert lautet:

00000000	= Programmende im LOCAL-Betrieb.
00000001	= Programmausführung
00000002	= Software Remote
00000004	= Endschalter einer Achse
00000008	= Endstufenfehler einer Achse
00000010	= Programmierfehler (falscher Befehlscode oder nicht erlaubt)
00000020	= Terminal aktiviert
00000040	= Eingangsscan läuft
00000080	= Remote
00000100	= Achsenmodul nicht verfügbar
00000200	= Achse nicht verfügbar
00000400	= I/O Modul nicht verfügbar
00000800	= I/O nicht verfügbar
00001000	= Interner Busfehler während Datentransfer
00002000	= Fehler auf einer Karte auf internem Bus
00004000	= AIOM Modul nicht verfügbar
00008000	= AIOM Kanal nicht verfügbar

Status Reset

STC Status ist zurückgesetzt.

6.21 Synchronstart

Befehl	***Bedeutung***
S1	Synchronstart der Achsen vorbereiten
S0	Synchronstart der Achsen ausführen
SC	Synchronstart Abbruch

6.22 Daten ins Flash EPROM schreiben

Befehl	***Bedeutung***
	Achsenparameter speichern (NUR PC)
SA	Achsenparameter auf das Flash EPROM schreiben. ab Version V1.1.13 gilt: die Achsenparameter werden auch auf der externen Endstufe bei ServiceBus-Anschluss gespeichert: MSD2$^+$, MCD2$^+$,…
SAE	Die Achsenparameter (P40 bis P45) auf die Endstufe schreiben.

6.23 IO Status

Befehl	***Bedeutung***
SAIOCm	Status des AIOM Moduls zurücksetzen. m = Modulnummer
SAIOm	Status des AIOM Moduls lesen m = Modulnummer **Antwort:** [Bit0] =1: Fehler [Bit16]=1: DAC-Fault [Bit17]=1: ISO_PWR-Error [Bit18]=1: Record-Mode [Bit19]=1: Record-Busy [Bit20]=1: Record-Memory-Half-Full [Bit21]=1: Record-Memory-Full Bit16 und Bit17 erzeugen einen ERROR in Bit0
SIOCm	Status des DIOM Moduls zurücksetzen. m = Modulnummer
SIOm	Status des DIOM Moduls lesen m = Modulnummer Antwort: [Bit0] =1: Fehler [Bit16]=1: Output-Driver-Error [Bit17]=1: ISO_PWR-Error Bit16 und Bit17 erzeugen einen ERROR in Bit0
SPIDm	Status des PID Reglermoduls m lesen Antwort: [Bit0]=1: Fehler [Bit16]=1: Ausgangstreiber-Fehler [Bit17]=1: 24 V_{DC} Spannungsfehler [Bit18]=1: Fehler im PID Messeingang 1 [Bit19]=1: Fehler im PID Messeingang 2

Befehl	***Bedeutung***
	[Bit20]=1: Fehler im PID Messeingang 3 [Bit21]=1: Fehler im PID Messeingang 4
SPIDCm	Status des PID Reglermoduls m zurücksetzen

6.24 Touch Panel

6.24.1 Touch Panel Modus (NUR PROG)

> **i** Die Bedienoberfläche des Touch Panel ist nur im Passiv-Modus designbar!
>
> Design-Elemente sind: Schalter, Taster: je 32 Stück
> Label und andere Elemente: je 50 Stück
>
> x → Nummer des Design-Elements; x=1 bis 32 bzw. 50

Befehl	***Bedeutung***
TA0	Das Touch Panel vom Passiv-Modus in den Aktiv-Modus umschalten
TA?	Den Modus des Touch Panels abfragen 0 → aktiv 1 → passiv
TA=1	Das MCM Modul fordert das Touch Panel auf, in den Passiv Modus umzuschalten
TA==1	Abfrage: Passiv-Modus aktiv ?
TA!=1	Abfragen: Aktiv-Modus aktiv ?

6.24.2 Touch Panel - Taster (Button)

Befehl	***Bedeutung***
TBA?	Information von allen Tastern abfragen
TBx**==1** **TB**x**!=1**	Zustand des Tasters x abfragen und Bedingung setzen x → Nummer des Elements (max. 32)

Befehl	***Bedeutung***
TBCx=xPos;yPos;xGröße; yGröße;Schriftgröße;Ausrichtung	Taster x definieren xPos;yPos: Position des Tasters in x und y-Richtung xGröße;yGröße; Größe des Tasters in Pixel Schriftgröße in Pixel Ausrichtung der Taster-Beschriftung 0 → zentriert 1 → links 2 → rechts
TBDx	Taster x löschen
TBEx=n	Tasterfunktion erlauben x → Taster n → Funktion n=0 bis 2 0 → Taster nicht sichtbar (keine Funktion) 1 → Tasterfunktion sichtbar (keine Funktion) 2 → Tasterfunktion sichtbar und aktiv
TBFx**"**rrggbb**"**	RGB Farbe für Taster x setzen rrggbb → RGB Farbcode (6 Stellen in hexadezimal)
TBTx**"**text**"** **TBT**x**TR**n	Taster mit ‚text' beschriften Taster mit Inhalt des Textregisters beschriften (6 Stellen in hexadezimal)
TB?x	Status des Taster lesen 0 → Taster nicht gedrückt 1 → Taster gedrückt

6.24.3 Touch Panel - Schalter

Befehl	***Bedeutung***
TCA?	Information von allen Schaltern abfragen
TCA=Rxx	Zustand aller Schalter setzen
TCx**==1** **TC**x**!=1**	Zustand des Schalters x abfragen und Bedingung setzen

Befehl	***Bedeutung***
TCCx**=** xPos;yPos;xGröße;yGröße; Schriftgröße;Ausrichtung	Schalter x definieren xPos;yPos: Position des Tasters in x und y-Richtung xGröße;yGröße; Größe des Tasters in Pixel Schriftgröße in Pixel Ausrichtung: 0 → zentriert 1 → links 2 →rechts
TCDx	Schalter x löschen
TCEx**=**n	Funktion des Schalters x setzen x → Schalter n=0 bis 2 0 → Schalter nicht sichtbar (keine Funktion) 1 → Schalterfunktion sichtbar (keine Funktion) 2 → Schalterfunktion sichtbar und aktiv
TCFONx**“** rrggbb**“**	rrggbb RGB-Farbe setzen für Schalter x „gedrückt“
TCFOFFx**“** rrggbb**“**	rrggbb RGB-Farbe setzen für Schalter x„nicht gedrückt“
TCTONx**“**text**“** **TCTON**x**TR**n	Schalter x „gedrückt“ mit „text“ beschriften Schalter x „gedrückt“ mit Inhalt des Textregisters beschriften
TCTOFFx**“**text**“** **TCTOFF**x**TR**n	Schalter x „nicht gedrückt“ mit „text“ beschriften Schalter x „nicht gedrückt“ mit Inhalt des Textregisters beschriften
TC?x	Status des Schalter x lesen 0 → Schalter nicht gedrückt 1 → Schalter gedrückt
TCSx	Schalter x auf „gedrückt“ setzen
TCRx	Schalter x auf „nicht gedrückt“ setzen

6.24.4 Touch Panel – Drop Down Box

Befehl	***Bedeutung***
TDCx**=** xPos;yPos;xGröße;yGröße; Schriftgröße;Ausrichtung	Drop Down Box x definieren xPos;yPos: Position der Drop Down Box in x und y-Richtung xGröße;yGröße; Größe der Drop Down Box in Pixel Schriftgröße in Pixel Ausrichtung: 0 → zentriert 1 → links 2 → rechts
TDDx	Drop Down Box x löschen
TDGx	Aktiven Text x der Drop Down Box lesen
TDPx	Aktive Position x des Textes der Drop Down Box lesen
TDSx**=**n	Text der Position n der Drop Down Box als aktiv setzen
TDTx“text“ **TDT**x**TR**nn **TDT**x**R**nn	Drop Down Box x mit „text“ beschreiben Drop Down Box x mit Inhalt des Textregisters beschreiben Drop Down Box x mit Inhalt des Registers beschreiben
TDTxIR	Drop Down Box x mit directory der Register füllen
TDTxITR	Drop Down Box x mit directory der Textregister füllen

