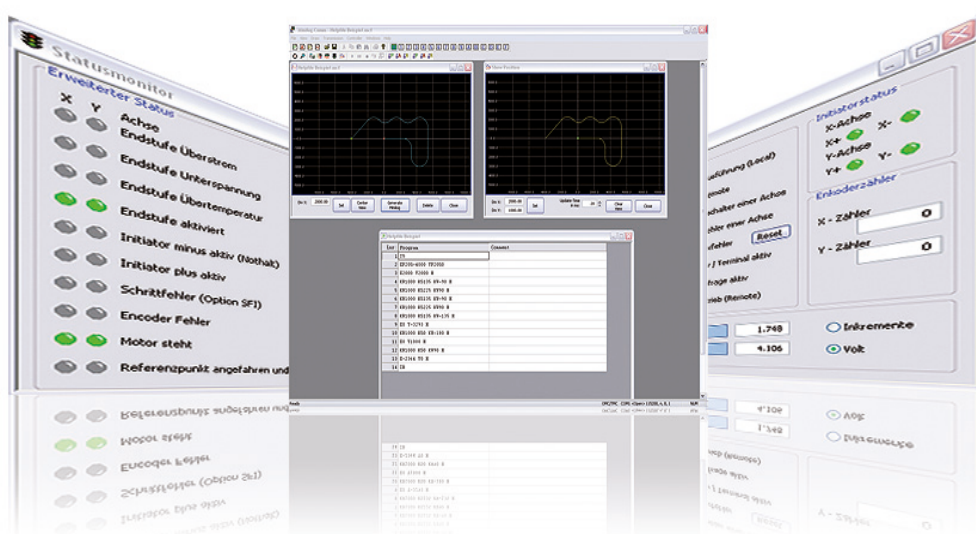


# Minilog MCC

## Programmiermanual für MCC



**Programmiermanual MINILOG  
für die Steuerungen  
MCC-1, MCC-2 und MCC-2 LIN**

**ORIGINAL BETRIEBSANLEITUNG**

## MINILOG

---

Version	Änderung
7	S.31 Bit 0 korrigiert Parameter P40 bis P42 (MCC-2 LIN) ergänzt gültig ab Seriennr. 15XXXXXXXX
8	S.49 Kap. 5 „P27 bis P49 sind spezielle Parameter für MCC-2“ entfernt

© 2018

Alle Rechte bei:

Phytron GmbH

Industriestraße 12

82194 Gröbenzell, Deutschland

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

Fax: +49(0)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 jederzeit ohne Ankündigung vorzunehmen.

Für Anregungen und Kritik sind wir dankbar.

(E-Mail-Adresse: [doku@phytron.de](mailto:doku@phytron.de))

Den neuesten Stand des Handbuchs finden Sie im Internet unter [www.phytron.de](http://www.phytron.de).

## Inhaltsverzeichnis

1	Struktur der Befehle .....	4
1.1	Aufbau des Befehlscodes .....	4
1.2	Aufbau von MiniLog-Programmen .....	5
1.3	Adressierungsarten.....	5
1.4	Bedingte Befehle .....	6
1.5	Daten– und Telegrammformat .....	7
2	MINILOG Befehle.....	9
2.1	Ausgänge .....	9
2.2	A/D-Wandler .....	10
2.3	Reset.....	10
2.4	Schreibausgabe über serielle Schnittstelle .....	10
2.5	Eingangsabfragen.....	11
2.6	Programmbeeinflussung bei Nothalt (NUR PROG).....	13
2.7	Programmunterbrechungen .....	13
2.8	Systemanpassung im Programmablauf .....	13
2.9	Sprungbefehle (NUR PROG).....	15
2.10	Programmzeilen wiederholen .....	16
2.11	Passwort.....	17
2.12	Programmaufruf beenden oder unterbrechen (NUR PROG).....	18
2.13	Programm- und Dateiverwaltung (NUR PC).....	18
2.14	Register .....	21
2.15	Registerbefehle .....	22
2.16	Systemstatus (NUR PC) .....	31
2.17	Daten ins Flash EPROM schreiben .....	33
2.18	Zeitschleifen .....	33
2.19	Unterprogramme (NUR PROG) .....	34
2.20	Terminalbefehle (auch von PC bei Terminalanschluss) .....	35
2.21	Achsenbefehle.....	36
2.22	Tastenabfrage Bedienterminal BT24 (auch von PC) .....	40
3	Minilog Befehle .....	41
4	DIN-Befehle .....	45
5	Parameter.....	48
5.1	Parameterliste .....	49
5.2	Übertragen des Parametersatzes in die Steuerung.....	54
6	Programmierbeispiele .....	55
6.1	Allgemein.....	55
6.2	Programmbeispiel A/D Wandler.....	55
7	Speicherung der Programme, Parameter und Register.....	56
8	Stromformung CS .....	57
9	Stichwortverzeichnis .....	58

## 1 Struktur der Befehle

### 1.1 Aufbau des Befehlscodes

---

**X**wert      **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 Befehlscode für die X-Achse abgedruckt, weil die Achse bei Einachsensteuerungen (MCC-1) immer „X“ genannt wird. Bei Mehrachsensteuerungen 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 -

**z**wert      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 +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 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. 2.12, muss der erste alphanumerische Programmname durch eine Leerstelle vom zweiten Programmnamen bzw. dem nachfolgenden alphanumerischen Teil des Befehlscodes getrennt werden.

## 1.2 Aufbau von MiniLog-Programmen

MiniLog-Programme bestehen aus bis zu 2000 Programmzeilen, die durchnummeriert werden. Die Zeilennummern vergibt MiniLog-Comm automatisch.

Die einzelnen Befehle in der Programmzeile durch Leerzeichen voneinander trennen.

Zwischen die zu einem Befehl gehörenden Zeichen keine zusätzlichen Leerzeichen einfügen.

Die Befehle werden seriell abgearbeitet.

Mit Hilfe der Zeilennummern können Sprungbefehle und Unterprogramme definiert werden.

Parameter und Register sollte man am Programmanfang festlegen.

Zeilen-, Parameter- und Registernummern können mit oder ohne führende Nullen eingegeben werden.

