MINILOG

phytron

Programmiermanual
für OMC und TMC

Manual 1173-A009 D

customized solutions
in motion

Programmiermanual MINILOG

Programmierung der Steuerungen
OMC und TMC

ORIGINAL BETRIEBSANLEITUNG

Phytron GmbH

Industriestraße 12

82194 Gröbenzell, Deutschland

Tel.: +49 8142/503-0

Fax: +49 8142/503-190

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

Den neuesten Stand des Handbuchs finden Sie im Internet unter www.phytron.de.

Inhaltsverzeichnis

1 Struktur der Befehle

1.1 Aufbau des Befehlscodes

Xrzwert	**X**	Fettgedruckte Zeichen sind der Befehlscode und müssen unverändert eingegeben werden. In diesem Beispiel: **X** = Fahrbefehl für relative Positionierung der X-Achse **Bei allen Befehlen wurde der Befehlcode für die X-Achse abgedruckt, weil die Achse bei Einachsensteuerungen (OMC) immer „X“ genannt wird. Bei Mehrachsen-steuerungen (auch Master/Slave bis 8 Achsen) muss statt X der entsprechende Buchstabe X, Y, ... oder 1, 2, ... eingesetzt werden.**
	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. 5) festgelegt.
Beispiel: X+1000		Relativer Fahrbefehl an die X-Achse: Fahre 1000 Schritte in Plusrichtung.

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 A001S wird als A1S 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. 2.13, Seite 18, muss der 1. alphanumerische Programmname durch eine Leerstelle vom 2. Programmnamen bzw. dem nachfolgendem alphanumerischenTeil des Befehlscodes getrennt werden.**

1.2 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**nn**–**mm	Das Register nn wird mit dem Status der Eingänge nn bis mm binär beschrieben. **Beispiel: R1BE1–8** Die Eingänge 1 bis 8 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 **Indirekte Adressierung:**

Befehl	*Bedeutung*
R[Rnn**]BE**nn**–**mm	Das Register, das durch das Register nn adressiert wird, wird mit dem Status der Eingänge nn bis mm binär beschrieben **Beispiel: R1S10 R[R1]BE1–8** Die Eingänge 1 bis 8 haben z. B. den Zustand: **1010 0101.** Das Register 1 wird mit dem Befehl **R1S10** auf den Wert 10 gesetzt. 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 (ab Seite 16) und Unterprogrammaufrufen (ab Seite 33) 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 6 alphanumerische Zeichen haben. Innerhalb eines Programms können bis maximal 100 Labels eingesetzt werden.

Beispiel: *[Labelname]*

Programmname:

Programmnamen [name] im Befehlscode können bis zu 8 alphanumerische Zeichen haben.

1.3 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 S. 10) oder durch einen Register Vergleich (siehe S. 23) gesetzt worden sein.

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.

1.4 Daten– und Telegrammformat

Datenformat: No Parity
1 Stopbit
8 Bit ASCII-Code
115 200 Baud

Das **Sendetelegramm** vom PC via RS232 ist wie folgt definiert:

Ohne Prüf-summe: <STX> | Adresse | Befehl | <ETX> | <CR> | <LF>

Mit Prüf-summe: <STX> | Adresse | Befehl | Separator | Prüfsumme | <ETX> | <CR> | <LF>