6.24.5 Touch Panel - Text anzeigen

Befehl	***Bedeutung***
TLCx**=** xPos;yPos;xGröße;yGröße; Schriftgröße;Ausrichtung	Label x definieren xPos;yPos: Position des Labels in x und y-Richtung xGröße;yGröße; Größe des Labels in Pixel Schriftgröße in Pixel Ausrichtung: 0 → zentriert 1 → links 2 → rechts
TLDx	Label x löschen
TLFBKx“rrggbb“	RGB Farbe für Hintergrund x setzen rrggbb → RGB Farbcode (6 Stellen) default: FFFFFF

Befehl	***Bedeutung***
TLFTXx“rrggbb“	RGB Farbe für Schrift x setzen rrggbb → RGB Farbcode (6 Stellen) default:000000
TLTx**“**text**“** **TLT**x**TR**n	Label x mit “text”‘ beschreiben Label x mit Inhalt des Textregisters n beschreiben (6 Stellen in hexadezimal)

6.24.6 Touch Panel - Eingabefeld

Befehl	***Bedeutung***
TECx**=** xPos;yPos;xGröße;yGröße;Schriftgröße;Ausrichtung;Funktion	Eingabefeld x definieren xPos;yPos: Position des Eingabefeldes in x und y-Richtung xGröße;yGröße; Größe des Eingabefeldes in Pixel Schriftgröße in Pixel Ausrichtung: 0 → zentriert 1 → links 2 →rechts Funktion: 0 → alphanumerisch 1 → numerisch
TEDx	Eingabefeld x löschen
TETx**“**text**“** **TET**x**TR**nn **TDS**x**=R**nn **TDS**x**=R**[**R**nn]	Ins Eingabefeld x „text“ schreiben Ins Eingabefeld x den Inhalt des Textregisters nn schreiben Ins Eingabefeld x den Inhalt des Registers nn schreiben Ins Eingabefeld x den Inhalt der Registers von Rnn schreiben (mit 3 Nachkommastellen beschreibbar)
TE?x	Status des Eingabefeldes x lesen 0 → nicht gedrückt 1 → gedrückt
TEGx	Den ins Eingabefeld x geschriebenen Text zurückgeben.
LR=TEx	Eingabefeld x lesen und Wert in Register schreiben
LTR=TEx	Registersatz mit Namen vom Eingabefeld x vom Flash laden

6.24.7 Touch Panel – Image (Bild) anzeigen

Befehl	***Bedeutung***
TICx**=** xPos;yPos;xGröße;yGröße	Image x definieren xPos;yPos: Position des Images in x- und y-Richtung (Position 0;0 ist links oben) xGröße;yGröße; Größe des Images in Pixel
TIDx	Image x löschen
TISx“name“	Image x setzen mit Inhalt von „name“ Auswahl von name → LED_GN_OFF LED_GN_ON LED_YE_OFF LED_YE_ON LED_RD_OFF LED_RD_ON

6.24.8 Touch Panel – Fortschrittsbalken (Progressbar) anzeigen

Befehl	***Bedeutung***
TPCx**=** xPos;yPos;xGröße;yGröße; Maximalwert	Progressbar x definieren xPos;yPos: Position der Progressbar in x- und y-Richtung xGröße;yGröße: Größe der Progressbar in Pixel Maximalwert: Endwert der Progressbar
TPDx	Progressbar x löschen
TPPx**=**nn **TPP**x**=R**nn	In Progressbar x Wert nn setzen In Progressbar x den Inhalt des Registers nn setzen

6.24.9 Touch Panel - Slider anzeigen

Befehl	***Bedeutung***
TSCx**=** xPos;yPos;xGröße;yGröße; Maximalwert	Slider x definieren xPos;yPos: Position des Slider in x- und y-Richtung xGröße;yGröße: Größe des Sliders in Pixel Maximalwert: Endwert des Sliders
TSDx	Slider x löschen
TSPx**=**nn **TSP**x**=R**nn **TSP**x**=R**[Rnn]	In Slider x Wert nn setzen In Slider x das Register nn setzen In Slider x den Wert des Registers Rnn setzen

Befehl	***Bedeutung***
TSGx	Slider Wert x lesen

6.24.10 Gruppenbefehl

Befehl	***Bedeutung***
TG= Befehl1|Befehl2|...	Befehlsgruppe senden **Beispiel: TG=TBC1=10;10;50;50;16;0|TBF1“00FF00“** Taster 1 definieren und die Farbe des Tasters setzen. **Wichtig:** gilt **nicht** für Registerwerte!

6.25 Temperatur-Auswertung

Befehl	***Bedeutung***
TKm oder **TK**m**R**	Temperatur auf den 4 Kanälen über die Schnittstelle lesen Antwort **<STX><ACK>wert1;wert2,wert3,wert4:CS<ETX>**
TKm**.**n oder **TK**m**.**n**R**	Temperatur auf einem Kanal n des Moduls m lesen
TKm**.**n**T** oder **TK**m**.**n**TR**	Auf Kanal n des Moduls m den Temperaturmesstyp lesen **<STX><ACK>typ:CS<ETX>** typ=1: B Platinum/Rhodium typ=2: E Constantan typ=3: J Constantan typ=4: K Constantan typ=5: N Nisil typ=6: R Platinum typ=7: S Platinum typ=8: T Constantan
TKm**.**n**T**typ	Kanal n des Moduls m auf den Temperaturmesstyp typ setzen
TKm**.**n**CR**	Auf Kanal n des Moduls m die Anzahl der Lesezyklen auslesen
TKm**.**n**C**wert	Auf Kanal n des Moduls m die Anzahl der Lesezyklen für Mittelwertbildung auslesen wert=1 bis 255

6.26 Bedienterminal

Befehl	***Bedeutung***
TT=y	Terminal-Anschlussart y definieren; y=0 bis 3 0 → kein Terminal 1 → BT5 2 →Touch Panel auf Terminalschnittstelle 3 → Touch Panel auf Feldbusschnittstelle (Ethernet oder RS232)
TT==1 **TT!=1**	Zustand des BT5 Terminals abfragen: **Antwort: <STX><ACK>E:CS<ETX> :** BT5 ist aktiv **oder <STX><ACK>N:CS<ETX> :** BT5 ist nicht aktiv
TTR	Terminaltyp lesen **Antwort: <STX><ACK>y:CS<ETX>** y=0 bis 3 0 → kein Terminal 1 → BT5 2 →Touch Panel auf Terminalschnittstelle 3 → Touch Panel auf Feldbusschnittstelle (Ethernet oder RS232)

6.27 Textregister

Befehl	***Bedeutung***
TRn**“**text**“** **TR**n**=**"text"	Textregister schreiben (text muss innerhalb von „“ stehen) n = Nummer des Textregisters
TRn**R**	Textregister lesen
TRn**C**	Textregister löschen

6.28 Time Loops

Befehl	***Bedeutung***
Tzwert **TR**nn **TR[R**nn**]**	Bei der Zeitschleife wird die Zeit (zwert, Inhalt von Register nn oder Inhalt von Register [Rnn]) in Millisekunden angegeben. Das Programm wartet hier, bis die Zeit abgelaufen ist. **Antwort: <STX><ACK>:CS<ETX>**
TTn**S**zwert **TT**n**SR**nn **TT**n**SR[R**nn**]**	Der Timer n wird mit dem Wert (zwert, Inhalt von Register nn oder Inhalt von Register [Rnn]) geladen. Der Timer zählt den Wert bis auf null und das Programm wird nicht unterbrochen. n=1 bis 10
TTn**==**0	Der Timer n wird mit dem Wert 0 verglichen. Ist der Timerwert gleich 0, dann wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt. Timer = 0 bedeutet, dass die vorgegebene Zeit abgelaufen ist. **Antwort: <STX><ACK>:CS<ETX>**
TTn!=0	Der Timer n wird mit dem Wert 0 verglichen. Ist der Timerwert ungleich 0, dann wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt. **Antwort: <STX><ACK>:CS<ETX>**
TTn**>**zwert **TT**n**>R**nn **TT**n**>R[R**nn**]** **TT**n**<**zwert **TT**n**<R**nn **TT**n**<R[R**nn**]**	Der Timer n wird mit dem Wert zwert, dem Inhalt von Register nn oder Register [Rnn] verglichen. Ist der Timerwert größer/kleiner als der Wert zwert, Inhalt von Register nn oder Register [Rnn] dann wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt. **Antwort: <STX><ACK>E:CS<ETX>** oder **<STX><ACK>N:CS<ETX>**

6.29 Unterprogramme (NUR PROG)

Befehl	***Bedeutung***
	Unterprogramm abbrechen
UA	Alle Unterprogramm-Ebenen werden zurückgesetzt. Das Programm kann mit einem Sprungbefehl fortgesetzt werden.
	Unterprogramm beenden
UE	Das Unterprogramm wird beendet und es wird an der Stelle fortgefahren, an der das Unterprogramm aufgerufen wurde.
	Unterprogramm aufrufen
U* la*	Das Unterprogramm beginnt in der Zeile, die durch das Label ***la*** gekennzeichnet ist. Beendet wird das Unterprogramm mit dem UE Befehl.