Beispiel: R0001 oder R1

Ein Zeilenumbruch im Programm erfolgt durch ein **CR (0x0D)**

Beispiel:       A1S T500 A1R **0x0D**  
                  A2S T500 A2R **0x0D**

## 1.3 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:**

<u><b>Befehl</b></u>	<u><b>Bedeutung</b></u>
<b>RnnBE<sub>nn</sub>–mm</b>	Das Register nn wird mit dem Status der Eingänge nn bis mm binär beschrieben.
<b>Beispiel:       R1BE1–8</b>	
	Die Eingänge 1 bis 8 haben z. B. den Zustand: <b>1010 0101</b> .
	Das Register 1 wird nun mit dem Binärwert der Eingänge beschrieben. Nach dem Befehl hat das Register 1 den Wert 165.

Beispiel **Indirekte Adressierung**:

<u><b>Befehl</b></u>	<u><b>Bedeutung</b></u>
<b>R[Rnn]BEnn–mm</b>	Das Register, das durch das Register nn adressiert wird, wird mit dem Status der Eingänge nn bis mm binär beschrieben
<b>Beispiel: R1S10 R[R1]BE1–8</b>	
	Die Eingänge 1 bis 8 haben z. B. den Zustand: <b>1010 0101</b> .
	Das Register 1 wird mit dem Befehl <b>R1S10</b> 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 15) und Unterprogrammaufrufen (ab Seite 34) kann die Ziel- bzw. Startzeile im Befehlscode mit einem Label (\*la\*), das dieser Programmzeile zugeordnet wurde, angegeben 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.

## **1.4 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. 2.5) oder durch einen Register Vergleich (siehe Kap. 2.14) 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.

## 1.5 Daten- und Telegrammformat

**Datenformat:** No Parity  
1 Stopbit  
8 Bit ASCII-Code  
57600 Baud

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

**Ohne Prüfsumme:** <STX> | Adresse | Befehl | <ETX>

**Mit Prüfsumme:** <STX> | Adresse | Befehl | Separator | Prüfsumme | <ETX>

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

<STX> | ACK | Antwort | <ETX> oder  
<STX> | ACK | <ETX> oder  
<STX> | NAK | <ETX>

	Bedeutung
<STX>	<STX> (Start of Text, 02 <sub>H</sub> ) als Kennzeichen für den Start eines neuen Telegramms.
Adresse	Adresse der Steuerung mit den Werten „0“ bis „9“ und „A“ bis „F“ (30 <sub>H</sub> ..39 <sub>H</sub> bzw. 41 <sub>H</sub> ..46 <sub>H</sub> ). Außerdem die Broadcast <sup>1</sup> Adresse @ (40 <sub>H</sub> ).
Befehl	MINILOG Befehlscode
Separator	;, 3A <sub>H</sub> 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 <sub>H</sub> ) als Telegrammende-Kennung.
ACK	(Acknowledge 06 <sub>H</sub> ), der Befehl wurde quittiert
NAK	(Negative Acknowledge 15 <sub>H</sub> ), der Befehl wurde negativ quittiert
Antwort	Antwort als Zahl oder String, z.B. E oder N

<sup>1</sup> 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.



## MINILOG

---

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

$$CS = \text{Adresse} \oplus \text{Datenbyte1} \oplus \text{Datenbyte2} \dots \oplus \text{DatenbyteN} \oplus \text{Separator}$$

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

Die MCC 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 MCC 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>

## 2 MINILOG Befehle

### 2.1 Ausgänge

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Ausgänge schalten</b>
<b>Annnz</b>	Man kann einen Ausgang oder mehrere Ausgänge gleichzeitig schalten. nnn, mmm, xxx → Nummer des Ausgangs
<b>Annnzmmmzxxxz</b>	z = S → setzen z = R → rücksetzen <b>Beispiel: A1S2R3S</b> Ausgang 1 und 3 einschalten, Ausgang 2 ausschalten
	<b>Ausgangsgruppe lesen</b>
<b>AGnR</b>	Die Ausgangsgruppe n lesen. (NUR PC) <b>Beispiel: AG1R</b> Die 1. Ausgangsgruppe wird gelesen <b>Antwort : &lt;STX&gt;&lt;ACK&gt;nnnnnnnn&lt;ETX&gt;</b> n = 0 Ausgang nicht gesetzt n = 1 Ausgang gesetzt
	<b>Ausgangsgruppe Ausgänge setzen</b>
<b>AGnSzzzzzzzz</b>	Ausgangsgruppe setzen n=1, z= 0 oder 1. z muss immer 8 Stellen haben <b>Beispiel: AG1S10101001</b> Die 1. Ausgangsgruppe wird mit der Information '10101001' gesetzt
	<b>Ausgangszustand lesen</b>
<b>ARnnn;mmm;xxx</b>	Der Zustand der Ausgänge nnn, mmm, xxx wird gelesen. (NUR PC) <b>Antwort : &lt;STX&gt;&lt;ACK&gt;nnn&lt;ETX&gt;</b> n = 0 Ausgang nicht gesetzt n = 1 Ausgang gesetzt <b>Wichtig:</b> Zwischen den Ausgangsnummern ein ; setzen.

## 2.2 A/D-Wandler

---

<u>Befehl</u>	<u>Bedeutung</u>
ADnR	AD-Wandler-Einstellung lesen n → Adresse des A/D-Wandlers: n=1 oder n=2  <b>Antwort : &lt;STX&gt;&lt;ACK&gt;[0 bis 1023]&lt;ETX&gt; (NUR PC)</b> 0 bis 1023 = 0 bis 5 V

## 2.3 Reset

---

<u>Befehl</u>	<u>Bedeutung</u>
CR	Die Steuerung wird über die Schnittstelle zurück gesetzt.
CT	Die Terminalanzeige wird über die Schnittstelle gelöscht. <b>Antwort : &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt; (NUR PC)</b>

## 2.4 Schreibausgabe über serielle Schnittstelle

---

<u>Befehl</u>	<u>Bedeutung</u>
	Es können über die serielle Schnittstelle (X5 Com) Informationen ausgegeben werden. Die Schreibausgabe erfolgt ohne Formatierung. s = 1 → Schnittstelle RS232 (Com X5)
Ds <Text>	Den Text, der in Klammern steht, ausgeben.
DsRnn	Den Inhalt des Registers nn ausgeben.
DsR[Rnn]	Den Inhalt des Registers, das durch das Register nn adressiert wird, ausgeben.
DsxPmm	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  <b>Beispiel: D14P10</b> Hier wird der Parameter 10 der Achse 4 über die Schnittstelle Com X5 ausgegeben.