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

<STX> | ACK | Antwort | <ETX> | <CR> | <LF> oder

<STX> | ACK | <ETX> | <CR> | <LF> oder

<STX> | NAK | <ETX> | <CR> | <LF>

	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	MINILOG Befehlscode
Separator	:, $3A_H$ zur Trennung von Nutzdaten und Prüfsumme
Prüfsumme	Höherwertiges Byte der Prüfsumme (Berechung s.u.)
	Niederwertiges Byte der Prüfsumme (Berechung s.u.)
<ETX>	(End of Text, 03_H) als Telegrammende-Kennung.
<CR>	(Carriage Return, $0D_H$) als Zeilenumschaltung-Kennung
<LF>	(Line Feed, $0A_H$) als Zeilenvorschub-Kennung
ACK	(Acknowledge 06_H), der Befehl wurde quittiert
NAK	(Negative Acknowledge 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 ⊕ Datenbyte1 ⊕ Datenbyte2 ... ⊕ DatenbyteN ⊕ Separator
```

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 OMC/TMC 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. Die OMC/TMC akzeptiert auch Telegramme, bei denen statt der beiden Prüfsummenbytes **zwei X** gesendet werden, im Beispiel also

<STX>| 1 | X | + | 1 | 0 | 0 | : | X | X <ETX> |<CR> |<LF>

[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.

2 MINILOG Befehle

2.1 Ausgänge

Befehl	*Bedeutung*
	Ausgänge schalten
Annnz	Man kann einen Ausgang oder mehrere Ausgänge gleichzeitig schalten. nnn, mmm, xxx → Nummer des Ausgangs
Annnzmmmzxxxz	z = S → setzen z = R → rücksetzen **Beispiel: A1S2R3S** Ausgang **1** und **3** einschalten, Ausgang **2** ausschalten
	Ausgangsgruppe lesen
AGn**R**	Die Ausgangsgruppe n lesen. (NUR PC)
	Beispiel: AG2R Die 2. Ausgangsgruppe wird gelesen
	Antwort : <STX><ACK>nnnnnnnn<ETX><CR><LF> n = 0 Ausgang nicht gesetzt n = 1 Ausgang gesetzt
	Ausgangsgruppe Ausgänge setzen
AGn**S**zzzzzzzz	Ausgangsgruppe setzen n=1 oder 2, z= 0 oder 1.
	z muss immer 8 Stellen haben
	Beispiel: AG1S10101001 Die 1. Ausgangsgruppe wird mit der Information '10101001' gesetzt
	Ausgangszustand lesen
ARnnn;mmm;xxx	Der Zustand der Ausgänge nnn, mmm, xxx wird gelesen. (NUR PC)
	Antwort : <STX><ACK>nnn<ETX><CR><LF>
	n = 0 Ausgang nicht gesetzt
	n = 1 Ausgang gesetzt
	Wichtig: Zwischen den Ausgangsnummern ein **;** setzen.

2.2 A/D-Wandler (Option OMC/TMC)

Befehl	*Bedeutung*
ADn**C**x**S**y	A/D-Wandler-Einstellung übertragen n→ Adresse des A/D-Wandlers: n=0 oder n=1 x→ Kanal des A/D-Wandlers x=0 bis 3 y→ Übertragungsart: y=0 bipolar y=1 unipolar
ADn**C**x**R**	AD-Wandler-Einstellung lesen **Antwort : <STX><ACK>0<ETX><CR><LF>** oder **<STX><ACK>1<ETX><CR><LF>** (NUR PC)

2.3 Reset

Befehl	*Bedeutung*
CR	Die Steuerung wird über die Schnittstelle zurück gesetzt.
CT	Die Terminalanzeige wird über die Schnittstelle gelöscht. **Antwort : <STX><ACK><ETX><CR><LF>** (NUR PC)

2.4 Schreibausgabe über serielle Schnittstelle

Befehl	*Bedeutung*
	Es können über die 3 seriellen Schnittstellen (X31, X32, X9) Informationen ausgegeben werden. Die Schreibausgabe erfolgt ohne Formatierung. s = 1,2 oder 3 → Schnittstellenbezeichnung 1 → Schnittstelle RS232 (X31) 2 → Schnittstelle RS232 (X32) 3 → Terminalschnittstelle (X9)
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.

Befehl	*Bedeutung*
Dsx**P**mm	Den Parameter mm der Achse x ausgeben. mm = 1 bis 45 → Parameternummer x = 1 bis 8 oder X,Y,Z,W,5,6,7,8 → Achsenbezeichnung **Beispiel: D24P10** Hier wird der Parameter 10 der Achse 4 über die Schnittstelle X32 ausgegeben.

2.5 Eingangsabfragen

Befehl	*Bedeutung*
	UND-Verknüpfung
E^nnzmmzxxz	Es werden die Eingänge nn, mm, xx als UND-Verknüpfung abgefragt. Wenn die UND-Bedingung erfüllt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt. nn. mm. xx → Nummer des Eingangs z = S → Eingang gesetzt z = R → Eingang rückgesetzt **Beispiel: E^1S2R3S** 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 <ETX><CR><LF>** oder **<STX><ACK> N <ETX><CR><LF>** (NUR PC)

Befehl	*Bedeutung*
Evnnzmmzxx	**ODER-Verknüpfung**

Es werden die Eingänge nn, mm, xx als ODER Verknüpfung abgefragt. Wenn die ODER-Bedingung erfüllt ist, wird das Bedingungsbyte gesetzt, sonst wird es rückgesetzt.

nn. mm. xx → Nummer des Eingangs
z = S → Eingang gesetzt
z = R → Eingang rückgesetzt

Beispiel: Ev1S2R3S

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 <ETX><CR><LF> oder
<STX><ACK> N <ETX><CR><LF> (NUR PC)

Warten bis Zustand erfüllt

Ennz	Warten auf den vorgegebenen Eingangszustand. Das Programm wartet hier, bis der Zustand erreicht wird. Das Bedingungsbyte wird nicht beeinflusst. (NUR PROG)
Ennzmmz	Bei Eingangsabfragen mit mehreren Eingängen werden die Zustände der Eingänge nacheinander abgefragt (keine UND-Verknüpfung). Das Bedingungsbyte wird nicht beeinflusst. (NUR PROG)

Beispiel: E1S2R3S

Hier werden die Zustände der Eingänge 1, 2 und 3 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.

Programmstart und –stopp über Eingang (im Mastergerät)

EBSx	Eingang für Programmstopp setzen. x=0 bis 16(OMC) oder 32(TMC). Eingang stoppt das Programm, wenn er auf 0 geht.
EBR	Eingang über Schnittstelle zurücklesen
ESSx	Eingang für Programmstart setzen. x=0 bis 16(OMC) oder 32(TMC). Eingang startet das Programm wieder, wenn er auf 1 geht und Programmstopp wieder auf 1 steht..
ESR	Eingang über Schnittstelle zurücklesen

Befehl	*Bedeutung*
	Eingangsgruppe lesen
EGn**R**	Die Eingangsgruppe n lesen. (NUR PC) n=1 bis 16
	Antwort : <STX><ACK>nnnnnnnnnnnnnnnn<ETX><CR><LF> n = 0 Eingang nicht gesetzt n = 1 Eingang gesetzt
	Eingangszustand lesen
ERnn;mm;xx	Der Zustand der Eingänge nn, mm, xx wird gelesen. (nur PC)
	Antwort : <STX><ACK>nnn<ETX><CR><LF> n = 0 Eingang nicht gesetzt n = 1 Eingang gesetzt
	Wichtig: Zwischen den Eingangsnummern ein **;** setzen

2.6 Programmbeeinflussung bei Nothalt (NUR PROG)

Befehl	*Bedeutung*
FE	Rücksetzen des Fehlerbytes bei SFI (Schrittfehlererkennung).
FNznr	Es wird die Zeile, an der das Programm beim Nothalt fortfahren soll, festgelegt.
FN* la**	Es wird die Zeile, an der das Programm beim Nothalt fortfahren soll, durch ein Label bestimmt.
FP[name**]**	Programmangabe für den Nothalt. Beim Nothalt wird in das Programm mit dem Namen name gesprungen.

2.7 Programmunterbrechungen

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

2.8 Systemanpassung im Programmablauf

Befehl	*Bedeutung*
	Achsenanzahl
IAR	Die Anzahl der vorhandenen Achsen auslesen. (NUR PC) **Antwort: <STX><ACK>n<ETX><CR><LF>**
	Automatikstart
IBSname	Der Programmname wird in das Autostartregister geschrieben. Mit diesem Programm wird gestartet, wenn der REMOTE/LOCAL-Schalter auf LOCAL steht. **Antwort: <STX><ACK><ETX><CR><LF>** oder **<STX><NAK><ETX><CR><LF>** (NUR PC)
IBR	Der Programmname für den Autostart wird ausgelesen. (NUR PC) **Antwort: <STX><ACK>name<ETX><CR><LF>**
	Baudrate einstellen/lesen (NUR PC)
ICn**Sbaud**	Die Baudrate für die OMC/TMC Schnittstellen setzen. n = 1 → COM 1, n = 2 → COM 2 von OMC/TMC baud = Baudrate (9600, 19200, 38400, 57600 oder 115200 Baud)
ICn**R**	Baudraten-Einstellung der OMC/TMC Schnittstellen wird ausgelesen. n = 1 → COM 1, n = 2 → COM 2 von OMC/TMC
	Endstufenfehlerüberwachung setzen
IESn	n=0 : Endstufenfehlerüberwachung wird abgeschaltet n=1: Endstufenfehlerüberwachung wird angeschaltet
	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><ETX><CR><LF>**
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><ETX><CR><LF>**
	Inhaltsverzeichnis RAM
IPn	n-ten Programmnamen der Programmliste vom RAM auslesen. Wenn kein Programmname vorhanden ist, kommt als Antwort NAK (NUR PC).

Befehl	*Bedeutung*
	name (8 Zeichen), Zeilenanzahl (5 Zeichen)
	Antwort: <STX><ACK>nameZeilenanzahl_Zeile(n)<ETX><CR><LF> **Antwort: <STX><NAK><ETX><CR><LF>** wenn kein Name vorhanden
	Information des Übertragungsprotokolls (ab OMC/TMC-Systemsoftware V2.71)
ITR	Zustand des RS-Schnittstellen-Übertragungsprotokoll lesen
ITSn	RS-Schnittstellen-Übertragungsprotokoll definieren n=0 Übertragung des Befehls **ohne** Prüfsumme n=1 Übertragung des Befehls **mit** Prüfsumme
	Information Bedienterminal
ITTSn	Bedienterminaltyp bestimmen n=0 ohne Bedienterminal fahren n=1 mit Bedienterminal BT5 fahren n=2 mit Bedienterminal TP11 fahren
	Versionsabfrage
IVR	Softwareversion der Steuerung auslesen (NUR PC). **Antwort: <STX><ACK>Software Version<ETX><CR><LF>**
	Speicherplatz Test
IZ	Freien Speicherplatz im EPROM in Zeilen auslesen (NUR PC). **Antwort: <STX><ACK>wert Zeile(n) frei<ETX><CR><LF>** wert = Anzahl der freien Programmzeilen im EPROM

2.9 Interpolationsbefehle

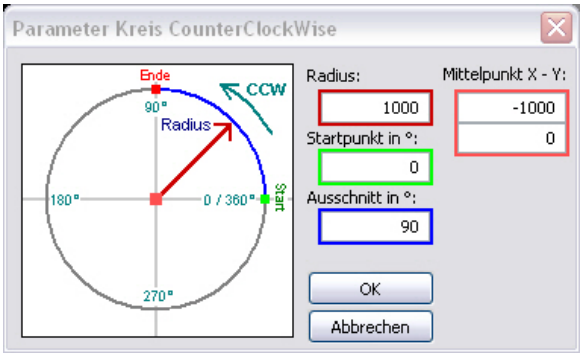
Linearinterpolation

IR Linearinterpolation rücksetzen.