Befehl	***Bedeutung***
	Bedingter Unterprogrammaufruf
	Für den bedingten Unterprogrammaufruf stehen alle Befehlsvarianten des unbedingten Unterprogrammaufrufs zur Verfügung. Der Befehlscode wird lediglich ergänzt durch den Buchstaben „E“ für Bedingung erfüllt bzw. „N“ für Bedingung nicht erfüllt.
	„E“ für Bedingung erfüllt
UE*la*	siehe **U***la*, S. 57
UN*la*	siehe **U***la*, S. 57

6.30 Achsenbefehle

Folgende Achsenbefehle **können** auch mit **„M“** beginnend angewendet werden:

Beispiele: Mm.a**R–** entspricht m.a**R–**
Mm.a**MA** entspricht m.a**MA**

Bei der **indirekten (Register)** Adressierung **müssen** die Achsenbefehle mit **„M“** beginnen:

Beispiel: **M**R[x].R[y]**+1000** bedeutet:
- Inhalt von R[x]=Modulnummer
- Inhalt von R[y]=Achsennummer
- Achse R[y] des Moduls R[x] um 1000 in +Richtung fahren

Befehl	***Bedeutung***
m.a**C**	Achse a des Moduls m rücksetzen m= Modul 1 bis n a= Achse 1 bis 4
	Antwort: <STX><ACK>:CS<ETX> (NUR PC)
m.a**GP**	Achse a des Moduls m die Defaultwerte der Parameter schreiben (ohne speichern!) m= Modul 1 bis n a= Achse 1 bis 4
	Achsenzustand abfragen
m.a**==A** m.a**!=A**	Abfrage, ob sich Achse in der Rampe befindet. Das Bedingungsbyte wird gesetzt, wenn sich die Achse in der Rampe (**==**) oder nicht in der Rampe (**!=**) befindet, sonst wird es rückgesetzt.
m.a**==E** m.a**!=E**	Eine Achse auf Endstufenfehler abfragen. Das Bedingungsbyte wird gesetzt, wenn ein Endstufenfehler ansteht (**==**) bzw. **nicht** ansteht (**!=**), sonst wird es rückgesetzt. Es wird der Fehler “Endstufenfehler“ abgefragt
m.a**==H** m.a!**=H**	Eine Achse auf Stillstand abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse steht (**==**) bzw. läuft (**!=**), sonst wird es rückgesetzt.

Befehl	***Bedeutung***
m.a**==I+** m.a**!=I+** m.a**==I–** m.a**!=I–** m.a**==IC** m.a**!=IC** m.a**==IS+** m.a**!=IS+** m.a**==IS–** m.a**!=IS–**	Eine Achse auf Initiatorzustand abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse auf dem Initiator steht oder Initiator nicht angeschlossen ist, sonst wird es rückgesetzt. IC→ Endschalter Center IS+→Software-Endschalter + IS–→Software-Endschalter – I+ →Endschalter+ I–→Endschalter–
m.a**==M** m.a**!=M**	Eine Achse auf Schrittfehler abfragen. Das Bedingungsbyte wird gesetzt, wenn ein Schrittfehler ansteht, sonst wird es rückgesetzt. Der Fehler kann nur bei Steuerungen mit optionaler Encoder-Karte auftreten, da diese eine Schrittfehlerüberwachung haben. Voraussetzung: Der Parameter P36 muss auf 1 gesetzt sein, um die Achse auf Schrittfehler abzufragen.
m.a**==N** m.a**!=N**	Eine Achse auf Nothalt abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse auf einem Nothaltendschalter steht (**==**), sonst wird es rückgesetzt. **Antwort: <STX><ACK>E:CS<ETX>** oder **<STX><ACK>N:CS<ETX>** (NUR PC)
m.a**==R** m.a**!=R**	Eine Achse auf erfolgreiche Referenzfahrt abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse erfolgreich die Referenzfahrt abgeschlossen hat (**==**), sonst wird es rückgesetzt. **Antwort: <STX><ACK>E:CS<ETX>** oder **<STX><ACK>N:CS<ETX>** (NUR PC)
	Warten bis Position erreicht
m.a**>**zwert m.a**>R**nn m.a**>R[R**nn**]**	Die Achse m.a wird positioniert und das Programm wartet bei diesem Befehl bis der Zähler P20 größer ist als der Wert zwert. Wenn der Wert größer ist oder die Achse steht, wird im Programm fortgefahren. **Beispiel:** **m.aP20S0 m.aP14S2000 m.a+10000** **m.a>5000 m.aP14S1000** **m.a>9999 m.aS m.aP14S2000**

Befehl	***Bedeutung***
	In diesem Beispiel soll die Achse 10000 Schritte mit einer Frequenz von 2000 Hz fahren. Nach 5000 Schritten wird die Frequenz auf 1000 Hz abgesenkt und nach Stillstand der Achse wird die Frequenz wieder auf 2000 Hz gesetzt. Bei dem Befehl **m.a >5000** wird gewartet, bis die Achse die Position 5000 erreicht hat oder die Achse vorzeitig durch einen Nothalt gestoppt wird.
m.a**<**zwert m.a**<R**nn m.a**<R[R**nn**]**	Die Achse m.a wird positioniert und das Programm wartet bei diesem Befehl, bis der Zähler P20 kleiner ist als der Wert zwert. Wenn der Wert größer ist oder die Achse steht, wird im Programm fortgefahren. **Antwort: <STX><ACK>:CS<ETX>** (NUR PC), Achse steht oder die Positionsbedingung erfüllt ist, sonst wird gewartet.
	Endstufe einer Achse schalten
	Reset
m.a**C**	Die Endstufe der Achse a des Moduls m wird rückgesetzt.
	Aktivieren
m.a**MA**	Die Endstufe der Achse a des Moduls m wird aktiviert.
	Deaktivieren
m.a**MD**	Die Endstufe der Achse a des Moduls m wird deaktiviert.
	Achsenparameter
m.a**P**mm**R**	Der Parameter mm der Achse m.a wird ausgelesen. mm = Parameternummer. (NUR PC) **Antwort : <STX><ACK>zwert:CS<ETX>**
m.a**P**mm**S**zwert m.a**P**mm**SR**nn m.a**P**mm**SR[R**nn**]** oder m.a**P**mm**=**zwert m.a**P**mm**=R**nn m.a**P**mm**=R[R**nn**]**	Der Parameter mm der Achse a des Moduls m wird mit dem vorgegebenen Wert (zwert, Inhalt von Register nn oder Register [Rnn]) geladen. mm = Parameternummer