## 2.5 Eingangsabfragen

### Befehl

### Bedeutung

#### UND-Verknüpfung

**E<sup>nnzmmzxx</sup>** 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<sup>1S2R3S</sup>**

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

**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: **Ev<sup>1S2R3S</sup>**

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> oder  
<STX><ACK> N <ETX> (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)

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>Ennzmzmz</b>	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.

### **Definition der Ein- / Ausgänge nur bei MCC-1**

<b>EASnnnnnnnn</b>	Die Steuerung MCC-1 verfügt über acht galvanisch getrennte, bidirektionale, digitale I/Os. Per Minilog-Programmierung kann vom Benutzer definiert werden, welche I/Os als Eingänge oder Ausgänge gesetzt werden sollen.
--------------------	---

n = Zuordnung → Eingang oder Ausgang

n = 1 → Eingang

n = 0 → Ausgang

**Beispiel: EAS00000011**

Die I/Os 1 bis 6 sind Ausgänge, 7 und 8 werden als Eingänge geschaltet.

### **Eingangsgruppe lesen**

<b>EGnR</b>	Die Eingangsgruppe n lesen. (NUR PC) n=1 bis 8
-------------	---

**Antwort : <STX><ACK>nnnnnnnn<ETX>**

n = 0 Eingang nicht gesetzt

n = 1 Eingang gesetzt

### **Eingangszustand lesen**

<b>ERnn;mm;xx</b>	Der Zustand der Eingänge nn, mm, xx wird gelesen. (nur PC)
-------------------	--

**Antwort : <STX><ACK>nnn<ETX>**

n = 0 Eingang nicht gesetzt

n = 1 Eingang gesetzt

**Wichtig:** Zwischen den Eingangsnummern ein ; setzen

## 2.6 Programmbeeinflussung bei Nothalt (NUR PROG)

<u>Befehl</u>	<u>Bedeutung</u>
<b>FNznr</b>	Es wird die Zeile, an der das Programm beim Nothalt fortfahren soll, festgelegt.
<b>FN*la**</b>	Es wird die Zeile, an der das Programm beim Nothalt fortfahren soll, durch ein Label bestimmt.
<b>FP[name]</b>	Programmangabe für den Nothalt. Beim Nothalt wird in das Programm mit dem Namen name gesprungen.

## 2.7 Programmunterbrechungen

<u>Befehl</u>	<u>Bedeutung</u>
<b>H</b>	Das Programm wartet hier, bis alle Achsen stehen. (NUR PROG)

## 2.8 Systemanpassung im Programmablauf

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Achsenanzahl</b>
<b>IAR</b>	Die Anzahl der vorhandenen Achsen auslesen. (NUR PC) <b>Antwort: &lt;STX&gt;&lt;ACK&gt;n&lt;ETX&gt;</b>
	<b>Automatikstart</b>
<b>IBSname</b>	Der Programmname wird in das Autostartregister geschrieben. Mit diesem Programm wird gestartet, wenn der REMOTE/LOCAL-Schalter auf LOCAL steht. <b>Antwort: &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt; oder &lt;STX&gt;&lt;NAK&gt;&lt;ETX&gt; (NUR PC)</b>
<b>IBR</b>	Der Programmname für den Autostart wird ausgelesen. (NUR PC) <b>Antwort: &lt;STX&gt;&lt;ACK&gt;name&lt;ETX&gt;</b>
	<b>Baudrate einstellen/lesen (NUR PC)</b>
<b>ICnSbaud</b>	Die Baudrate für die MCC Schnittstelle setzen. n = 1 → COM 1 von MCC baud = Baudrate (9600, 19200, 38400, 57600 oder 115200 Baud)
<b>ICnR</b>	Baudraten-Einstellung der MCC Schnittstelle wird ausgelesen. n = 1 → COM 1 von MCC

## Befehl

## Bedeutung

### **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>**

**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>**

### **Inhaltsverzeichnis RAM**

**IPn**

n-ten Programmnamen der Programmliste vom RAM auslesen. Wenn kein Programmname vorhanden ist, kommt als Antwort NAK (NUR PC).

**Antwort: <STX><ACK>name<ETX>**

**Antwort: <STX><NAK><ETX>** wenn kein Name vorhanden

### **Information des Übertragungsprotokolls**

**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>**

## 2.9 Sprungbefehle (NUR PROG)

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



## MINILOG

---

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

## 2.10 Programmzeilen wiederholen

---

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>NWnn</b>	Die Programmzeile nn mal wiederholen.
<b>NWRnn</b> <b>NWR[Rnn]</b>	Die Programmzeile so oft wiederholen, wie es im Register nn angegeben ist.
<b>Antwort: &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt;</b>	

## 2.11 Passwort

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>PA_</b>	Die Steuerung wird freigeschaltet, wenn kein Passwort vergeben ist. Dann ist eine Sperrung der Steuerung auch nicht möglich.
<b>PAname</b>	Schaltet die passwortgeschützte Steuerung frei.
<b>PSname</b>	Dieser Befehl vergibt ein Passwort für die Steuerung. Es besteht aus maximal 8 Stellen.

### **Freigabezustand für Programme, Parameter und Register**

<b>PWSp</b>	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 freigebeben 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: <b>PWS5</b> Programm und Register gesperrt (1+4=5)
-------------	---

<b>PWR</b>	Freigabezustand lesen  Die Antwort ist eine zweistellige Zahlenkombination. <b>&lt;ACK&gt; sp</b> s □ Freigabezustand der Steuerung s = 0 Steuerung gesperrt s = 1 Steuerung frei p → Freigabezustand für Programm, Parameter und Register siehe oben  Beispiel: <b>PWR</b> <b>&lt;ACK&gt; 15</b> s = 1 → Steuerung frei p = 5 → Programm und Register gesperrt (1+4=5)
------------	--

### 2.12 Programmaufruf beenden oder unterbrechen (NUR PROG)

---

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

### 2.13 Programm- und Dateiverwaltung (NUR PC)

---

**Wichtig:**

Während Programme an die Steuerung übertragen oder von der Steuerung gelesen werden, vermeiden Sie es unbedingt, in dieser Zeit die Steuerung abzuschalten oder das Kabel abzuziehen. Es kann zu Programmverlust kommen.