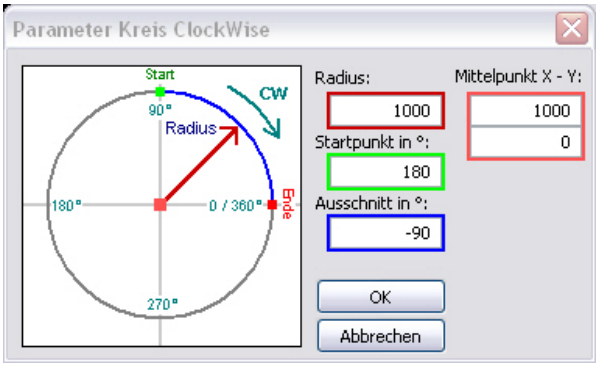
IS Linearinterpolation setzen. Der Befehl bleibt so lange gesetzt, bis der Befehl IR kommt. Es werden immer zwei aufeinander folgende Achsen in einer Zeile interpoliert.

Antwort: <STX><ACK><ETX><CR><LF> (NUR PC)

Zirkularinterpolation



gegen Uhrzeigersinn (CCW)



im Uhrzeigersinn (CW)

KRn Radius n des Kreisbogens setzen,
Einheit und Faktor von n sind in P2 und P3 definiert (siehe Kap. 5)

KSn Startpunkt n auf der Kreisbahn in Grad (°) setzen
n =0 bis 360°

KWn Weg (Ausschnitt) n in Grad (°) vom Startpunkt setzen
n =0 bis 360° (CCW)
n =0 bis -360° (CW)

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

Beispiel: KR100 KS90 KW180

2.10 Sprungbefehle (NUR PROG)

Befehl	*Bedeutung*
	Sprungbefehle relativ
N+nn **N–**nn	Relativer Sprung um nn Zeilen vorwärts (+) bzw. rückwärts (–).
N+Rnn **N–R**nn **N+R[R**nn**]** **N–R[R**nn**]**	Relativer Sprung vorwärts (+) bzw. rückwärts (–) um die Anzahl der Zeilen, die im Register nn angegeben sind.
	Sprungbefehle absolut
Nnn	Absoluter Sprung zur Zeile nn.
N*la****	Absoluter Sprung zur Zeile, die durch das Label *la* gekennzeichnet ist.
NRnn **NR[R**nn**]**	Absoluter Sprung zur Zeile, die durch den Inhalt des Registers nn angegeben wird.
NP[name**]**	Sprung zu einem Programm mit dem Namen name. Beginn ab Zeile 1.
NP[name**]N**nn	Sprung zu einem Programm mit dem Namen name. Beginn ab Zeile nn.
NP[name**]NR**nn **NP[**name**]NR[R**nn**]**	Sprung zu einem Programm mit dem Namen name. Beginn ab der Zeile, die durch den Inhalt des Registers nn angegeben wird.
NP[name**]N***la****	Sprung zu einem Programm mit dem Namen name. Beginn ab der Zeile, die durch das Label ***la*** gekennzeichnet ist.
	Bedingter Sprung relativ E = Bedingung erfüllt
NE+nn **NE–**nn	Relativer Sprung um nn Zeilen vorwärts.
NE+Rnn **NE–R**nn **NE+R[R**nn**]** **NE–R[R**nn**]**	Relativer Sprung vorwärts um nn Zeilen vorwärts (+) bzw. rückwärts (–) um die Anzahl der Zeilen, die im Register nn angegeben sind.
	Bedingter Sprung absolut E = Bedingung erfüllt
NEnn	Absoluter Sprung zur Zeile nn.
NE*la****	Absoluter Sprung zur Zeile, die durch das Label ***la*** gekennzeichnet ist.
NERnn **NER[R**nn**]**	Absoluter Sprung zur Zeile, die durch den Inhalt des Registers nn angegeben wird.
NEP[name**]**	Sprung zu einem Programm mit dem Namen name. Beginn ab Zeile 1.

Befehl	*Bedeutung*
NEP[name**]N**nn	Sprung zu einem Programm mit dem Namen name. Beginn ab Zeile nn.
NEP[name**]NR**nn **NEP[**name**]NR[R**nn**]**	Sprung zu einem Programm mit dem Namen name. Beginn ab der Zeile, die durch den Inhalt des Registers nn angegeben wird.
NEP[name**]N*** la****	Sprung zu einem Programm mit dem Namen name. Beginn ab der Zeile, die durch das Label ***la*** gekennzeichnet ist.
	Bedingter Sprung relativ N = Bedingung nicht erfüllt
NN+nn **NN–**nn	Relativer Sprung um nn Zeilen vorwärts (+) bzw. rückwärts (–).
NN+Rnn **NN–R**nn **NN+R[R**nn**]** **NN–R[R**nn**]**	Relativer Sprung vorwärts (+) bzw. rückwärts (–) um die Anzahl der Zeilen, die im Register nn angegeben sind.
	Bedingter Sprung absolut N = Bedingung nicht erfüllt
NNnn	Absoluter Sprung zur Zeile nn.
NN* la****	Absoluter Sprung zur Zeile, die durch das Label *la**** gekennzeichnet ist.
NNRnn **NNR[R**nn**]**	Absoluter Sprung zur Zeile, die durch den Inhalt des Registers nn angegeben wird.
NNP[name**]**	Sprung zum Programm mit dem Namen name. Beginn ab Zeile 1.
NNP[name**]N**nn	Sprung zum Programm mit dem Namen name. Beginn ab Zeile nn.
NNP[name**]NR**nn **NNP[**name**]NR[R**nn**]**	Sprung zum Programm mit dem Namen name. Beginn ab der Zeile, die durch den Inhalt des Registers nn angegeben wird.
NNP[name**]N*** la****	Sprung zum Programm mit dem Namen name. Beginn ab der Zeile, die durch das Label ***la*** gekennzeichnet ist.

2.11 Programmzeilen wiederholen

Befehl	*Bedeutung*
NWnn	Die Programmzeile nn mal wiederholen.
NWRnn **NWR[R**nn**]**	Die Programmzeile so oft wiederholen, wie es im Register nn angegeben ist.
	Antwort: <STX><ACK><ETX><CR><LF>

2.12 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.
	Achsenstopp und -start
PS	Alle Achsen stoppen und Fahrbefehl merken.
PR	Alle Achsen starten, die durch **PS** gestoppt wurden.

2.13 Programm- und Dateiverwaltung (NUR PC)

Befehl	*Bedeutung*
	Programme kopieren
QCPname1 name2	Das Programm mit dem Namen name1 wird in das Programm mit den Namen name2 kopiert.
	Programme und Dateien löschen
QDPname	Das Programm mit dem Namen name wird gelöscht. .
QDP*.*	Es werden alle Programme im RAM-Speicher gelöscht.
QDR	Es werden alle Register im RAM auf null gesetzt.
	Programmzeile schreiben
QPname **N**nn**S**befehle	Es wird die Zeile nn des Programms name mit den Befehlen beschrieben. Es können maximal 32 Zeichen beschrieben werden.
	Programmzeile lesen
QPname **N**nn**R**	Es wird die Zeile nn des Programms name ausgelesen.
	Programmstart ab Zeile
QPname **N**nn**A**	Das Programm name wird ab der Zeile nn gestartet.
	Programm Abbruch
QPE	Wird der QPE Befehl vom Rechner gesendet, dann wird in die Programmebene zurückgesprungen, aus der das Programm gestartet wurde.

Befehl	*Bedeutung*
	Programm umbenennen
QRPname1 name2	Das Programm mit dem Namen name1 wird in das Programm mit den Namen name2 umbenannt.
	Programmübertragung mit Abfrage
QPname **S**znr	Das Programm mit dem Namen name soll im Block übertragen werden. Bei der Übertragung des Programms muss die gesamte Steuersequenz eingehalten weden. Der Name muss 8 Zeichen und jede Zeile 32 Zeichen lang sein. Die Zeilennummer wird nicht übertragen. name → maximal 8 Zeichen znr → Anzahl der zu übertragenen Zeilen 1. vom Rechner: **<STX>QPname_Sznr<ETX><CR><LF>** Die Programmübertragungssequenz wird gestartet. 2. Antwort der Steuerung: **STX><ACK>O<ETX><CR><LF>** wenn das Programm nicht auf der Steuerung vorhanden ist und das Programm in den RAM-Speicher der Steuerung passt. 3. vom Rechmer: **<STX><ACK><ETB>name_Programm<EOT><ETX><CR><LF>** Der Programmname und das Programm werden zusammenhängend übertragen. Kein Leerzeichen zwischen Namen und Programm. 4. Antwort der Steuerung: **<STX><ACK><ETX><CR><LF>** Die Programmübertragung wurde erfolgreich abgeschlossen. 5. vom Rechner: **<STX>QPname_Sznr<ETX><CR><LF>** Die Programmübertragungssequenz wird gestartet. 6. Antwort der Steuerung: **<STX><ACK>K<ETX><CR><LF>** wenn das Programm nicht auf der Steuerung vorhanden ist und das Programm nicht in den RAM-Speicher der Steuerung passt. Die Übertragung wird beendet. 7. vom Rechner: **<STX>QPname_Sznr<ETX><CR><LF>** Die Programmübertragungssequenz wird gestartet.

Befehl	*Bedeutung*
	8. Antwort der Steuerung: **<STX><ACK>V<ETX><CR><LF>** wenn das Programm auf der Steuerung vorhanden ist. 9. Rechner: **<STX>J<ETX><CR><LF>** Das Programm auf der Steuerung soll mit dem Programm im Rechner überschrieben werden. Weiter bei Punkt 2 10. vom Rechner: **<STX>QPname_Sznr<ETX><CR><LF>** Die Programmübertragungssequenz wird gestartet. 11. Antwort der Steuerung: **<STX><ACK>V<ETX><CR><LF>** wenn das Programm auf der Steuerung vorhanden ist. 12. vom Rechner: **<STX>N<ETX><CR><LF>** Das Programm auf der Steuerung soll nicht mit dem Programm in Rechner überschrieben werden. 13. Antwort der Steuerung: **<STX><ACK><ETX><CR><LF>** Die Programmübertragung wurde beendet.
	Programm lesen mit Abfrage
QPname_**R**	Das Programm mit dem Namen name soll aus der Steuerung ausgelesen werden. Der Programmname ist 8 Zeichen und jede Programmzeile 32 Zeichen lang. Der Programmname und die Zeilen werden in einen Block übertragen. **Abfrage und senden** 1. vom Rechner: **<STX>QPname_R<ETX><CR><LF>** Das Programm mit dem Namen name soll gelesen werden. 2. von der Steuerung: **<STX><ACK>Oznr<ETX><CR><LF>** Die Steuerung meldet ein O und die Anzahl der Programmzeilen, wenn das Programm vorhanden ist. 3. vom Rechner: **<STX>J<ETX><CR><LF>** Der Rechner gibt die Meldung, dass das Programm übertragen

Befehl	*Bedeutung*
	werden kann.
	4. von der Steuerung:
	<STX><ACK><ETB>name_Programm<EOT><ETX><CR><LF> Der Programmname und das Programm werden zusammenhängend übertragen. Kein Leerzeichen zwischen Namen und Programm. Die Zeilennummer wird bei der Übertragung nicht mitgesendet. Mit dem Steuerzeichen EOT wird das Programmende markiert. Die Steuerzeichen ETX, CR, LF schließen die Übertragung ab.
	Abfrage und Abbruch vom Rechner
	5. vom Rechner:
	<STX>QPname_R<ETX><CR><LF> Das Programm mit dem Namen name soll gelesen werden.
	6. von der Steuerung:
	<STX><ACK>Oznr<ETX><CR><LF> Die Steuerung meldet ein O und die Anzahl der Programmzeilen, wenn das Programm vorhanden ist.
	7. vom Rechner:
	<STX>N<ETX><CR><LF> Der Rechner meldet, dass das Programm nicht übertragen werden soll. Die Befehlssequenz wird beendet.
	Abfrage und Abbruch von der Steuerung
	8. vom Rechner:
	<STX>QPname_R<ETX><CR><LF> Das Programm mit dem Namen name soll gelesen werden.
	9. von der Steuerung:
	<STX><ACK>K<ETX><CR><LF> Das Programm mit dem Namen name ist nicht vorhanden. Die Befehlssequenz wird beendet.

2.14 Registerbefehle

Befehl	*Bedeutung*
	Registerinhalt ganzzahlig
Rnn.z	Die Nachkommastellen des Registerinhalts nn löschen ohne Rundung des Wertes. z = 0 – 6 Nachkommastellen
	Registerinhalt binär ausgeben
Rnn**BA**nn–mm **R[R**nn**]BA**nn–mm	Der Inhalt des Registers nn wird binär an die Ausgänge nn bis mm ausgegeben.
	Register binär beschreiben (über Eingänge)
Rnn**BE**nn–mm **R[R**nn**]BE**nn–mm	Die Eingänge nn bis mm in den Inhalt des Registers nn binär einlesen. **Beispiel: R1BE1–8** Für unser Beispiel haben die Eingänge 1 bis 8 folgenden Zustand **1010 0101.** Das Register 1 wird nun mit dem Binärwert der Eingänge beschrieben. Nach dem Befehl hat das Register 1 den Wert 165.
	Register hexadezimal beschreiben
Rnn**BS**zwert **R[R**nn**]BS**zwert	Den Inhalt des Registers nn binär mit dem Wert zwert laden. 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 bitweise schieben
Rnn**BL**m **R[R**nn**]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 27 → 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><ETX><CR><LF>** (NUR PC)
Rnn**BR**m **R[R**nn**]BR**m	Der Inhalt des Registers nn wird binär um m Stellen nach rechts (→ LSB) geschoben. Es wird links mit 0 aufgefüllt.

Befehl	*Bedeutung*
	m = 1 bis 27 → 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 **R[R**nn**]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 27 → Maximalwert des Registerinhaltes **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 gesetzt da das 4. Bit auf 1 steht. **Antwort: <STX><ACK> E <ETX><CR><LF>** oder **<STX><ACK> N <ETX><CR><LF>** (NUR PC)
	Register logische Verknüpfung
	Und Verknüpfung
Rnn**B^**zwert **R[R**nn**]B^**zwert	Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert UND verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt. **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**	**Bin**
	680	**2A8**	**1010101000**
	416	**1A0**	**0110100000**
Ergebnis	**160**	**0A0**	**0010100000**

Rnn**B^R**mm **R[R**nn**]B^R**mm **R**nn**B^R[R**mm**]** **R[R**nn**]B^R[R**mm**]**	Den Inhalt des Registers nn mit dem Inhalt des Registers mm UND verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt.

Befehl *Bedeutung*

Oder Verknüpfung

Rnn**Bv**zwert
R[Rnn**]Bv**zwert

Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert ODER verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt.

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.

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

Rnn**BvR**mm
R[Rnn**]BvR**mm
Rnn**BvR[R**mm**]**
R[Rnn**]BvR[R**mm**]**

Den Inhalt des Registers nn mit dem Inhalt des Registers mm ODER verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt.

Antwort: <STX><ACK> <ETX><CR><LF> (NUR PC)

XOR Verknüpfung

Rnn**BX**zwert
R[Rnn**]BX**zwert

Den Inhalt des Registers nn mit dem hexadezimalen Wert zwert XOR verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt.

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**	**Bin**
	680	**2A8**	**1010101000**
	416	**1A0**	**0110100000**
Ergebnis	**776**	**308**	**1100001000**

Rnn**BXR**mm
R[Rnn**]BXR**mm
Rnn**BXR[R**mm**]**
R[Rnn**]BXR[R**mm**]**

Den Inhalt des Registers nn mit dem Inhalt des Registers mm XOR verknüpfen. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis null ist, sonst wird es rückgesetzt.

Registerinhalte vergleichen (mit einem Zahlenwert)

Rnn=zwert
R[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.

Befehl	*Bedeutung*
Rnn#zwert **R[R**nn**]**#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 **R[R**nn**]>**zwert	Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis größer ist, sonst wird es rückgesetzt.
Rnn**<**zwert **R[R**nn**]<**zwert	Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner ist, sonst wird es rückgesetzt.
	Registerinhalte vergleichen (mit einem Achsenparameter)
Rnn**=XP**mm **R[R**nn**]=XP**mm	Vergleich des Inhalts von Register nn mit dem Parameter mm der Achse X. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis gleich ist, sonst wird es rückgesetzt. mm = Parameternummer
Rnn**#XP**mm **R[R**nn**]#XP**mm	Vergleich des Inhalts von Register nn mit dem Parameter mm der Achse X. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis ungleich ist, sonst wird es rückgesetzt. mm = Parameternummer
Rnn**>XP**mm **R[R**nn**]>XP**mm	Vergleich des Inhalts von Register nn mit dem Parameter mm der Achse X. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis größer ist, sonst wird es rückgesetzt. mm = Parameternummer
Rnn**<XP**mm **R[R**nn**]<XP**mm	Vergleich des Inhalts von Register nn mit dem Parameter mm der Achse X. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner ist, sonst wird es rückgesetzt. mm = Parameternummer
	Registerinhalte miteinander vergleichen
Rnn**=R**mm **R[R**nn**]=R**mm **R**nn**=R[R**mm**]** **R[R**nn**]=R[R**mm**]**	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis gleich ist ist, sonst wird es rückgesetzt.
Rnn**#R**mm **R[R**nn**]#R**mm **R**nn**#R[R**mm**]** **R[R**nn**]#R[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[R**nn**]>R**mm **R**nn**>R[R**mm**]** **R[R**nn**]>R[R**mm**]**	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis größer ist, sonst wird es rückgesetzt.

Befehl	*Bedeutung*
Rnn**<R**mm **R[R**nn**]<R**mm **R**nn**<R[R**mm**]** **R[R**nn**]<R[R**mm**]**	Vergleich des Inhalts von Register nn mit dem Inhalt von Register mm. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner ist, sonst wird es rückgesetzt. **Antwort bei allen Vergleichen:** **<STX><ACK> E <ETX><CR><LF>** oder **<STX><ACK> N <ETX><CR><LF>** (NUR PC)
	Rechenoperationen mit Registern
	Addieren
Rnn**+**zwert **R[R**nn**]+**zwert	Zum Inhalt des Registers nn wird der Wert zwert addiert.
Rnn**+R**mm **R**nn**+R[R**mm**]** **R[R**nn**]+R**mm **R[R**nn**]+R[R**mm**]**	Zum Inhalt des Registers nn wird der Inhalt des Registers mm addiert.
	Subtrahieren
Rnn**–**zwert **R[R**nn**]–**zwert	Vom Inhalt des Registers nn wird der Wert zwert subtrahiert.
Rnn**–R**mm **R**nn**–R[R**mm**]** **R[R**nn**]–R**mm **R[R**nn**]–R[R**mm**]**	Vom Inhalt des Registers nn wird der Inhalt des Registers mm subtrahiert.
	Multiplizieren
Rnn*****zwert **R[R**nn**]***zwert	Der Inhalt des Registers nn wird mit dem Wert zwert multipliziert.
Rnn***R**mm **R[R**nn**]*R**mm **R**nn***R[R**mm**]** **R[R**nn**]*R[R**mm**]**	Der Inhalt des Registers nn wird mit dem Inhalt des Registers mm multipliziert.
	Dividieren
Rnn**:**zwert **R[R**nn**]:**zwert **R**nn**/**zwert **R[R**nn**]/**zwert	Der Inhalt des Registers nn wird durch den Wert zwert dividiert.
Rnn**:R**mm **R**nn**:R[R**mm**]** **R[R**nn**]:R**mm **R[R**nn**]:R[R**mm**]**	Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.
	Der Inhalt des Registers nn wird durch den Wert zwert dividiert.
Rnn**/R**mm **R**nn**/R[R**mm**]** **R[R**nn**]/R**mm **R[R**nn**]/R[R**mm**]**	Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.

Befehl	*Bedeutung*
	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.
	Quadratwurzel
Rnn**QW**	Aus dem Wert des Registers nn wird die Quadratwurzel berechnet und in das Register nn zurückgeschrieben.
	Zufallszahl
Rnn**RAND**	Das Register nn wird mit Zufallszahl im Bereich 0 bis 32767 beschrieben.
	Register auslesen
Rnn**R** **R[R**nn**]R**	Der Inhalt des Registers nn wird ausgelesen. (NUR PROG) **Antwort: <STX><ACK>wert<ETX><CR><LF>** **Antwort bei allen Rechenoperationen:** **<STX><ACK><ETX><CR><LF>** (NUR PC)
	Register beschreiben:
	mit Dezimalwerten
Rnn**S**zwert **R[R**nn**]S**zwert	Das Register nn mit dem Zahlenwert zwert beschreiben.
	mit Registerinhalten
Rnn**SR**mm **R[R**nn**]SR**mm **R**nn**SR[R**mm**]** **R[R**nn**]SR[**mm**]**	Das Register nn mit dem Inhalt des Registers mm beschreiben.
	mit Parameterwerten
Rnn**SXP**mm **R[R**nn**]SXP**mm	Das Register nn mit dem Parameter mm der Achse X beschreiben.
	mit Zeilennummer Nothalt
Rnn**SN** **R[R**nn**]SN**	Das Register nn mit der Zeilennummer, in der ein Nothalt ausgelöst wurde, beschreiben.

Beispiel :