Befehl	***Bedeutung***
	Referenzsuchlauf / Initialisierung
	Zur Initialisierung der Achse muss ein Referenzsuchlauf durchgeführt werden. Als Referenzpunkte dienen die Initiatoren, auch Endschalter, Endlagenschalter oder Grenzwertschalter genannt. Die Achse fährt einen der Initiatoren an. Wenn das Initiatorsignal erkannt wird, stoppt der Motor und dreht dann solange in die Gegenrichtung, bis kein Initiatorsignal mehr anliegt. Falls ein Initiator-Offset eingestellt wurde, wird noch die Offsetstrecke gefahren und dann die Achse angehalten. Der auf diese Weise gefundene Punkt heißt „mechanischer Nullpunkt“ oder „Referenzpunkt“.
m.a**R–**	Die Achse fährt den Initiator der — Richtung an.
m.a**R–C**	Die Achse fährt den Initiator Mitte via — Richtung an (siehe Manual: Grundlagen des Positionierens; Kap. 5.4).
m.a**RC–**	Die Achse fährt den Initiator Mitte in — Richtung an, bei dem die Hälfte der Strecke bedämpft ist, die andere Hälfte ist frei (siehe Manual: Grundlagen des Positionierens; Kap. 5.4).
m.a**R+**	Die Achse fährt den Initiator der + Richtung an.
m.a**R+C**	Die Achse fährt den Initiator via + Richtung an (siehe Manual: Grundlagen des Positionierens; Kap. 5.4).
m.a**RC+**	Die Achse fährt den Initiator Mitte in + Richtung an, bei dem die Hälfte der Strecke bedämpft ist, die andere Hälfte ist frei (siehe Manual: Grundlagen des Positionierens; Kap. 5.4).
m.a**R–I**	Die Achse fährt in Minusrichtung und stoppt mit dem Nullimpuls des Inkrementellen Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**R+I**	Die Achse fährt in Plusrichtung und stoppt mit dem Nullimpuls des Inkrementellen Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**R–^I**	Die Achse fährt den Initiator der —Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des Inkrementellen Encoders die Achse stoppt. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**R+^I**	Die Achse fährt den Initiator der +Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des Inkrementellen Encoders die Achse stoppt. Nur inkrementell, kein SSI, Endat und BiSS Encoder!

Befehl	***Bedeutung***
m.a**R-C^I**	Die Achse fährt den Mitte Initiator via —Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des Inkrementellen Encoders die Achse stoppt. nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**R+C^I**	Die Achse fährt den Mitte Initiator via +Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des Inkrementellen Encoders die Achse stoppt. N, nur inkrementell, kein SSI, Endat und BiSS Encoder!!
m.a**RC–^I**	Die Achse fährt den Initiator Mitte in — Richtung nach Offset weiter auf Nullimpuls des Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**RC+^I**	Die Achse fährt den Initiator Mitte in +Richtung nach Offset weiter auf Nullimpuls des Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**RCW**	Die Achse fährt den Initiator Mitte im Uhrzeigersinn an.
m.a**RCCW**	Die Achse fährt den Initiator Mitte gegen den Uhrzeigersinn an.
m.a**RCW^I**	Die Achse fährt den Initiator im Uhrzeigersinn nach Offset weiter auf Nullimpuls des Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
m.a**RCCW^I**	Die Achse fährt den Initiator gegen den Uhrzeigersinn nach Offset weiter auf Nullimpuls des Encoders. Nur inkrementell, kein SSI, Endat und BiSS Encoder!
	Freier Lauf
m.a**L**r	Die Achse wird im freien Lauf gestartet. Sie läuft so lange, bis sie durch den Befehl m.aS oder von einem Endschalter gestoppt wird. r = + oder – für die Laufrichtung
	Relative Positionierung
m.arzwert m.ar**R**nn m.ar**R[R**nn**]**	Die vorgegebene Strecke (zwert , Inhalt von Register nn oder Register [Rnn]) wird relativ gefahren. r = + oder – für die Laufrichtung

Befehl	***Bedeutung***
	Mit Stoppbefehl über Eingang
m**.**arzwert**vE**m**.**nz m**.**ar**R**nn**vE**m**.**nz m**.**ar**R[R**nn**]vE**m**.**nz	Es wird die vorgegebene Strecke (zwert , Inhalt von Register nn oder Register [Rnn]) relativ mit Start/Stoppfrequenz P4 gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn einen Wechsel zum Zustand z erfährt oder ein Endschalter die Positionierung abbricht. r = + oder – für die Laufrichtung z = S → Eingang gesetzt z = R → Eingang rückgesetzt
m**.**arzwert**vvE**m**.**nz m**.**ar**R**nn**vvE**m**.**nz m**.**ar**R[R**nn**]vvE**m**.**nz	Es wird die vorgegebene Strecke (zwert, Inhalt von Register nn oder Register [Rnn]) relativ mit P14 (mit Rampe) gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn einen Wechsel zum Zustand z erfährt oder ein Endschalter die Positionierung abbricht. r = + oder – für die Laufrichtung z = S → Eingang gesetzt z = R → Eingang rückgesetzt
	Absolute Positionierung bezogen auf den MØP
m**.**a**A**rzwert m**.**a**A**r**R**nn m**.**a**A**r**R[R**nn**]**	Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]) absolut, bezogen auf den mechanischen Nullpunkt MØP (m.aP20), angefahren. r = + oder – für die positive oder negative Position
	mit Stoppbefehl über Eingang
m**.**a**A**rzwert**vE**m**.**nz m**.**a**A**r**R**nn**vE**m**.**nz m**.**a**A**r**R[R**nn**]vE**m**.**nz	Es wird die vorgegebene Position (zwert , Inhalt von Register nn oder Register [Rnn]) absolut mit Start/Stoppfrequenz P4 gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt oder ein Endschalter die Positionierung abbricht. r = + oder – für die positive oder negative Position z = S → Eingang gesetzt z = R → Eingang rückgesetzt
m**.**a**A**rzwert**vvE**m**.**nz m**.**a**A**r**R**nn**vvE**m**.**nz m**.**a**A**r**R[R**nn**]vvE**m**.**nz	Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]) absolut mit P14 (mit Rampe) gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt oder ein Endschalter die Positionierung abbricht. r = + oder – für die positive oder negative Position z = S → Eingang gesetzt z = R → Eingang rückgesetzt

Befehl	***Bedeutung***
	Achsen Status
m.a**SE**	Es wird der Status der Schnittstelle gelesen.
m.a**SEC**	Es wird der Achsenstatus Reset gelesen.
m.a**SE?X**	Es wird der Achsenstatus Reset hexadezimal zurückgegeben.
m.a**SE?B**	Es wird der Achsenstatus Reset binär zurückgegeben.
	Achsen Stopp
m.a**S**	Mit dem Befehl m.aS können alle Fahrbefehle abgebrochen werden. Die Achse stoppt mit der eingestellten Rampe.
m.a**SN**	Die Achse stoppt mit der eingestellten Not halt-Rampe (Parameter P7).
	Externe Achse Takt (nur für I4X)
	(Werte werden in der MCM gespeichert und sind nach dem Einschalten aktiv)
m.a**TB**=wert	Die Taktbreite für den externen Takt einstellen.
m.a**TBR**	Den Wert der Taktbreite auslesen.
m.a**TE=**n	Den externen Taktausgang einer Achse zugeordnen und aktivieren. n = 0 → aus n = 1 → ein Wichtig: bei temporärem Taktausgang muss n=0 sein.
m.a**TER**	Die gewählte Achse wird ausgelesen.
m.a**TD=**wert	Der Taktteiler für den externen Takt wird gesetzt.
m.a**TDR**	Der Wert des Teilers wird ausgelesen.
m.awert^**C**teiler	Achse a relativ verfahren und den Takt der Achse auf den externen Takt geben. Nach der Positionierung wird der externe Takt ausgeschaltet. wert → Strecke in Schritten oder mm teiler = 1/(n+1) * $Takt_{Eingang}$ 0: 1 / (0+1) = 1 1: 1 / (1+1) = 1/2 2: 1 / (2+1) = 1/3 3: 1 / (3+1) = 1/4 4: 1 / (4+1) = 1/5 5: 1 / (5+1) = 1/6 etcpp.

6.31 Bedienterminal BT5 AM Befehle

> **i** Die Funktionen können auch über die Schnittstelle aktiviert werden, wenn das Bedienterminal BT5 AM angeschlossen ist.