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Programme und Dateien löschen</b>
<b>QDP*.*</b>	Es werden alle Programme im RAM-Speicher gelöscht.
<b>QDR</b>	Es werden alle Register im RAM auf null gesetzt.
	<b>Programmzeile lesen</b>
<b>QPname NnnR</b>	Es wird die Zeile nn des Programms name ausgelesen.
	<b>Programmstart ab Zeile</b>
<b>QPname NnnA</b>	Das Programm name wird ab der Zeile nn gestartet.
	<b>Programm Abbruch</b>
<b>QPE</b>	Wird der QPE Befehl vom Rechner gesendet, dann wird in die Programmebene zurückgesprungen, aus der das Programm gestartet wurde.

## Programmübertragung mit Abfrage

**QPname Sbyte**

Das Programm mit dem Namen name kann nur blockweise übertragen werden. Bei der Übertragung des Programms muss die gesamte Steuersequenz eingehalten werden. Der Name muss 8 Zeichen lang sein.

name → maximal 8 Zeichen

byte → Anzahl der zu übertragenden Bytes

1. vom Rechner:

**<STX>Steuerungsadresse QPname Sbyte<ETX>**

Die Programmübertragungssequenz wird gestartet.

2. Antwort der Steuerung:

**<STX><ACK>O<ETX>**

wenn das Programm nicht auf der Steuerung vorhanden ist und das Programm in den RAM-Speicher der Steuerung passt.

**<STX><ACK>E<ETX>**

wenn das Programm auf der Steuerung vorhanden ist und überschrieben werden muss:

Wie wird überschrieben:

- 1) Alle Programme der MCC auf dem PC sichern.
- 2) Alle Programme auf dem Flash-RAM der MCC löschen.
- 3) Programme (incl. geändertes Progr.) in MCC zurückschreiben.

3. Programmübertragung:

Beginn: **<STX>Steuerungsadresse Block 1<ETX>**

Block 1 = 256 Byte lang und beginnt mit <ETB>Programmname, wobei Programmname 8 Zeichen lang sein muss!

- Weitere Blöcke (Block 1+x) müssen immer 256 Byte lang sein und sind in <STX><ETX> eingeschlossen.

**<STX>Steuerungsadresse Block 1+x<ETX>**

- Letzter Block:

Es muss als letztes Zeichen ein 0x04 (EOT) stehen. Ist der letzte Block kleiner als die angegebenen 256 Byte, muss der Rest des Blocks mit EOT aufgefüllt werden.

Beispiel:

**<STX>Steuerungsadresse Block Ende<EOT><EOT>.....**

**<EOT><EOT><ETX>**

Antwort der Steuerung nach jedem Block:

**<STX><ACK><ETX>**

### Programm lesen mit Abfrage

**QPname R**

Das Programm mit dem Namen name soll aus der Steuerung ausgelesen werden. Der Programmname muss 8 Zeichen lang sein. Das Programm wird zeilenweise gelesen.

#### Abfrage und senden

1. vom Rechner:

**<STX>Steuerungsadresse QPname R<ETX>**

Das Programm mit dem Namen name soll gelesen werden.

2. von der Steuerung:

**<STX><ACK>Oznr<ETX>**

Die Steuerung meldet ein O und die Anzahl der Programmzeilen znr, wenn das Programm vorhanden ist.

3. vom Rechner:

**<STX>Steuerungsadresse J<ETX>**

Der Rechner empfängt die erste Zeile von der Steuerung.

4. von der Steuerung:

**<STX>Daten Programmzeile x<ETX>**

Die Daten werden zeilenweise mit Angabe der Zeilennummer eingelesen.

Punkt 3. und 4. wird so lange wiederholt bis alle Zeilen empfangen wurden.

An die letzte Zeile sind 0x04 (EOT) angehängt. Damit ist die Übertragung abgeschlossen.

Beispiel letzte Zeile:

**<STX>Daten letzte Programmzeile.....<EOT><ETX>**

## 2.14 Register

Die Steuerungen MCC-2 enthalten 1000 Speicherplätze zur Eingabe von Variablen, die in MiniLog-Programmen Register genannt werden.

Die Register werden mit R1 bis R1000 bezeichnet.

In jedes Register können Zahlen mit max. zehn Stellen geschrieben werden. Auch Dezimalzahlen sind möglich. Dabei dürfen vor oder nach dem Punkt (Bedeutung: Komma) bis zu sieben Stellen stehen. Die gesamte Zahl darf nicht mehr als acht Stellen haben.

Die Register sollten möglichst in den ersten Programmzeilen mit den gewünschten Werten geladen werden.

Register beschreiben: **RnnnnSzz**

Register auslesen: **RnnnnR**

Erklärungen:

<b>R</b>	Kennbuchstabe Register
nnnn	Registernummer
<b>S</b>	Schreiben
zz	max. zehnstellige Zahl

Im Programm können Register zur indirekten Eingabe von Positionen verwendet werden. In Verbindung mit Rechenoperationen sind die Register für Zählerfunktionen während des Programmablaufs einsetzbar.

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:	Vergleich Registerwert mit Zahl und bedingter Sprung
R999=1 NE11 N77	Wenn Register 999 den Wert 1 enthält, wird zu Zeile 11 gesprungen, wenn nicht, zu Zeile 77.

## 2.15 Registerbefehle

---

### Befehl

### Bedeutung

#### Registerinhalt ganzzahlig

Rnn.z

Die Nachkommastellen des Registerinhalts nn löschen ohne Rundung des Wertes.

z = 0 – 6    Nachkommastellen

#### Registerinhalt binär ausgeben

RnnBA<sub>nn–mm</sub>  
R[Rnn]BA<sub>nn–mm</sub>

Der Inhalt des Registers nn wird binär an die Ausgänge nn bis mm ausgegeben.