```
ZNR 001 FN10
ZNR 002 X+1000
ZNR 003 X–1000 H N2
ZNR 010 R1SN
```

In diesem Beispiel wird ein Nothaltprogramm ab Zeile 10 definiert. Die X-Achse fährt 1000 Schritte in Plus – und Minus - Richtung. Tritt beim Fahren der Achse ein Nothalt auf, wird in die Zeile 10 gesprungen und die Achse wird gestoppt. Beim Befehl **R1SN** wird nun das Register 1

Befehl	*Bedeutung*
	mit der Zeilennummer beschrieben, in der der Nothalt ausgelöst wurde. Nun kann ausgewertet werden, bei welcher Positionierung ein Nothalt aufgetreten ist.
	mit dem Timertickerwert
Rnn**STT**	Das Register nn mit dem Timertickerwert beschreiben.
	mit der Zeilennummer
Rnn**SZ**	Das Register nn mit der Zeilennummer, in der dieser Befehl aufgerufen wird, beschreiben.
R[Rnn**]SZ**	Das Register, das durch Register nn adressiert wird, mit der Zeilennummer, in der dieser Befehl aufgerufen wird, beschreiben.
	Beispiel : ZNR 041 R1S10 **ZNR 042 R[R1]SZ** In diesem Beispiel wird das Register 1 mit dem Wert 10 beschrieben. Beim Befehl **R[R1]SZ** wird nun das Register 10 mit der aktuellen Zeilennummer beschrieben. Der Inhalt des Registers 10 beträgt nun 42. Dieser Befehl kann für Automatikstartfunktionen genutzt werden.
	Über Eingänge
Rnn**SE**mm–xx.k **R[R**nn**]SE**mm–xx.k	Das Register nn mit einem BCD Wert über die Eingänge mm bis xx beschreiben. Der Wert wird mit k Nachkommastellen ins Register geschrieben.
	Beispiel: R1SE1–8.1 Für unser Beispiel haben die Eingänge 1 bis 8 folgende Zustände: **1001 0011**. Das Register 1 wird nun im BCD-Format mit den Werten der Eingänge beschrieben. Nach dem Befehl hat das Register 1 den Wert 9,3.
	Register mit Terminal ändern
Rnn**ST**	Das Register nn in Zeile 4 ab Position 10 anzeigen. Eingabe neuer Daten an der Cursorposition. Mit Drücken der ENTER-Taste am Terminal wird das Register nn neu beschrieben.
	Beispiel : R41ST In diesem Beispiel wird das Register 41 in der 4. Zeile des Terminaldisplays ab der 10. Position angezeigt und steht zum Editieren bereit.
	Antwort: <STX><ACK><ETX><CR><LF>wenn Terminal vorhanden **<STX><NAK><ETX><CR><LF> wenn kein Terminal vorhanden** (NUR PC)
Rnn**ST.**z	Das Register nn in Zeile 4 anzeigen mit z Nachkommastellen (z=0 bis 6). Eingabe neuer Daten an der Cursorposition. Mit Drücken der

Befehl	*Bedeutung*
	ENTER-Taste am Terminal wird das Register nn neu beschrieben. **Beispiel : R2ST.6** Das Register 2 wird mit 6 Nachkommastellen am Terminal angezeigt und neu beschrieben.
Rnn**ST**y	Das Register nn in Zeile y anzeigen (y=1 bis 4) und nach Neueingabe mit ENTER-Taste neu beschreiben. **Beispiel : R2ST3** Das Register 2 wird in der 3.Zeile am Terminal ab Position 1angezeigt und neu beschrieben.
Rnn**ST**y.z	Das Register nn in Zeile y an Position 1 anzeigen (y=1 bis 4) mit z Nachkommastellen (z=0 bis 6) und nach Neueingabe mit ENTER-Taste neu beschreiben. **Beispiel : R2ST3.4** Das Register 2 wird in der 3.Zeile mit 4 Nachkommastellen ab Postion 1 am Terminal angezeigt und neu beschrieben.
Rnn**ST**y;m	Das Register nn in Zeile y (y=1 bis 4). ab Position m (m=1 bis 20) anzeigen und nach Neueingabe mit ENTER-Taste neu beschreiben. **Beispiel : R1ST3;7** Das Register 1 wird in der 3.Zeile ab Position 7 angezeigt und neu beschrieben.
Rnn**ST**y;m.z	Das Register nn in Zeile y (y=1 bis 4) ab Position m (m=1 bis 20) mit z (z=0 bis 6) Nachkommastellen anzeigen und nach Neueingabe mit ENTER-Taste neu beschreiben. **Beispiel : R2ST2;2.6** Das Register 2 wird in der 2.Zeile ab Position 2 mit 6 Nachkommastellen angezeigt und neu beschrieben.
	mit A/D-Wandlerwerten (Option OMC/TMC)
Rnn**SAD**x**C**y **R[R**nn**]SAD**x**C**y	Das Register nn mit dem Wert des A/D-Wandlers beschreiben. x =0 oder 1: Adresse des A/D-Wandlers, y=1 bis 4: Kanal des A/D-Wandlers.
	Register mit Terminal anzeigen
Rnn**W**y	Den Registerwert nn in Zeile y ab Position 1 anzeigen (y=1 bis 4). **Beispiel : R2W2** Das Register 2 wird in der 2.Zeile ab der 1.Position angezeigt.
Rnn**W**y.z	Den Registerwert nn in Zeile y ab Position 1 mit Nachkommastellen z (y=1 bis 4, z= 0 bis 6) anzeigen. **Beispiel : R1W4.6** Das Register 1 wird in der 4.Zeile ab der 1.Position mit 6 Nachkommastellen angezeigt.

Befehl	*Bedeutung*
Rnn**W**y;m	Den Registerwert nn in Zeile y ab Position m (y=1 bis 4, m= 1 bis 20) anzeigen. **Beispiel : R2W3;5** Das Register 2 wird in der 3.Zeile ab der 5.Position angezeigt.
Rnn**W**y;m.z	Den Registerwert nn in Zeile y ab Position m mit z Nachkommastellen (y=1 bis 4, m= 1 bis 20, z=0 bis 6) anzeigen. **Beispiel : R7W2;5;3** Das Register 7 wird in der 2.Zeile ab der 5.Position mit 3 Nachkommastellen angezeigt. **Antwort: <STX><ACK><ETX><CR><LF>** (NUR PC)

2.15 Systemstatus (NUR PC)

Befehl	*Bedeutung*
	Systemstatus Allgemein
S	Achsentest mit Ausgabe der Anzahl der Achsen. **Antwort:<STX><ACK>n IO <ETX><CR><LF>** n = Anzahl der Achsen
	Systemstatus binär
SB	Den Status der Steuerung auslesen. Der Status wird im achtstelligen Binärformat (d_B = 0 oder 1) ausgegeben. **Antwort: <STX><ACK>d_{B8}..... d_{B1}<ETX><CR><LF>** d_B 1 = 1 → Programmausführung d_B 2 = 1 → Software Remote d_B 3 = 1 → Nothaltendschalter einer Achse d_B 4 = 1 → Endstufenfehler einer Achse d_B 5 = 1 → Programmierfehler (wird nach Status auslesen rückgesetzt) d_B 6 = 1 → Terminal aktiv d_B 7 = 1 → Eingangsabfrage aktiv d_B 8 = 1 → Rechneraufruf
	Systemstatus erweitert
SE	Den Status der Steuerung auslesen. Der Status wird im Hexadecimal Code ausgegeben. Pro Achse stehen zwei Bytes (4 Hexadezimal Digits d_H) zur Verfügung: 1. + 2. Byte für die X-Achse, 3. + 4. Byte für die Y-Achse. **Antwort:<STX><ACK>$d_{HX}d_{HX}d_{HX}d_{HX}d_{HY}d_{HY}d_{HY}d_{HY}$<ETX><CR><LF>** Bit 0 = 1 → Endstufe Überstrom Bit 1 = 1 → Endstufe Unterspannung Bit 2 = 1 → Endstufe Übertemperatur

Befehl *Bedeutung*

Bit 3 = 1 → Endstufe aktiviert
Bit 4 = 1 → Initiator minus aktiv (Nothalt)
Bit 5 = 1 → Initiator plus aktiv
Bit 6 = 1 → Schrittfehler (nur mit Option SFI = Step Failure Indication)
Bit 7 = 1 → Encoder Fehler
Bit 8 = 1 → Motor steht
Bit 9 = 1 → Referenzpunkt angefahren und OK (wird bei Stopp über Initiator rückgesetzt)
(Bit 10 bis Bit 15 nicht belegt)
Wenn Bit 0 bis Bit 2 gleichzeitig gesetzt sind, ist keine Endstufe angeschlossen. Andernfalls kann immer nur ein Fehler zur Zeit anstehen.

Fehlermeldung Slave

SF1 Fehlermeldung Slave aktivieren. Wenn ein Fehler im Slave auftritt, wird die Meldung ERROR an den Master geschickt. Alle Achsen werden gestoppt.

SF0 Fehlermeldung Slave ausschalten

SH **Systemstatus Achsen**

Achsentest mit Ausgabe, ob Achsen stehen.

Antwort:<STX><ACK> E <ETX><CR><LF>, wenn alle Achsen stehen.
<STX><ACK> N <ETX><CR><LF>, wenn 1 oder mehr Achsen laufen.

Systemstatus dezimal

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

Antwort: <STX><ACK>wert <ETX><CR><LF>
wert = Zahlenwert zwischen 0 und 255
0 = Programmende im LOCAL-Betrieb.
1 = Programmausführung