6.31.1 Funktionen im Ablaufprogramm

Befehl	***Bedeutung***
TF=y	Anzeigefunktion y aufrufen y = HA (Handbetrieb) y = PAR (Parameter) y = REG (Register) y = INI (Initiatorstatus) y = DI (digitaler Eingang) y = DO (digitaler Ausgang) y = AI (analoger Eingang) y = AO (analoger Ausgang) y = LREG (Registersatz vom Speicher laden) y = LTREG (Textregistersatz vom Speicher laden)

6.31.2 Daten durch Bedienterminal anzeigen

Befehl	***Bedeutung***
TPx;p=**"**ASCII text**"**	„ASCIItext“ in Zeile x an Position p anzeigen x→Zeilennummer x=1 bis 4 p→Position innerhalb einer Zeile p=1 bis 20
TPx;p**M**m.n**P**yy	Achsenparameter Pyy der Achse n der Moduls m in Zeile x an Position p anzeigen y→Parameternummer m → Modulnummer n → Achsennummer x → Zeilennummer x=1 bis 4 p→Position innerhalb einer Zeile p=1 bis 20
TPx;p**=R**nn	Inhalt des Registers n in Zeile x an Position p anzeigen x → Zeilennummer x=1 bis 4 p→Position innerhalb einer Zeile p=1 bis 20 nn → Registernummer **Antwort: <STX><ACK> :CS<ETX>**

TPx**;**p**=R**nn**;**s	Inhalt des Registers n mit s Nachkommastellen in Zeile x an Position p anzeigen x → Zeilennummer x=1 bis 4 p→ Position innerhalb einer Zeile p=1 bis 20 nn → Registernummer s → Anzahl der Nachkommastellen **Antwort: <STX><ACK> :CS<ETX>**
TPx**;**p**=R**nn**;**s+"text"	„text" an den Inhalt des Registers n mit s Nachkommastellen anfügen und in Zeile x an Position p anzeigen x → Zeilennummer x=1 bis 4 p→ Position innerhalb einer Zeile p=1 bis 20 nn → Registernummer s → Anzahl der Nachkommastellen **Antwort: <STX><ACK> :CS<ETX>+"text"**

6.31.3 Anzeige löschen (NUR PROG)

Befehl	***Bedeutung***
CT	Die gesamte Bedienterminalanzeige wird über die Schnittstelle gelöscht.
CTn	Eine einzelne Zeile löschen. n--> Zeilennummer n=1 bis 4
CTn;m	Ausgewählte Zeilen löschen. n oder m--> Zeilennummer n=1 bis 4; m=1 bis 4 **Antwort: <STX><ACK> :CS<ETX>**

6.31.4 Über Register beschreiben (NUR PROG)

Befehl	***Bedeutung***
Rnn**ST**	Das Register nn in die 4. Zeile ab Position 11 des BT5 AM schreiben.
Rnn**ST**x;p.s	Das Register nn in Zeile x an Position p mit Nachkommastellen s des BT5 AM schreiben.

6.31.5 Tastenabfrage Bedienterminal BT5 AM (auch von PC)

Befehl	***Bedeutung***
	Bedingte Tastenabfrage
#vFn	Wenn die Funktionstaste n gedrückt ist, wird das Bedingungsbyte gesetzt. Ist die Taste nicht gedrückt, dann ist das Bedingungsbyte rückgesetzt. n = Funktionstaste F1 bis F6
#vnmx	Im Programm: wenn die Taste n oder m oder x gedrückt ist, wird das Bedingungsbyte gesetzt. Ist die Taste nicht gedrückt, dann ist das Bedingungsbyte rückgesetzt. n, m, x = 0 bis 9 (Taste 0 bis 9) n, m, x = L (Taste CURSOR LINKS) n, m, x = R (Taste CURSOR RECHTS) n, m, x = U (Taste CURSOR OBEN) n, m, x = D (Taste CURSOR UNTEN) n, m, x = H (Taste CURSOR HOME) n, m, x = B (Taste BLÄTTERN) n, m, x = C (Taste CLEAR) n, m, x = E (Taste ENTER) n, m, x = P (Taste PRINT) n, m, x = ? (Taste ?) n, m, x = + (Taste +) n, m, x = – (Taste –) n, m, x = . (Taste .) **Beispiel: *wait* #vH1? NN*wait*** Hier wird die Tastatur des Bedienterminals so lange abgefragt, bis die Taste **H, 1** oder **?** gedrückt wird. Das Bedingungsbyte ist rückgesetzt, wenn keine der Tasten **HOME,1** oder **?** gedrückt ist. Beim Befehl **NN*wait*** wird zum Zeilenlabel *wait* gesprungen. **Antwort: : <STX><ACK> E:CS<ETX>** oder **<STX><ACK> N:CS<ETX>** (NUR PC) **Wichtig:** Die EINGABE-Taste ist für eine Abfrage nicht definiert.

6.31.6 Tastenabfragen programmiert (an „C“ angelehnte *phy*LOGIC® Befehle) (NUR PROG)

Bedingungsvergleich: == gleich

!= ungleich

#**F**n und #n können in den C-Abfragen genutzt werden.

#**F**n == 1 // Funktionstaste n auf Bedingung gleich abfragen

#**F**n != 1 // Funktionstaste n auf Bedingung ungleich abfragen

#n == 1 // Taste n auf Bedingung gleich abfragen

#n != 1 // Taste n auf Bedingung ungleich abfragen

Beispiele:

```
while(#F1!=1) {}          // warten bis Funktionstaste F1 gedrückt wird

while(#F1==1) {}          // warten bis Funktionstaste F1 losgelassen wird
```

7 An „C“ angelehnte *phy***LOGIC**® Befehle

ACHTUNG – Programmierfehler!

Durch falschen Aufbau eines Befehls kann es zu Fehlfunktionen kommen.

- Die AUSFÜHRUNG eines Befehls muss immer **zwischen** den geschweiften Klammern geschrieben werden: z.B. do**{**Ausführung**}** while (Bedingung);

7.1 Befehl „if“

If (Bedingung) {Ausführung}

If (Bedingung && Bedingung) {Ausführung}

If (Bedingung) {Ausführung} else {Ausführung}

If (Bedingung) {Ausführung} else if (Bedingung) {Ausführung}

Bedingung:	Abfrage von Register, Eingänge, Ausgänge, Achsenzustände, Timer
Bedingungsvergleich:	== gleich
	!= ungleich
	<= kleiner gleich
	>= größer gleich
	> größer
	< kleiner
Verknüpfung:	&& UND Verknüpfung
	|| ODER Verknüpfung

Beispiele: if(R10>100) { A1.1S}

if (R10>100 && R10<200)

{ A1.1S} else { A1.2S}

if (R10>100 && R10<200)

{ A1.2S} else if (R10<100){ A1.1S}

7.2 Befehl „while“

while (Bedingung) {Ausführung}

while (Bedingung && Bedingung) {Ausführung}

Bedingung:	Abfrage von Register, Eingänge, Ausgänge, Achsenzustände, Timer
Bedingungsvergleich:	== gleich
	!= ungleich
	<= kleiner gleich
	>= größer gleich
	> größer
	< kleiner
Verknüpfung:	&& UND Verknüpfung
	|| ODER Verknüpfung

Beispiele:

```
while(M1.1!=H) {}            // warten bis Motor steht
while(M1.1!=H || M1.2!=H)
        { A1.1S A1.2S}
```

7.3 Befehl „do while“

do{Ausführung} while (Bedingung);
do{Ausführung} while (Bedingung && Bedingung);

Bedingung:	Abfrage von Register, Eingänge, Ausgänge, Achsenzustände, Timer
Bedingungsvergleich:	== gleich
	!= ungleich
	<= kleiner gleich
	>= größer gleich
	> größer
	< kleiner
Verknüpfung:	&& UND Verknüpfung
	|| ODER Verknüpfung

Beispiel:

```
do {
        if(E1.1==S){            // Abfrage ob Eingang 1 gesetzt
          break;                // wenn ja while Schleife abbrechen
                    }
   } ; while(M1.1!=H);          // warten bis Motor steht
      M1.1S                     // Achse 1 stoppen

   while(M1.1!=H || M1.2!=H);
  {A1.1S A1.2S}
```

7.4 Befehl „for“

for (Wert Initialisierung;Wert Vergleich;Wert Manipulation){ Ausführung }
for (Wert Initialisierung;Wert Vergleich && Wert Vergleich;Wert Manipulation){ Ausführung}

Wert:	Register
Initialisierung:	Rnn=value (value=Zahl)
Vergleich:	Rnn<xxx (xxx = Zahl oder Register)
Manipulation:	Rnn++ alle Funktionen, die den Registerinhalt verändern

Bedingungsvergleich:	== gleich
	!= ungleich
	<= kleiner gleich
	>= größer gleich
	> größer
	< kleiner
Verknüpfung:	&& UND Verknüpfung
	|| ODER Verknüpfung

```
Beispiel:  for (R1=1; R1<9;R1++)      // Register auf 1 setzen ;Vergleich R1<9 ;R1+1
               { A1.R1S T100}         // Ausgang 1 bis 8 einschalten
```

7.5 Befehl „break“

„break“ in einer while,do while und for Schleife bricht die Schleife ab.