#### Register binär beschreiben (über Eingänge)

RnnBE<sub>nn–mm</sub>  
R[Rnn]BE<sub>nn–mm</sub>

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

RnnBS<sub>wert</sub>  
R[Rnn]BS<sub>wert</sub>

Den Inhalt des Registers nn binär mit dem Wert wert 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

RnnBL<sub>m</sub>  
R[Rnn]BL<sub>m</sub>

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> (NUR PC)

**Befehl**

**RnnBRm**  
**R[Rnn]BRm**

**Bedeutung**

Der Inhalt des Registers nn wird binär um m Stellen nach rechts (→ LSB) geschoben. Es wird links mit 0 aufgefüllt.

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**

**RnnBTm**  
**R[Rnn]BTm**

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> oder**

**<STX><ACK> N <ETX> (NUR PC)**

**Register logische Verknüpfung**

**Und Verknüpfung**

**RnnB^zwert**  
**R[Rnn]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



## Befehl

**RnnB^Rmm**  
**R[Rnn]B^Rmm**  
**RnnB^R[Rmm]**  
**R[Rnn]B^R[Rmm]**

## Bedeutung

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.

## Oder Verknüpfung

**RnnBvzwert**  
**R[Rnn]Bvzwert**

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
<b>Ergebnis</b>	<b>936</b>	<b>3A8</b>	<b>1110101000</b>

**RnnBvRmm**  
**R[Rnn]BvRmm**  
**RnnBvR[Rmm]**  
**R[Rnn]BvR[Rmm]**

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> (NUR PC)**

## XOR Verknüpfung

**RnnBXzwert**  
**R[Rnn]BXzwert**

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
<b>Ergebnis</b>	<b>776</b>	<b>308</b>	<b>1100001000</b>

**RnnBXRmm**  
**R[Rnn]BXRmm**  
**RnnBXR[Rmm]**  
**R[Rnn]BXR[Rmm]**

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.

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>Registerinhalte vergleichen (mit einem Zahlenwert)</b>	
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.
Rnn#zwert R[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 R[Rnn]>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[Rnn]<zwert	Vergleich des Inhalts von Register nn mit dem Zahlenwert zwert. Das Bedingungsbyte wird gesetzt, wenn das Ergebnis kleiner ist, sonst wird es rückgesetzt.
<b>Registerinhalte miteinander vergleichen</b>	
Rnn=Rmm R[Rnn]=Rmm Rnn=R[Rmm] R[Rnn]=R[Rmm]	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#Rmm R[Rnn]#Rmm Rnn#R[Rmm] R[Rnn]#R[Rmm]	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>Rmm R[Rnn]>Rmm Rnn>R[Rmm] R[Rnn]>R[Rmm]	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.
Rnn<Rmm R[Rnn]<Rmm Rnn<R[Rmm] R[Rnn]<R[Rmm]	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.
<b>Antwort bei allen Vergleichen:</b>	
<STX><ACK> E <ETX> oder	
<STX><ACK> N <ETX> (NUR PC)	
<b>Rechenoperationen mit Registern</b>	
<b>Addieren</b>	
Rnn+zwert R[Rnn]+zwert	Zum Inhalt des Registers nn wird der Wert zwert addiert.

## MINILOG

---

### Befehl

### Bedeutung

$R_{nn}+R_{mm}$   
 $R_{nn}+R[R_{mm}]$   
 $R[R_{nn}]+R_{mm}$   
 $R[R_{nn}]+R[R_{mm}]$

Zum Inhalt des Registers nn wird der Inhalt des Registers mm addiert.

### **Subtrahieren**

$R_{nn}-\text{zwert}$   
 $R[R_{nn}]-\text{zwert}$

Vom Inhalt des Registers nn wird der Wert zwert subtrahiert.

$R_{nn}-R_{mm}$   
 $R_{nn}-R[R_{mm}]$   
 $R[R_{nn}]-R_{mm}$   
 $R[R_{nn}]-R[R_{mm}]$

Vom Inhalt des Registers nn wird der Inhalt des Registers mm subtrahiert.

### **Multiplizieren**

$R_{nn}*\text{zwert}$   
 $R[R_{nn}]*\text{zwert}$

Der Inhalt des Registers nn wird mit dem Wert zwert multipliziert.

$R_{nn}*R_{mm}$   
 $R[R_{nn}]*R_{mm}$   
 $R_{nn}*R[R_{mm}]$   
 $R[R_{nn}]*R[R_{mm}]$

Der Inhalt des Registers nn wird mit dem Inhalt des Registers mm multipliziert.

### **Dividieren**

$R_{nn}:\text{zwert}$   
 $R[R_{nn}]:\text{zwert}$   
 $R_{nn}/\text{zwert}$   
 $R[R_{nn}]/\text{zwert}$

Der Inhalt des Registers nn wird durch den Wert zwert dividiert.

$R_{nn}:R_{mm}$   
 $R_{nn}:R[R_{mm}]$   
 $R[R_{nn}]:R_{mm}$   
 $R[R_{nn}]:R[R_{mm}]$

Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.

$R_{nn}/R_{mm}$   
 $R_{nn}/R[R_{mm}]$   
 $R[R_{nn}]/R_{mm}$   
 $R[R_{nn}]/R[R_{mm}]$

Der Inhalt des Registers nn wird durch den Inhalt des Registers mm dividiert.

### **Winkelfunktionen**

$R_{nn}\text{SIN}$   
 $R_{nn}\text{COS}$   
 $R_{nn}\text{TAN}$

Vom Wert des Registers nn wird Sinus, Cosinus oder Tangens berechnet und das Ergebnis ins Register nn zurückgeschrieben.

### **Quadratwurzel**

$R_{nn}\text{QW}$

Aus dem Wert des Registers nn wird die Quadratwurzel berechnet und in das Register nn zurückgeschrieben.