2 = Software Remote
4 = Nothaltendschalter einer Achse
8 = Endstufenfehler einer Achse
16 = Programmierfehler (wird nach Status auslesen rückgesetzt)
32 = Terminal aktiv
64 = Eingangsabfrage aktiv
128 = Rechnerbetrieb

Initiatoren

SUI Den Status der Initiatoren abfragen.

Antwort:<STX><ACK>I=n <ETX><CR><LF>
n = 0 → Achse ist frei, kein Initiator hat angesprochen

Befehl	*Bedeutung*
	n = + → Initiator Plusrichtung hat angesprochen n = – → Initiator Minusrichtung hat angesprochen n = 2 → beide Initiatoren haben angesprochen (d.h. falsche Polarität der Initiatoren, Drahtbruch oder keine 24 V Versorgungsspannung)
	Syncronstart
S1	Syncronstart der Achsen vorbereiten
S0	Syncronstart der Achsen ausführen

2.16 Daten ins Flash EPROM schreiben

Befehl	*Bedeutung*
	Programme und Achsenparameter speichern (NUR PC)
SA	Achsenparameter auf das Flash EPROM schreiben.
SP*.*	Programme auf das Flash EPROM speichern.

2.17 Zeitschleifen

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><ETX><CR><LF> (NUR PC)
TTSzwert **TTSR**nn **TTSR[R**nn**]**	Der Timer 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.
	Antwort: <STX><ACK><ETX><CR><LF> (NUR PC)
TT=0	Der Timer 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.
TT>zwert **TT>R**nn **TT>R[R**nn**]** **TT<**zwert **TT<R**nn **TT<R[R**nn**]**	Der Timer 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 <ETX><CR><LF> oder **<STX><ACK> N <ETX><CR><LF>** (NUR PC)

2.18 Unterprogramme (NUR PROG)

Befehl	*Bedeutung*
	Unterprogramm abbrechen
UA	Alle Unterprogramme werden abgebrochen und der Stack wird neu gesetzt. 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
Unn	Das Unterprogramm mit Beginn ab der Zeile nn wird aufgerufen. Beendet wird das Programm mit dem UE Befehl.
URnn **UR[R**nn**]**	Das Unterprogramm beginnt mit der Zeile, die durch den Inhalt des Registers nn bzw. [Rnn] aufgerufen wird. Beendet wird das Programm mit dem UE Befehl.
U* la*	Das Unterprogramm beginnt in der Zeile, die durch das Label ***la*** gekennzeichnet ist. Beendet wird das Unterprogramm mit dem UE Befehl.
UP[name**]**	Das Unterprogramm mit dem Namen name, Beginn ab der Zeile 1, wird aufgerufen. Beendet wird das Unterprogramm mit dem UE Befehl.
UP[name**]N**nn	Das Unterprogramm mit dem Namen name, Beginn ab der Zeile nn, wird aufgerufen. Beendet wird das Unterprogramm mit dem UE Befehl.
UP[name**]NR**nn **UP[**name**]NR[R**nn**]**	Das Unterprogramm mit dem Namen name beginnt mit der Zeile, die durch den Inhalt des Registers nn bzw. [Rnn] aufgerufen wird. Beendet wird das Programm mit dem UE Befehl.
UP[name**]N*** la****	Das Unterprogramm mit dem Namen name beginnt in der Zeile, die durch das Label *la**** gekennzeichnet ist. Beendet wird das Unterprogramm mit dem UE Befehl.
	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
UEnn	siehe **U**nn, S. 33
UERnn **UE[R**nn**]**	
UE*la*	siehe **U*** la*, S. 33
UEP[name**]**	siehe **UP[**name**]**, S. 33
UEP[name**]N**nn	
UEP[name**]NR**nn **UEP[**name**]NR[R**nn**]**	
UEP[name**]N***la*	siehe **UP[**name**]N***la******, S. 33
	„N“ für Bedingung nicht erfüllt
UNnn	siehe **U**nn, S. 33
UNRnn **UNR[R**nn**]**	
UN*la*	siehe **U*** la*, S. 33
UNP[name**]**	siehe **UP[**name**]**, S. 33
UNP[name**]N**nn	
UNP[name**]NR**nn **UNP[**name**]NR[R**nn**]**	
UNP[name**]N***la*	siehe **UP[**name**]N***la******, S. 33

2.19 Terminalbefehle (auch von PC bei Terminalanschluss)

Befehl	*Bedeutung*
<>Wy	Zeile y löschen (y=1 bis 4)
<text>Wy	Text in Zeile y anzeigen ab Position 1 (y=1 bis 4)
<text>Wy;m	Text in Zeile y ab Position m anzeigen (y=1 bis 4; M=1 bis 20)
	Antwort: <STX><ACK><ETX><CR><LF>

2.20 Achsenbefehle

Befehl	*Bedeutung*
XC **YC** n**C**	x-Achse rücksetzen y-Achse rücksetzen n Achse rücksetzen n=1 bis 8
	Antwort: <STX><ACK><ETX><CR><LF> (NUR PC)
	Achsenzustand abfragen
X=E **X#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 “Störung” abgefragt.
X=H **X#H**	Eine Achse auf Stillstand abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse steht (**=**) bzw. läuft (#), sonst wird es rückgesetzt.
X=I+ **X=I–**	Eine Achse auf Initiatorzustand abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse auf dem Initiator steht oder Initator nicht angeschlossen ist, sonst wird es rückgesetzt
X=M **X#M**	Eine Achse auf Endstufenfehler abfragen. Das Bedingungsbyte wird gesetzt, wenn ein Endstufenfehler ansteht (**=**) bzw. **nicht** ansteht (**=**), sonst wird es rückgesetzt. Es wird der Fehler **“Schrittfehler”** abgefragt. Der Fehler kann nur bei Steuerungen mit optionaler SFI-Karte auftreten, da diese eine Schrittfehlerüberwachung haben.
X=N **X#N**	Eine Achse auf Nothalt abfragen. Das Bedingungsbyte wird gesetzt, wenn die Achse auf einem Nothaltendschalter steht (**=**), bzw. **nicht** (#) auf einem Nothaltendschalter steht, sonst wird es rückgesetzt.
	Antwort: <STX><ACK>E<ETX><CR><LF> oder **<STX><ACK>N<ETX><CR><LF> (NUR PC)**
	Warten bis Position erreicht
X>zwert **X>R**nn **X>R[R**nn**]**	Die Achse X wird positioniert und das Programm wartet bei diesem Befehl bis der Zähler XP21 größer ist als der Wert zwert. Wenn der Wert größer ist oder die Achse steht, wird im Programm fortgefahren.
	Beispiel: **ZNR 005 XP21S0 XP14S2000 XL+** **ZNR 006 X>5000 XP14S1000** **ZNR 007 X>10000 XS XP14S2000**

Befehl	*Bedeutung*
	In diesem Beispiel soll die X 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 **X>5000** wird gewartet, bis die Achse die Position 5000 erreicht hat oder die Achse vorzeitig durch einen Nothalt gestoppt wird.
X<zwert **X<R**nn **X<R[R**nn**]**	Die Achse X wird positioniert und das Programm wartet bei diesem Befehl, bis der Zähler XP21 kleiner ist als der Wert zwert. Wenn der Wert größer ist oder die Achse steht, wird im Programm fortgefahren.
	Antwort: <STX><ACK><ETX><CR><LF> (NUR PC), wenn die Achse steht oder die Positionsbedingung erfüllt ist, sonst wird gewartet.
	Endstufe einer Achse schalten
	Aktivieren
XMA	Die Endstufe der Achse X wird aktiviert.
	Deaktivieren
XMD	Die Endstufe der Achse X wird deaktiviert.
	Achsenparameter
XPmm**R**	Der Parameter mm der Achse X wird ausgelesen.
	Antwort : <STX><ACK>wert<ETX><CR><LF> mm = Parameternummer (NUR PROG)
XPmm**S**zwert **XP**mm**SR**nn **XP**mm**SR[R**nn**]**	Der Parameter mm der Achse X wird mit dem vorgegebenen Wert (zwert, Inhalt von Register nn oder Register [Rnn]) geladen. mm = Parameternummer
	Referenzsuchlauf / Initialisierung
	Zur Initialisierung der Achse muss eine 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“.
X0–	Die Achse fährt den Initiator der — Richtung an.
X0+	Die Achse fährt den Initiator der + Richtung an.

X0–I	Die Achse fährt in Minusrichtung und stoppt mit dem Nullimpuls des **I**nkrementalencoders. Nur inkrementell, kein SSI Encoder!
X0+I	Die Achse fährt in Plusrichtung und stoppt mit dem Nullimpuls des **I**nkrementalencoders. Nur inkrementell, kein SSI Encoder!