7.6 Befehl „continue“

„continue“ in einer while- oder do-while Schleife springt zum Vergleich der Schleife.

„continue“ in einer for-Schleife springt zur Wert Manipulation und dann zum Wert Vergleich.

7.7 Befehl „goto“

goto *label* : Sprunganweisung zu „label“

7.8 Befehl „switch“

switch(Rnn){case x:break:default:}

„switch“ im Ablaufprogramm: Abfrage, ob der Wert des Registers Rnn x entspricht.

break: die case-Anweisung wird beendet.

default: das Register Rnn wird mit dem Defaultwert gesetzt.

Bedingung:	Abfrage von Register
Bedingungsvergleich:	== gleich

Beispiel:

```
switch (R100){
        case 1:
                R1=1
                R101=1
                U*Test2*
                break;
```

```
        case 2:
                R1=2
                R101=6
                U*Test2*
                break;
        case 3:
                R1=3
                break;
        case 4:
                R1=4
                break;
        default:
                break;
}
```

8 *phy*LOGIC® Befehle

9 Parameter

Zum Betrieb der Steuerung werden verschiedene Voreinstellungen wie Frequenzen, Beschleunigungsrampen oder Wartezeiten benötigt, die als **Parameter** bezeichnet werden.

Bei Auslieferung sind Grundparameter hinterlegt, mit denen die Steuerung in vielen Anwendungsfällen betrieben werden kann. Diese Parameter können in *phy***LOGIC**® ToolBox ausgelesen und editiert werden.

Zu den Parametern gehören auch Zähler, die vom Programm fortlaufend aktualisiert werden. Es ist möglich, die Zähler auszulesen und z.T. auch zu editieren.

Die Parameter gibt man für jede Achse separat ein. Zur Kennzeichnung der Achse muss vor der Parameternummer Modul- und Achsen-Nummer eingefügt werden.

Beispiel: m.aP15 ist die Beschleunigungsrampe für die Achse a des Moduls m.

Auch innerhalb des Programms ist es möglich, Parameter (z.B. Geschwindigkeit) zu ändern.

Parameter können entweder beschrieben oder ausgelesen werden.

P49 kann nur ausgelesen werden.

P19 bis P22 sind Zähler, die vom Programm bei Verfahren der Achsen laufend aktualisiert werden.

- P27 bis P54 sind spezielle Parameter für die *phy***MOTION**®.
- Stromwerte (P40 bis P42) und P45 gelten nur für INTERNE Endstufen oder Endstufen, die über einen Bus verbunden sind:

*phy***MOTION**®	**Versorgung**	**Endstufen-Modul(en)**	**P45**
	EXTERN	INAM,I1AM01,I1AM02,…	wie in Kap.9.1 beschrieben
		EXAM	ohne Funktion; externe Endstufe wird via DIP- bzw. Kodier-Schalter eingestellt
	INTERN	integriert (MSX+, ZMX+,…)	ungültig; Aufteilung der Schrittauflösung (0 bis 15) gemäß der Endstufentabelle (siehe Endstufenmanual)

9.1 Parameterliste

Nr.	Bedeutung	Auslieferungszustand
P01	Art der Bewegung (freier Lauf, relativ / absolut, Referenzfahrt) 0 = rotatorisch (Endschalter werden ignoriert) 1 = Hardware Endschalter werden überwacht für XY Tische oder andere lineare Systeme, zwei Endschalter: Mechanischer Nullpunkt und Begrenzung in −Richtung, Begrenzung in +Richtung 2 = Software Endschalter werden überwacht 3 = Hardware und Software Endschalter werden überwacht	0
P02	Maßeinheit der Bewegung: nur für die Anzeige 1 = Schritt 2 = mm 3 = Zoll 4 = Grad	1
P03	Umrechnungsfaktor Spindelsteigung (Skalierung) 1 Schritt entspricht ... Bei P03 = 1 (Schritte) ist der Umrechnungsfaktor 1 Berechnung des Umrechnungsfaktors: $Umrechnungsfaktor = \frac{Spindelsteigung}{Motorschrittzahl\ pro\ Umdrehung}$ Beispiel: 4 mm Spindelsteigung 200-schrittiger Motor = 400 Schritte/U im Halbschrittbetrieb $Umrechnungsfaktor = \frac{4}{400} = 0{,}01$	1
P04	Start-/Stoppfrequenz Die Start-/Stoppfrequenz ist die maximale Frequenz, bei der der Schrittmotor noch ohne Rampe starten oder stoppen kann, ohne dass Schrittverluste auftreten. Die Start-/Stoppfrequenz ist abhängig von verschiedenen Größen wie Motortyp, Last, Mechanik, Endstufe. Eingabe der Frequenz in Hz	400
P05 **P06**	nicht belegt	

Nr.	Bedeutung	Auslieferungszustand
P07	Achsenrampe für Nothalt Eingabe bei I1AM0x: in 4000 Hz/s-Schritten I4XM01: in 1 Hz/s-Schritten	100 000
P08	f_{max} MØP (Mechanischer Nullpunkt) Fahrfrequenz beim Initialisieren (Referenzieren) Eingabe in Hz (ganzzahliger Wert) I1AM0x: max. 40 000 I4XM01: max. 4 000 000	4000
P09	Rampe MØP für Initialisierung, zugehörig zu Parameter P08 Eingabe bei I1AM0x: in 4000 Hz/s-Schritten I4XM01: in 1 Hz/s-Schritten	4000
P10	f_{min} MØP, Fahrfrequenz beim Verlassen der Endschalter Eingabe in Hz	400
P11	MØP Offset für Endschalter Plusrichtung (weg von „LIMIT+“ Schalter in Richtung „LIMIT−“ Schalter) Abstand des mechanischen Nullpunkts MØP (Referenzpunkt) vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt P11 ≥ 0	0
P12	MØP Offset für Endschalter Minusrichtung (weg von „LIMIT−“ Schalter in Richtung „LIMIT+“ Schalter) Abstand des mechanischen Nullpunkts MØP vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt P12 ≥ 0	0
P13	Beruhigungszeit MØP Wartezeit bei Initialisierung Eingabe in ms	20
P14	f_{max} Lauffrequenz bei Positionier Befehlen Eingabe in Hz (ganzzahliger Wert) I1AM0x max. 40 000 I4XM0x: max. 4 000 000	4000

Nr.	Bedeutung	Auslieferungszustand
P15	Rampe für Lauffrequenz (P14) Eingabe bei I1AM0x: in 4000 Hz/s-Schritten I4XM0x: in 1 Hz/s-Schritten	4000
P16	Beruhigungszeit Position Wartezeit nach Ausführung eines Fahrbefehls Eingabe in ms	20
P17	Boost (Strom definiert in P42) 0 = aus 1 = ein während der Motor fährt 2 = ein bei Hochlauf und Absenkung der Fahrfrequenz (Rampe) <u>Anmerkungen:</u> Der Booststrom wird in Parameter P42 programmiert für interne Endstufen. Mit dem Parameter P17 wird festgelegt, wann die Steuerung auf Booststrom umschaltet. P17 = 1 bedeutet, dass bei fahrendem Motor immer der Booststrom fließt. Bei Stillstand des Motors wird auf Stoppstrom umgeschaltet.	0
P18	Intern belegt für Positionsvorgabe	
P19	Abweichung zwischen P21 und P20	
P20	Streckenzähler wird mit dem Umrechnungswert P3 geschrieben. Dieser zählt die Impulse, die an die Endstufe gegeben werden. Er wird mit Referenzfahrt im Referenzpunkt automatisch auf 0 gesetzt.	0
P21	Absolutwertzähler Auf P21 wird der Wert von P22 per Software verlängert. Die Encoder-Zähler haben eine feste Auflösung, z.B. 10 Bit (bei Single-Turn-Encodern die Auflösung Bit per Turn), danach wiederholt sich der gelesene Wert. Bei kontinuierlichem Motorlauf entsteht ein Sägezahn-Verlauf der Zahlenwerte. Per Software wird dieser Verlauf „begradigt“. Mittels P3 und P39 können dann P20 und P21 auf gleiche Werte pro Umdrehung skaliert werden und sind somit direkt vergleichbar, siehe P36.	0