<u><b>Befehl</b></u>	<u><b>Bedeutung</b></u>
	<b>Zufallszahl</b>
<b>RnnRAND</b>	Das Register nn wird mit Zufallszahl im Bereich 0 bis 4294967296 ( $2^{32}$ ) beschrieben.
	<b>Register auslesen</b>
<b>RnnR</b> <b>R[Rnn]R</b>	Der Inhalt des Registers nn wird ausgelesen. (NUR PROG) <b>Antwort: &lt;STX&gt;&lt;ACK&gt;wert&lt;ETX&gt;</b> <b>Antwort bei allen Rechenoperationen:</b> <b>&lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt;</b> (NUR PC)
	<b>Register beschreiben:</b>
	<b>mit Dezimalwerten</b>
<b>RnnSzwert</b> <b>R[Rnn]Szwert</b>	Das Register nn mit dem Zahlenwert zwert beschreiben.
	<b>mit Registerinhalten</b>
<b>RnnSRmm</b> <b>R[Rnn]SRmm</b> <b>RnnSR[Rmm]</b> <b>R[Rnn]SR[mm]</b>	Das Register nn mit dem Inhalt des Registers mm beschreiben.
	<b>mit Parameterwerten</b>
<b>RnnSXPmm</b> <b>R[Rnn]SXPmm</b>	Das Register nn mit dem Parameter mm der Achse X beschreiben.
	<b>mit Zeilennummer Nothalt</b>
<b>RnnSN</b> <b>R[Rnn]SN</b>	Das Register nn mit der Zeilennummer, in der ein Nothalt ausgelöst wurde, beschreiben. <b>Beispiel :</b> <b>ZNR 001 FN10</b> <b>ZNR 002 X+1000</b> <b>ZNR 003 X-1000 H N2</b> <b>ZNR 010 R1SN</b>  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 <b>R1SN</b> wird nun das Register 1 mit der Zeilennummer beschrieben, in der der Nothalt ausgelöst wurde. Nun kann ausgewertet werden, bei welcher Positionierung ein Nothalt aufgetreten ist.
	<b>mit dem Timertickerwert</b>
<b>RnnSTT</b>	Das Register nn mit dem Timertickerwert beschreiben.

## Befehl

## Bedeutung

### mit der Zeilennummer

**RnnSZ**

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

**RnnSEmm-xx.k**

**R[Rnn]SEmm-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

**RnnST**

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>wenn Terminal**  
                  **vorhanden**  
                  **<STX><NAK><ETX> wenn kein Terminal**  
                  **vorhanden       (NUR PC)**

**RnnST.z**

Das Register nn in Zeile 4 anzeigen mit z Nachkommastellen (z=0 bis 6). Eingabe neuer Daten an der Cursorposition. Mit Drücken der 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.

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>RnnSTy</b>	<p>Das Register nn in Zeile y anzeigen (y=1 bis 4) und nach Neueingabe mit ENTER-Taste neu beschreiben.</p> <p><b>Beispiel: R2ST3</b></p> <p>Das Register 2 wird in der 3.Zeile am Terminal ab Position 1 angezeigt und neu beschrieben.</p>
<b>RnnSTy.z</b>	<p>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.</p> <p><b>Beispiel: R2ST3.4</b></p> <p>Das Register 2 wird in der 3.Zeile mit 4 Nachkommastellen ab Position 1 am Terminal angezeigt und neu beschrieben.</p>
<b>RnnSTy;m</b>	<p>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.</p> <p><b>Beispiel: R1ST3;7</b></p> <p>Das Register 1 wird in der 3.Zeile ab Position 7 angezeigt und neu beschrieben.</p>
<b>RnnSTy;m.z</b>	<p>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.</p> <p><b>Beispiel: R2ST2;2.6</b></p> <p>Das Register 2 wird in der 2.Zeile ab Position 2 mit 6 Nachkommastellen angezeigt und neu beschrieben.</p>
<b>mit A/D-Wandlerwerten</b>	
<b>RnnSADy</b> <b>R[Rnn]SADy</b>	<p>Das Register nn mit dem Wert des A/D-Wandlers beschreiben.</p> <p>y=1 bis 2: Kanal des A/D-Wandlers.</p>
<b>Register mit Terminal anzeigen</b>	
<b>RnnWy</b>	<p>Den Registerwert nn in Zeile y ab Position 1 anzeigen (y=1 bis 4).</p> <p><b>Beispiel: R2W2</b></p> <p>Das Register 2 wird in der 2.Zeile ab der 1.Position angezeigt.</p>
<b>RnnWy.z</b>	<p>Den Registerwert nn in Zeile y ab Position 1 mit Nachkommastellen z (y=1 bis 4, z= 0 bis 6) anzeigen.</p> <p><b>Beispiel: R1W4.6</b></p> <p>Das Register 1 wird in der 4.Zeile ab der 1.Position mit 6 Nachkommastellen angezeigt.</p>
<b>RnnWy;m</b>	<p>Den Registerwert nn in Zeile y ab Position m (y=1 bis 4, m= 1 bis 20) anzeigen.</p> <p><b>Beispiel : R2W3;5</b></p> <p>Das Register 2 wird in der 3.Zeile ab der 5.Position angezeigt.</p>

## MINILOG

---

### Befehl

RnnW<sub>y;m.z</sub>

### Bedeutung

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> (NUR PC)