	Antwort: <STX><ACK><ETX><CR><LF> (NUR PC)
X0–^I	Die Achse fährt den Initiator der —Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des **I**ncrementalencoders die Achse stoppt. Nur inkrementell, kein SSI Encoder!
X0 +^I	Die Achse fährt den Initiator der +Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des **I**nkrementalencoders die Achse stoppt. Nur inkrementell, kein SSI Encoder!
	Freier Lauf
XLr	Die Achse wird im freien Lauf gestartet. Sie läuft so lange, bis sie durch den Befehl XS oder von einem Endschalter gestoppt wird. r = + oder – für die Laufrichtung
	Relative Positionierung
Xrzwert **X**r**R**nn **X**r**R[R**nn**]**	Die vorgegebene Strecke (zwert , Inhalt von Register nn oder Register [Rnn]) wird relativ gefahren. r = + oder – für die Laufrichtung
	Mit Stoppbefehl über Eingang
Xrzwert**vE**nnz **X**r**R**nn**vE**nnz **X**r**R[R**nn**]vE**nnz	Es wird die vorgegebene Strecke (zwert , Inhalt von Register nn oder Register [Rnn]) relativ im Schleichgang gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt oder ein Endschalter die Positionierung abbricht. r = + oder – für die Laufrichtung z = S → Eingang gesetzt z = R → Eingang rückgesetzt
Xrzwert**vvE**nnz **X**r**R**nn**vvE**nnz **X**r**R[R**nn**]vvE**nnz	Es wird die vorgegebene Strecke (zwert, Inhalt von Register nn oder Register [Rnn]) relativ im Eilgang gefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt 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

XArzwert
XAr**R**nn
XAr**R[R**nn**]**

Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]) absolut, bezogen auf den mechanischen Nullpunkt MØP (XP20), angefahren.

r = + oder – für die Laufrichtung

mit Stoppbefehl über Eingang

XArzwert**vvE**nnz
XAr**R**nn**vvE**nnz
XAr**R[R**nn**]vvE**nnz

Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]), bezogen auf den mechanischen Nullpunkt MØP im Eilgang angefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt oder ein Endschalter die Positionierung abbricht.

r = + oder – für die Laufrichtung
z = S → Eingang gesetzt
z = R → Eingang rückgesetzt

Absolute Positionierung bezogen auf den ELØP

Xerzwert
XEr**R**nn
XEr**R[R**nn**]**

Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]) absolut, bezogen auf den elektronischen Nullpunkt ELØP, angefahren.

r = + oder– für die Laufrichtung
z = S → Eingang gesetzt
z = R → Eingang rückgesetzt

mit Stoppbefehl über Eingang

XErzwert**vvE**nnz
XEr**R**nn**vvE**nnz
XEr**R[R**nn**]vvE**nnz

Es wird die vorgegebene Position (zwert, Inhalt von Register nn oder Register [Rnn]) absolut, bezogen auf den elektronischen Nullpunkt ELØP im Eilgang angefahren. Die Positionierung wird vorzeitig gestoppt, wenn der Eingang nn den Zustand z einnimmt oder ein Endschalter die Positionierung abbricht.

r = + oder – für die Laufrichtung
z = S → Eingang gesetzt
z = R → Eingang rückgesetzt

Achsen Stopp

XS

Mit dem Befehl XS können alle Fahrbefehle abgebrochen werden. Die Achse stoppt mit der eingestellten Rampe.

XSN

Die Achse stoppt mit der eingestellten Nothalt-Rampe (Parameter P7).

2.21 Tastenabfrage Bedienterminal BT24 (auch von PC)

Befehl	*Bedeutung*
	Bedingte Tastenabfrage
#**vF**n	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
#**v**nmx	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: ZNR 005 #vH1? NN–0** Hier wird die Tastatur des BT24 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–0** wird zum Zeilenanfang der Zeile 5 gesprungen. **Wichtig:** Die EINGABE-Taste ist für eine Abfrage nicht definiert. **Antwort: <STX><ACK> E <ETX><CR><LF>** oder **<STX><ACK> N <ETX><CR><LF>** (NUR PC)

3 Minilog Befehle

4 DIN-Befehle

Das Steuerungsprogramm kann auch mit den DIN-Befehlen für Wegbedingungen und Zusatzfunktionen festgelegt werden. Diese nach DIN 66025 genormten Befehle können in einem Programm mit den MINILOG-Befehlen zusammen verwendet werden.

Befehl	*Bedeutung*
	G-Befehle (Wegbedingungen)
G00, **G0**	Punktsteuerungsverhalten Positionierung mit der größtmöglichen Geschwindigkeit (Eilgang) ohne Interpolation mit Parameter 14.
G01, **G1**	Linearinterpolation setzen
G04Tnn, **G4T**nn	Zeitlich vorbestimmte Programmunterbrechungen mit programmierter oder in der Steuerung festgelegter Dauer und automatischer Programmfortsetzung. n= in Sekunden mit Nachkommastellen Abbruch über Eingang 2
G05, **G5**	Warten auf Achsenstillstand aller Achsen, danach erst weiter im Programm
G20Lnn	Unbedingter Sprung auf Zeile nn
G20L+nn	Unbedingter Sprung um nn Zeilen in Plus-Richtung
G20L-nn	Unbedingter Sprung um nn Zeilen in Minus-Richtung
G20* label****	Unbedingter Sprung auf Label
G20L* label****	Unbedingter Sprung auf Label
G20LP[name]	Unbedingter Sprung ins Programm name in Zeile 1
G21z**L**nn	Bedingter Sprung auf Zeile nn z = E oder N
G21z**L+**nn	Bedingter Sprung um nn Zeilen in Plus Richtung z = E oder N
G21z**L-**nn	Bedingter Sprung um nn Zeilen in Minus Richtung z = E oder N
G21z***label****	Bedingter Sprung auf Label z = E oder N
G21z**L***label****	Bedingter Sprung auf Label z = E oder N
G21z**LP**[name]	Bedingter Sprung ins Programm name in Zeile 1 z = E oder N
G22Lnn	Unterprogrammaufruf des Unterprogramms nn Unterprogramm durch G98Lnn im Programm gekennzeichnet
G22* label****	Unterprogrammaufruf des Unterprogramms *label*
G22P[name]	Unterprogrammaufruf des Programms name

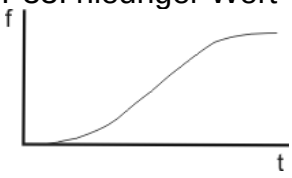
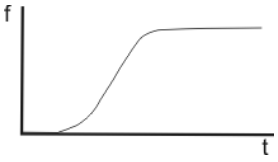
G23Lnn	Unterprogramm sofort beenden und Sprung zur Zeile nn
G23* label ****	Unterprogramm sofort beenden und Sprung zum Label
G74	Referenzlauf aller Achse minus Richtung
G74 x	Referenzlauf Achse x= X oder Y
G79Lnn	Automatischer Unterprogrammaufruf am Zeilenende Unterprogramm durch G98Lxx im Programm gekennzeichnet
G80	Beenden von G79
G90	Positionierung Absolutmaß bezogen auf Referenzpunktzähler Parameter 20
G91	Kettenmaß Positionierung
G92	Nullpunktverschiebung Parameter 20 setzen
G98Lnn	Unterprogramm Anfang und Deklaration nn Name des Unterprogramms maximal 6 Zeichen
G99	Unterprogramm Ende
	M-Befehle (Zusatzfunktionen)
M00, **M0**	Programmierter Halt Fortsetzung des Programms durch Setzen von Eingang 2
M01, **M1**	Programmierter Halt, wenn Eingang 3 eingeschaltet ist Fortsetzung des Programms durch Setzen von Eingang 2
M02, **M2**	Programmende
M03, **M3**	Einschalten der Spindeldrehung im Rechtslauf Ausgang 1 ein; Ausgang 2 aus
M04, **M4**	Einschalten der Spindeldrehung im Linkslauf Ausgang 1 aus; Ausgang 2 ein
M05, **M5**	Schnellstmögliches Anhalten der Spindeldrehung Ausgang 1 aus; Ausgang 2 aus
M07, **M7**	Kühlmittel 2 ein Ausgang 3 aus; Ausgang 4 ein
M08, **M8**	Kühlmittel 1 ein Ausgang 3 ein; Ausgang 4 aus
M09, **M9**	Kühlmittel aus Ausgang 3 aus; Ausgang 4 aus
M10	Klemmung ein; Ausgang 5 ein
M11	Klemmung aus; Ausgang 5 aus
M68	Werkstück spannen ; Ausgang 6 ein
M69	Werkstück entspannen ; Ausgang 6 aus

5 Parameter