Nr.	Bedeutung	Auslieferungszustand
P22	Encoderzähler Gibt die aktuelle Encoderposition an. Wird nur bei A/B-Encodern Null gesetzt (nach Reset), die Absolut-Encoder behalten ihren Wert. Kann bei A/B Encodern mit Befehl m.NP20=wert beschrieben werden.	0
P23	Software Endschalter (Achsenbegrenzung pos. Richtung +) Bei Erreichen dieser Strecke wird der Lauf in +Richtung abgebrochen. 0 = keine Begrenzung	0
P24	Software Endschalter (Achsenbegrenzung neg. Richtung −) Bei Erreichen dieser Strecke wird der Lauf in −Richtung abgebrochen. 0 = keine Begrenzung	0
P25	Spielausgleich Gibt die Strecke an, um die die Sollposition in der gewählten Richtung überfahren und anschließend in umgekehrter Richtung angefahren wird. 0 = kein Spielausgleich	0
P26	Mittels P26 wird die Daten-Transfer-Rate festgelegt (**NUR** für SSI Encoder), mit der der Encoder ausgelesen wird. Die Transfer-Rate ist abhängig von der Länge der Leitung mit der der Encoder am Gerät angeschlossen ist, je kürzer die Leitung desto schneller kann der Encoder ausgelesen werden Daten-Transfer-Rate 1 bis 10 (= 100 bis 1000 kHz) 1 = 100 kHz 2 = 200 kHz 3 = 300 kHz 4 = 400 kHz 5 = 500 kHz 6 = 600 kHz 7 = 700 kHz 8 = 800 kHz 9 = 900 kHz 10 = 1000 kHz	1

<table>
<tr><th>Nr.</th><th>Bedeutung</th><th>Auslieferungszustand</th></tr>
<tr><td>P27</td><td>Endschaltertyp
PNP-Öffner oder PNP-Schließer
<table>
<tr><th></th><th>− Begrenzung</th><th>Mitte/Ref</th><th>+ Begrenzung</th></tr>
<tr><td>0</td><td>Öffner</td><td>Öffner</td><td>Öffner</td></tr>
<tr><td>1</td><td>Öffner</td><td>Öffner</td><td>Schließer</td></tr>
<tr><td>2</td><td>Schließer</td><td>Öffner</td><td>Öffner</td></tr>
<tr><td>3</td><td>Schließer</td><td>Öffner</td><td>Schließer</td></tr>
<tr><td>4</td><td>Öffner</td><td>Schließer</td><td>Öffner</td></tr>
<tr><td>5</td><td>Öffner</td><td>Schließer</td><td>Schließer</td></tr>
<tr><td>6</td><td>Schließer</td><td>Schließer</td><td>Öffner</td></tr>
<tr><td>7</td><td>Schließer</td><td>Schließer</td><td>Schließer</td></tr>
</table></td><td>0</td></tr>
<tr><td>P28</td><td>Achsen Optionen
0 = Endstufe nach dem Einschalten deaktiviert
1 = Endstufe nach dem Einschalten aktiviert</td><td>1</td></tr>
<tr><td colspan="3">P29 nicht belegt</td></tr>
<tr><td>P30</td><td>Nur für I4XM01!
Einstellung Frequenzband
0 = manuell
1 = automatisch
<u>Anmerkung:</u>
Es wird empfohlen, mit der automatischen Frequenzbandeinstellung zu arbeiten. Der Controller wählt für die vorgegebene Frequenz und Rampe das richtige Frequenzband aus.</td><td>0</td></tr>
</table>

<table>
<tr><th>Nr.</th><th>Bedeutung</th><th>Auslieferungszustand</th></tr>
<tr><td>P31</td><td>Nur für I4XM01!

Manuelle Auswahl des Frequenzbandes
(nur wenn P30 auf manuell eingestellt ist)

Der Parameter verändert den Vorteiler der die Frequenz erzeugende Hardware mit einem von 20 MHz abgeleiteten Takt versorgt.
<table>
<tr><th>P31</th><th>Lauffrequenz</th><th>Auflösung</th><th>Vorteiler</th></tr>
<tr><td>0</td><td>1 Hz … 8 kHz</td><td>⅛ Hz</td><td>2440</td></tr>
<tr><td>1</td><td>1 Hz … 16 kHz</td><td>¼ Hz</td><td>1220</td></tr>
<tr><td>2</td><td>1 Hz … 32 kHz</td><td>½ Hz</td><td>609</td></tr>
<tr><td>3</td><td>1 Hz … 65 kHz</td><td>1 Hz</td><td>304</td></tr>
<tr><td>4</td><td>2 Hz … 130 kHz</td><td>2 Hz</td><td>152</td></tr>
<tr><td>5</td><td>4 Hz … 260 kHz</td><td>4 Hz</td><td>75</td></tr>
<tr><td>6</td><td>8 Hz … 520 kHz</td><td>8 Hz</td><td>37</td></tr>
<tr><td>7</td><td>16 Hz … 1 MHz</td><td>16 Hz</td><td>18</td></tr>
<tr><td>8</td><td>32 Hz … 2 MHz</td><td>32 Hz</td><td>9</td></tr>
<tr><td>9</td><td>64 Hz … 4 MHz</td><td>64 Hz</td><td>4</td></tr>
</table>
Der Parameter kann für individuelle Einstellungen verwendet werden, wenn die automatische Frequenz-bandeinstellung für den speziellen Anwendungsfall nicht geeignet ist.</td><td>3</td></tr>
<tr><td>P32</td><td>Rampenform bei Positionierungen

0 = s-förmig
1 = linear

<u>Anmerkung:</u> Die s-förmige Rampe kann mit P33 geändert werden.</td><td>1</td></tr>
<tr><td>P33</td><td>Bogenwertvorgabe für s-förmige Rampe