## 2.16 Systemstatus (NUR PC)

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Systemstatus Allgemein</b>
<b>S</b>	<p>Achsentest mit Ausgabe der Anzahl der Achsen.</p> <p><b>Antwort:</b> &lt;STX&gt;&lt;ACK&gt;n IO &lt;ETX&gt;</p> <p>n = Anzahl der Achsen</p>
	<b>Systemstatus binär</b>
<b>SB</b>	<p>Den Status der Steuerung auslesen. Der Status wird im achtstelligen Binärformat (<math>d_B = 0</math> oder <math>1</math>) ausgegeben.</p> <p><b>Antwort:</b> &lt;STX&gt;&lt;ACK&gt;<math>d_{B8}.....d_{B1}</math>&lt;ETX&gt;</p> <p><math>d_B 1 = 1 \rightarrow</math> Programmausführung</p> <p><math>d_B 2 = 1 \rightarrow</math> Software Remote</p> <p><math>d_B 3 = 1 \rightarrow</math> Nothaltendschalter einer Achse</p> <p><math>d_B 4 = 1 \rightarrow</math> Endstufenfehler einer Achse</p> <p><math>d_B 5 = 1 \rightarrow</math> Programmierfehler (wird nach Status auslesen rückgesetzt)</p> <p><math>d_B 6 = 1 \rightarrow</math> Terminal aktiv</p> <p><math>d_B 7 = 1 \rightarrow</math> Eingangsabfrage aktiv</p> <p><math>d_B 8 = 1 \rightarrow</math> Rechneraufruf</p>
	<b>Systemstatus erweitert</b>
<b>SE</b>	<p>Den Status der Steuerung auslesen. Der Status wird im Hexadecimal Code ausgegeben. Pro Achse stehen zwei Bytes (4 Hexadezimal Digits <math>d_H</math>) zur Verfügung: 1. + 2. Byte für die X-Achse, 3. + 4. Byte für die Y-Achse.</p> <p><b>Antwort:</b> &lt;STX&gt;&lt;ACK&gt;<math>d_{HX}d_{HX}d_{HX}d_{HX}d_{HY}d_{HY}d_{HY}d_{HY}</math>&lt;ETX&gt;</p> <p>Bit 0 = 1 <math>\rightarrow</math> Endstufenfehler</p> <p>Bit 1 = 1 <math>\rightarrow</math> Endstufe Unterspannung</p> <p>Bit 2 = 1 <math>\rightarrow</math> Endstufe Übertemperatur</p> <p>Bit 3 = 1 <math>\rightarrow</math> Endstufe aktiviert</p> <p>Bit 4 = 1 <math>\rightarrow</math> Initiator minus aktiv (Nothalt)</p> <p>Bit 5 = 1 <math>\rightarrow</math> Initiator plus aktiv</p> <p>Bit 6 = 1 <math>\rightarrow</math> Schrittfehler (nur mit Option SFI = Step Failure Indication)</p> <p>Bit 7 = 1 <math>\rightarrow</math> Encoder Fehler</p> <p>Bit 8 = 1 <math>\rightarrow</math> Motor steht</p> <p>Bit 9 = 1 <math>\rightarrow</math> Referenzpunkt angefahren und OK (wird bei Stop über Initiator rückgesetzt)</p> <p>(Bit 10 bis Bit 15 nicht belegt)</p> <p>Wenn Bit 0 bis Bit 2 gleichzeitig gesetzt sind, ist keine Endstufe angeschlossen. Andernfalls kann immer nur ein Fehler anstehen.</p>



## Befehl

## Bedeutung

**SH**

### **Systemstatus Achsen**

Achsentest mit Ausgabe, ob Achsen stehen.

**Antwort:**<STX><ACK> E <ETX>, wenn alle Achsen stehen.

          <STX><ACK> N <ETX>, wenn alle Achsen laufen.

### **Systemstatus dezimal**

**ST**

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

**Antwort:** <STX><ACK>wert <ETX>

wert = Zahlenwert zwischen 0 und 255

0     = Programmende im LOCAL-Betrieb.

1     = Programmausführung

2     = Software Remote

4     = Nothaltenschalter einer Achse

8     = Endstufenfehler einer Achse

16    = Programmierfehler (wird nach Status auslesen rückgesetzt)

32    = Terminal aktiv oder Enable aktiv

64    = Eingangsabfrage aktiv

128   = Rechnerbetrieb

### **Initiatoren**

**SUI**

Den Status der Initiatoren abfragen.

**Antwort:**<STX><ACK>I=n <ETX>

n = 0 → Achse ist frei, kein Initiator hat angesprochen

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)

### **Synchronstart**

**S1**

Synchronstart der Achsen vorbereiten

**S0**

Synchronstart der Achsen ausführen

## 2.17 Daten ins Flash EPROM schreiben

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Programme und Achsenparameter speichern (NUR PC)</b>
SA	Achsenparameter auf das Flash EPROM schreiben.

## 2.18 Zeitschleifen

<u>Befehl</u>	<u>Bedeutung</u>
Tzwert TRnn TR[Rnn]	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.  <b>Antwort: &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt; (NUR PC)</b>
TTSzwert TTSRnn TTSR[Rnn]	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.  <b>Antwort: &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt; (NUR PC)</b>
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>Rnn TT>R[Rnn] TT<zwert TT<Rnn TT<R[Rnn]	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.  <b>Antwort: &lt;STX&gt;&lt;ACK&gt; E &lt;ETX&gt; oder &lt;STX&gt;&lt;ACK&gt; N &lt;ETX&gt; (NUR PC)</b>

## 2.19 Unterprogramme (NUR PROG)

---

<u>Befehl</u>	<u>Bedeutung</u>
	<b>Unterprogramm abbrechen</b>
<b>UA</b>	Alle Unterprogramme werden abgebrochen und der Stack wird neu gesetzt. Das Programm kann mit einem Sprungbefehl fortgesetzt werden.
	<b>Unterprogramm beenden</b>
<b>UE</b>	Das Unterprogramm wird beendet und es wird an der Stelle fortgefahren, an der das Unterprogramm aufgerufen wurde.
	<b>Unterprogramm aufrufen</b>
<b>Unn</b>	Das Unterprogramm mit Beginn ab der Zeile nn wird aufgerufen. Beendet wird das Programm mit dem UE Befehl.
<b>URnn</b> <b>UR[Rnn]</b>	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.
<b>U*la*</b>	Das Unterprogramm beginnt in der Zeile, die durch das Label *la* gekennzeichnet ist. Beendet wird das Unterprogramm mit dem UE Befehl.
<b>UP[name]</b>	Das Unterprogramm mit dem Namen name, Beginn ab der Zeile 1, wird aufgerufen. Beendet wird das Unterprogramm mit dem UE Befehl.
<b>UP[name]Nnn</b>	Das Unterprogramm mit dem Namen name, Beginn ab der Zeile nn, wird aufgerufen. Beendet wird das Unterprogramm mit dem UE Befehl.
<b>UP[name]NRnn</b> <b>UP[name]NR[Rnn]</b>	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.
<b>UP[name]N*la*</b>	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.
	<b>Bedingter Unterprogrammaufruf</b>
	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.