Parameterliste		
Nr.	**Bedeutung**	**Auslieferungszustand**
P01	Art der Bewegung 0 rotatorisch Rundtisch, 1 Endschalter für Initialisierung 1 linear Lineartisch, 2 Endschalter: Initialisierung und Begrenzung –Richtung Begrenzung +Richtung	0
P02	Maßeinheit der Bewegung 1 Schritt 2 mm 3 Zoll 4 Grad	1
P03	Umrechungsfaktor Spindelsteigung 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
P07	Absenkrampe für Nothalt Eingabe in Hz/s	50 000
P08	f_{max} MØP, Fahrfrequenz beim Initialisieren Eingabe in Hz	4000
P09	Rampe MØP, für Initialisierung, zugehörig zu Parameter P08 Eingabe in Hz/s	25 000

Parameterliste		
P10	f_{min} MØP, Fahrfrequenz beim Verlassen der Endschalter Eingabe in Hz	400
P11	MØP Offset für Endschalter Plusrichtung Abstand des mechanischen Nullpunkts MØP (Referenzpunkt) vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt	0
P12	MØP Offset für Endschalter Minusrichtung Abstand des mechanischen Nullpunkts MØP vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt	0
P13	Beruhigungszeit MØP Wartezeit bei Initialisierung Eingabe in ms	20
P14	f_{max} Lauffrequenz bei Positionierbefehlen Eingabe in Hz (ganzzahlige Werte) siehe Kap. 6	4000
P15	Rampe für Lauffrequenz (P14) Eingabe in Hz/s (ganzzahlige Werte) OMC: 100 Hz/s bis 4 MHz/s TMC: 1000 Hz/s bis 4 MHz/s	25 000
P16	Beruhigungszeit Position Wartezeit nach Ausführung eines Fahrbefehls Eingabe in ms	20
P17	Boost (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 kann in Parameter P42 programmiert werden. 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
Die folgenden Parameter sind Zähler, die vom Programm beschrieben werden. Bei Aufruf der Parameterliste werden diese Parameter nicht angezeigt!		

Parameterliste		
P19	Elektronischer-Nullpunkt-Zähler Dient zur Bestimmung von Arbeitspunkten, kann bei Achsenstillstand gesetzt und ausgelesen werden.	
P20	Mechanischer-Nullpunkt-Zähler Zählt Impulse bezogen auf den mechanischen Nullpunkt. Kann bei Achsenstillstand ausgelesen werden. Am MØP wird P20 automatisch null gesetzt.	
P21	Absolutwertzähler Gibt die aktuelle Position an. P21 kann jeder Zeit abgefragt, beschrieben oder geändert werden. P21 wird am MØP <u>nicht</u> automatisch null gesetzt.	
P22	Encoderzähler Gibt die aktuelle Encoderposition an.	
P26 bis 45 wurden für Steuerungen OMC/TMC der Parameterliste hinzugefügt.		
P26	Teiler für SSI-Encoder Übertragungsgeschwindigkeit von 10 bis 80 (=100 bis 800 kHz)	0
P27	Initiatortyp 0 = + und – sind PNP-Öffner 1 = + ist Schließer, – ist Öffner 2 = + ist Öffner, – ist Schließer 3 = + und – sind PNP-Schließer	0
P28	Ausgang Motorbremse Ausgangsnummer für Motorbremse der Achse zuordnen. Eingabe für OMC: 1 - 8 für TMC: 1 - 16 Beispiel: Ausgangnummer = 4 Bei Fehler der Endstufe wird der Ausgang 4 gesetzt	0
P29	Bremsenfreischaltverzögerung Zeit vom Einschalten bis zum Lösen der Bremse Eingabe in s	0
P30	Einstellung Frequenzband 0 = manuell 1 = automatisch (siehe Kap. 6) <u>Anmerkung:</u> Es wird empfohlen, mit der automatischen Frequenzbandeinstellung zu arbeiten. Der Controller wählt zu jeder Lauffrequenz (P14) und Rampe (P15) einen geeigneten Wert.	1

Parameterliste		
P31	Frequenz- und Rampenvorteiler (nur wenn P30 = manuell) Vorteilerwerte: 3 oder 5 (OMC: 5, TMC: 3) Dieser Parameter kann für individuelle Einstellungen verwendet werden, wenn die automatische Frequenz-bandeinstellung für den speziellen Anwendungsfall nicht geeignet ist.	3
P32	Rampenform bei Positionierungen 0 = s-förmig 1 = linear <u>Anmerkung:</u> Die s-förmige Rampe kann mit P33 geändert werden.	1
P33	Bogenwertvorgabe für s-förmige Rampe Werte: OMC: 1 bis 8191 TMC: 1 bis 32767	1
	P33: niedriger Wert  P33: hoher Wert 	
P34	Encodertyp 0 = inkrementell 5,0 V 1 = inkrementell 5,5 V 2 = serielle Schnittstelle SSI Binär Code 5,0 V 3 = serielle Schnittstelle SSI Binär Code 5,5 V 4 = serielle Schnittstelle SSI Gray Code 5,0 V 5 = serielle Schnittstelle SSI Gray Code 5,5 V	0
P35	Auflösung bei Absolut-Encoder (SSI) Eingabe: maximale Auflösung in Bit (max. 32 Bit)	10
P36	Encoderfunktion 0 = Zähler 1 = dynamische Schrittfehlererkennung SFI	0
P37	Toleranz für Schrittfehlererkennung Eingabe: Toleranzwert für SFI-Auswertung	0
P38	Encoder Vorzugsdrehrichtung 0 = plus 1 = minus	0
P39	Encoder Umrechnungsfaktor 1 Inkrement entspricht ...	1

Parameterliste		
P40	Stoppstrom in Stufen von 0,1 A Bereich: 0 bis 6,3 A Eingabe: 0 bis 63	2
P41	Laufstrom in Stufen von 0,1 A Bereich: 0 bis 6,3 A Eingabe: 0 bis 63	4
P42	Booststrom in Stufen von 0,1 A Bereich: 0 bis 6,3 A Eingabe: 0 bis 63	4
P43	Stoppstromüberhöhungszeit in ms	20
P44	Taktumschaltung Endstufe 0 = Endstufe X (Y) auf Controller X (Y) 1 = Endstufe X (Y) auf Controller Y (X) 2 = Takt von außen auf Endstufe X (Y)	0
P45	Schrittauflösung 0 = Vollschritt 3 = 1/5 Schritt 1 = Halbschritt 4 = 1/10 Schritt 2 = 1/4 Schritt 5 = 1/20 Schritt 6 = 1/8 Schritt	4

6 Erklärungen zu den Parametern 30 und 31

Der Motorcontroller wird von verschiedenen chip-internen Werten wie Frequenz und Vorteiler parametrisiert. In der Einstellung P 30 = 1 werden diese zwei Werte automatisch aus den physikalischen Größen erzeugt, die durch die MINILOG-Parameter P 14 (Lauffrequenz) und P 15 (Rampe) vorgegeben sind. Auf Grund des binären Chipdesigns stehen nur bestimmte Frequenz-Kombinationen zur Verfügung:

Steuerung	**Vorteiler**	**f_{Max} [Hz]**	**$f_{Auflösung}$[Hz]**
OMC	1	16 383	1
	2	32 767	2
	3	65 535	4
	4	131 071	8
	5	262 143	16
TMC	1	65 535	1
	2	131 071	2
	3	262 143	4

Vorteiler/Frequenzen/Auflösung

Vorteiler: Definiert einen Frequenzbereich
f_{Max}: Maximalfrequenz in diesem Bereich
$f_{Auflösung}$: Mindestabstand zwischen zwei einstellbaren Frequenzen

Hinweis: Frequenz- und Zielposition können während des Laufes geändert werden!

Faustregel zur manuellen Einstellung des Frequenzbands:

Niedrige Frequenz – genaue Einstellung:	Geringen Vorteiler wählen Die Rampe ist relativ flach. Steile Rampen sind mit einem geringen Vorteiler nicht möglich!
Steile Beschleunigungsrampe: (nahezu Rechteckprofil)	Hohen Vorteiler wählen Das Frequenzraster ist in diesem Fall größer.

7 Programmbeispiel A/D Wandler

Programm	Kommentar
START	
R1SAD0C0	Lädt das Register 1 mit AD Karte 0 Ch 0
R2SAD0C1	Lädt das Register 2 mit AD Karte 0 Ch 1
R3SAD0C2	Lädt das Register 3 mit AD Karte 0 Ch 2
R4SAD0C3	Lädt das Register 4 mit AD Karte 0 Ch 3
R1W1	Schreiben des AD-Wertes in die Anzeige
R2W2	Schreiben des AD-Wertes in die Anzeige
R3W3	Schreiben des AD-Wertes in die Anzeige
R4W4	Schreiben des AD-Wertes in die Anzeige
N*START*	Rücksprung zum Start

8 Stichwortverzeichnis

Phytron GmbH • Industriestraße 12 • 82194 Gröbenzell, Germany
Tel. +49(0)8142/503-0 • Fax +49(0)8142/503-190 • E-Mail info@phytron.de • www.phytron.de

Phytron, Inc. • 600 Blair Park Road Suite 220 • Williston, VT 05495 USA
Tel. +1-802-872-1600 • Fax +1-802-872-0311 • Email info@phytron.com • www.phytron.com