Werte: 1 bis 32767

P33: niedriger Wert → steile S-Rampe

P33: hoher Wert → flache S-Rampe</td><td>1</td></tr>
</table>

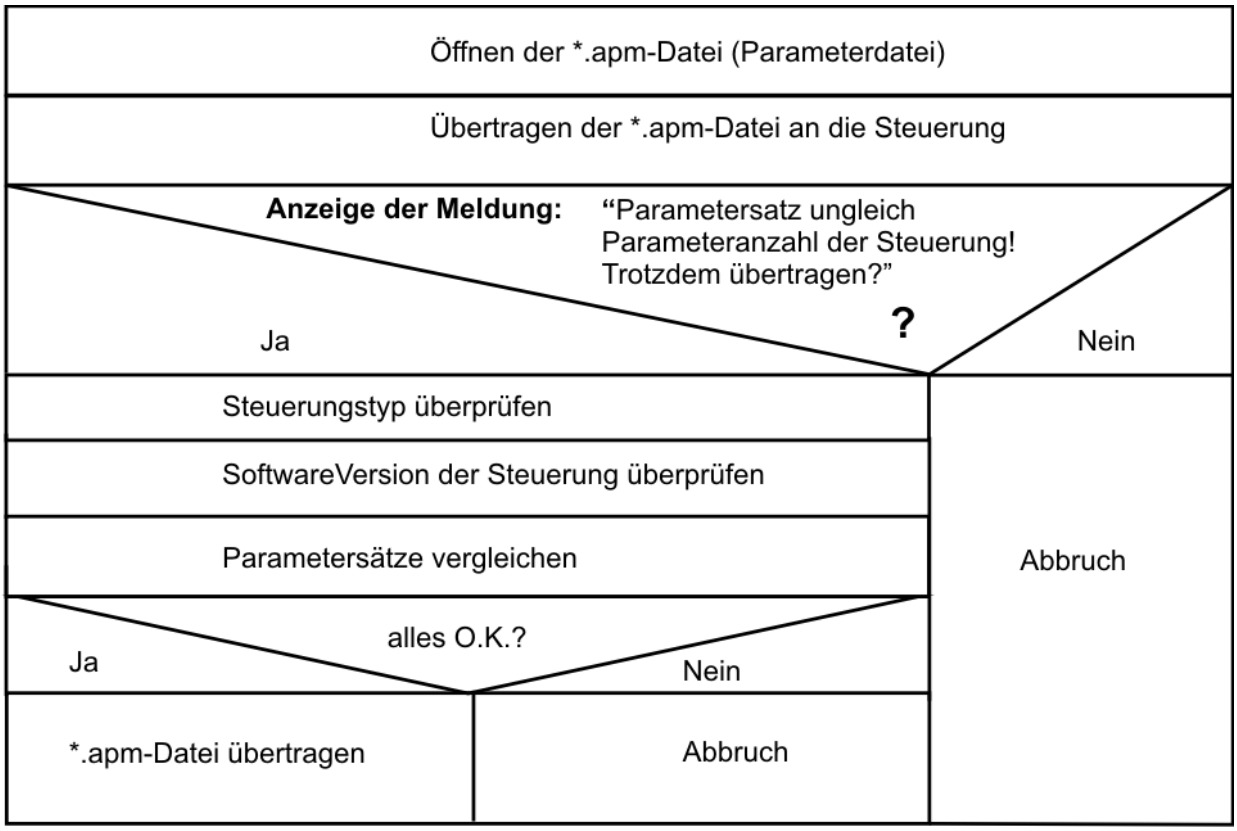
Nr.	Bedeutung	Auslieferungszustand
P34	Encodertyp 0 = kein Encoder 1 = inkrementell 5,0 V 2 = inkrementell 5,5 V 3 = serielle Schnittstelle SSI Binär Code 5,0 V 4 = serielle Schnittstelle SSI Binär Code 5,5 V 5 = serielle Schnittstelle SSI Gray Code 5,0 V 6 = serielle Schnittstelle SSI Gray Code 5,5 V 7 = EnDat 5,0 V 8 = EnDat 5,5 V 9 = Resolver 10 = 4-Draht-LVDT 11 = 5/6-Draht-LVDT 12 = BiSS 5 V 13 = BiSS 24 V	0
P35	Auflösung bei SSI Encoder und EnDat Eingabe: maximale Auflösung in Bit (max. 48 Bit) Besonderheit bei EnDat: wenn der Parameter Null gesetzt wird verwendet die Steuerung die Auflösung die aus dem angeschlossenen Messgerät ausgelesen wird.	10
P36	Encoderfunktion Der Parameter legt fest ob P21 als reiner Zähler verwendet wird oder ob dessen Wert kontinuierlich mit dem Wert des Zählers P20 verglichen wird. Bei SFI gilt: Ist die Abweichung P20 zu P21 größer als die Toleranz P37, dann Stoppen der Achsen und Fehlermeldung. 0 = Zähler 1 = Zähler und Schrittfehlererkennung SFI	0
P37	Toleranz für Schrittfehlererkennung Eingabe: Toleranzwert für SFI-Auswertung in der eingestellten Auflösung (P3 * P20). Wird P21 zur Schrittfehlererkennung verwendet muss die Skalierung des Zählers P20 * P3 gleich sein der Skalierung des Zählers P21 * P39 und P21 muss nach Initialisierung der Skalierung genullt werden (bzw. auf den gleichen Wert wie P20 gesetzt werden). Z.B. Skalierung auf 360° /U: Motor 200 Schritte pro Umdrehung, 1/20-Schritt, →P3 = 360 / 200 / 20 = 0.09, Encoder 10 Bit / U → P39 = 360 / 2^{10} = 0.3515625	0

Nr.	Bedeutung	Auslieferungszustand
P38	Encoder Zählrichtung 0 = + (positiv) 1 = − (negativ)	0
P39	Encoder Umrechnungsfaktor (Skalierung) 1 Inkrement entspricht ... Berechnung des Umrechnungsfaktors: $Umrechnungsfaktor = \frac{Spindelsteigung}{Encoder - Schrittzahl\ pro\ Umdrehung}$	1
P40	Stoppstrom in 0,01 A_{eff} oder 0,1 A_{eff} Stufen abhängig von der Endstufe einstellbar siehe Manual der Endstufe	2
P41	Laufstrom in 0,01 A_{eff} oder 0,1 A_{eff} Stufen abhängig von der Endstufe einstellbar siehe Manual der Endstufe	6
P42	Booststrom in 0,01 A_{eff} oder 0,1 A_{eff} Stufen abhängig von der Endstufe einstellbar siehe Manual der Endstufe	10
P43	Stoppstromüberhöhungszeit in ms	20
P44	Nur für I4XM01! Taktquelle für die Achse 0 = 1:1 (Eingang=Ausgang) 1 = von X 2 = von Y 3 = von Z 4 = von U 5 = von extern	0

Nr.	Bedeutung	Auslieferungszustand
P45	Schrittauflösung und Vorzugsdrehrichtung Die Schrittauflösung ist abhängig vom Endstufentyp: siehe im Im Manual der Endstufe (Kapitel Leistungsmerkmale) P45 gilt nur für INTERNE Endstufen oder Endstufen, die über einen Bus verbunden sind. Vorzugsdrehrichtung: bei APS05 und APS09 bzw. LPS gilt: Bit7=1: negative Vorzugsdrehrichtung Bit7=0: positive Vorzugsdrehrichtung Beispiel bei APS05 Endstufe: P45=3_{dez} (11_{bin}): ¼ Schrittauflösung und positive Vorzugsdrehrichtung P45=131_{dez} (10000011_{bin}): ¼ Schrittauflösung und negative Vorzugsdrehrichtung	3
P46	nicht belegt	
P47	nicht belegt	
P48	nicht belegt	
P49	Endstufentemperatur in 1/10 °C	(nur lesen)
P50	Teiler für Takt (nur für I4XM01) $Takt_{Ausgang}$ = 1/(n+1) * $Takt_{Eingang}$ 0: 1 / (0+1) = 1 1: 1 / (1+1) = 1/2 2: 1 / (2+1) = 1/3 3: 1 / (3+1) = 1/4 4: 1 / (4+1) = 1/5 5: 1 / (5+1) = 1/6 etcpp.	n = 0
P51	Taktbreite: (n+1) * 100 ns (nur für I4XM01) n: 0….255 z.B. n=19: (19+1)*100 ns=2000 ns= 2µs → F_{max}=1/(2*2 µs)=250 kHz	n = 19
P52	Intern belegt für Trigger-Position	
P53	Endstufenüberwachung 0 = off 1 = on	1

Nr.	Bedeutung	Auslieferungszustand
P54	Motortemperatur in 1/10 °C −999999: Temperaturmodul nicht vorhanden −9999: Überlauf negativ oder Temperatur kleiner −220 °C bei PT100 9999: Überlauf positiv oder Temperatur größer +390 °C bei PT100	−999999 (nur lesen)
P55	Motortemperatur **Warnung** in 1/10 °C Hat sich der Motor auf den definierten Temperaturwert erwärmt, erfolgt eine Warnung. Es wird empfohlen, den Motor erst nach Abkühlung wieder in Betrieb zu nehmen.	0
P56	Motortemperatur **Abschaltung** in 1/10 °C Hat der sich der Motor auf den definierten Temperaturwert erwärmt, schaltet die Steuerung ab und die Endstufe muss rückgesetzt werden.	0
P57	Resolver Spannung n=3…10 (Volt)	3
P58	Resolver Ratio (Verhältnis Primär zu Sekundär-Spule) 0=1/8 1=1/4 2=1/2 3=1 4=2	2
P59	Modulo Schrittweite Vorgabe der Schritte für 1 Motorumdrehung	0
P60	Modulo Encoder Vorgabe für Encoder für 1 Motorumdrehung	0
P61	Modulo Umdrehungen (Schritte) Zähler für Anzahl der Umdrehungen Kann nur auf 0 gesetzt werden.	0

9.2 Parametersatz Übertragung zur Steuerung



10 Programme, Parameter und Register speichern

Programme und Parameter können mit *phy***LOGIC**® ToolBox bearbeitet, an die Steuerung übertragen und dort gespeichert werden. Während des Programmlaufs können Register und Zähler vom Programm geändert werden.

Anzahl der Programme	512	
Programmarbeitsspeicher RAM	384 kB	Programm wird zum Ablauf in dieses RAM geschrieben und dann gestartet.
Flash Speicher	4 MB	Speicherung von Programmen, Register- und Textregistersätzen
Register FRAM	1000	Register bleiben nach dem Ausschalten erhalten.
Textregister FRAM	100	

11 Stichwortverzeichnis