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
	<b>„E“ für Bedingung erfüllt</b>
<b>UE<sub>nn</sub></b>	siehe <b>U<sub>nn</sub></b> , S. 31
<b>UER<sub>nn</sub></b> <b>UE[R<sub>nn</sub>]</b>	
<b>UE*la*</b>	siehe <b>U*la*</b> , S. 31
<b>UEP[name]</b>	siehe <b>UP[name]</b> , S. 31
<b>UEP[name]N<sub>nn</sub></b> <b>UEP[name]NR<sub>nn</sub></b> <b>UEP[name]NR[R<sub>nn</sub>]</b>	
<b>UEP[name]N*la*</b>	siehe <b>UP[name]N*la*</b> , S. 31
	<b>„N“ für Bedingung nicht erfüllt</b>
<b>UN<sub>nn</sub></b>	siehe <b>U<sub>nn</sub></b> , S. 31
<b>UNR<sub>nn</sub></b> <b>UNR[R<sub>nn</sub>]</b>	
<b>UN*la*</b>	siehe <b>U*la*</b> , S. 31
<b>UNP[name]</b>	siehe <b>UP[name]</b> , S. 31
<b>UNP[name]N<sub>nn</sub></b> <b>UNP[name]NR<sub>nn</sub></b> <b>UNP[name]NR[R<sub>nn</sub>]</b> <b>]</b>	
<b>UNP[name]N*la*</b>	siehe <b>UP[name]N*la*</b> , S. 31

## 2.20 Terminalbefehle (auch von PC bei Terminalanschluss)

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
<b>&lt;&gt;Wy</b>	Zeile y löschen (y=1 bis 4)
<b>&lt;text&gt;Wy</b>	Text in Zeile y anzeigen ab Position 1 (y=1 bis 4)
<b>&lt;text&gt;Wy;m</b>	Text in Zeile y ab Position m anzeigen (y=1 bis 4; M=1 bis 20)
	<b>Antwort: &lt;STX&gt;&lt;ACK&gt;&lt;ETX&gt;</b>

## 2.21 Achsenbefehle

---

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

**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<Rnn**  
**X<R[Rnn]**

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>** (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**

**XPmmR**

Der Parameter mm der Achse X wird ausgelesen.

**Antwort : <STX><ACK>wert<ETX>**  
mm = Parameternummer (NUR PROG)

**XPmmSzwert**  
**XPmmSRnn**  
**XPmmSR[Rnn]**

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.

## Befehl

## Bedeutung

**X0-I**

Die Achse fährt in Minusrichtung und stoppt mit dem Nullimpuls des Inkrementalencoders. Nur inkrementell, kein SSI Encoder!

**X0+I**

Die Achse fährt in Plusrichtung und stoppt mit dem Nullimpuls des Inkrementalencoders. Nur inkrementell, kein SSI Encoder!

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

**X0-^I**

Die Achse fährt den Initiator der —Richtung an. Nach der Offsetstrecke fährt die Achse weiter bis der Nullimpuls des Incrementalencoders 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 Inkrementalencoders 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**

**XrRnn**

**XrR[Rnn]**

Die vorgegebene Strecke (zwert , Inhalt von Register nn oder Register [Rnn]) wird relativ gefahren.

r = + oder – für die Laufrichtung

### **Mit Stopfbefehl über Eingang**

**XrzwertvEnnz**

**XrRnnvEnnz**

**XrR[Rnn]vEnnz**

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

**XrzwertvvEnnz**

**XrRnnvvEnnz**

**XrR[Rnn]vvEnnz**

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

<b><u>Befehl</u></b>	<b><u>Bedeutung</u></b>
	<b>Absolute Positionierung bezogen auf den MØP</b>
<b>XArzwert</b> <b>XArRnn</b> <b>XArR[Rnn]</b>	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
	<b>mit Stopbefehl über Eingang</b>
<b>XArzwertvvEnnz</b> <b>XArRnnvvEnnz</b> <b>XArR[Rnn]vvEnnz</b>	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
	<b>Absolute Positionierung bezogen auf den ELØP</b>
<b>Xerzwert</b> <b>XErRnn</b> <b>XErR[Rnn]</b>	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
	<b>mit Stopbefehl über Eingang</b>
<b>XErzwertvvEnnz</b> <b>XErRnnvvEnnz</b> <b>XErR[Rnn]vvEnnz</b>	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
	<b>Achsen Stop</b>
<b>XS</b>	Mit dem Befehl XS können alle Fahrbefehle abgebrochen werden. Die Achse stoppt mit der eingestellten Rampe.
<b>XSN</b>	Die Achse stoppt mit der eingestellten Nothalt-Rampe (Parameter P7).



## 2.22 Tastenabfrage Bedienterminal BT24 (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**

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

### 3 Minilog Befehle

#vFn.....	40	N*la*.....	15
#vnmX.....	40	N+nn.....	15
<>Wy.....	35	N+R[Rnn].....	15
<text>Wy.....	35	N+Rnn.....	15
ADnR.....	10	NE*la*.....	15
AGnR.....	9	NE+nn.....	15
AGnSZZZZZZZ.....	9	NE+R[Rnn].....	15
Annnz.....	9	NE+Rnn.....	15
Annnzmmmzxxxz.....	9	NEnn.....	15
ARnnn;mmm;xxx.....	9	NE–nn.....	15
CR.....	10	NEP[name].....	15
CT.....	10	NEP[name]N*la*.....	16
D1.....	10	NEP[name]Nnn.....	15
D2.....	10	NEP[name]NR[Rnn].....	16
D3.....	10	NEP[name]NRnn.....	16
E^nnzmmzxxxz.....	11	NER[Rnn].....	15
EASnnnnnnnn.....	12	NE–R[Rnn].....	15
EGnR.....	12	NERnn.....	15
Ennz.....	11	NE–Rnn.....	15
Ennzmmz.....	12	NN*la*.....	16
ERnn;mm;xx.....	12	NN+nn.....	16
Evnnzmmzxx.....	11	NN+R[Rnn].....	16
FN*la*.....	13	NN+Rnn.....	16
FNznr.....	13	Nnn.....	15
FP[name].....	13	N–nn.....	15
H.....	13	NNnn.....	16
IAR.....	13	NN–nn.....	16
IBR.....	13	NNP[name].....	16
IBSname.....	13	NNP[name]N*la*.....	16
ICnR.....	13	NNP[name]Nnn.....	16
ICnSbaud.....	13	NNP[name]NR[Rnn].....	16
IFL.....	14	NNP[name]NRnn.....	16
IFR.....	14	NNR[Rnn].....	16
IPn.....	14	NN–R[Rnn].....	16
ITR.....	14	NN–Rnn.....	16
ITSn.....	14	NNRnn.....	16
ITTSn.....	14	NP[name].....	15
IVR.....	14	NP[name]N*la*.....	15
<text>Wy.....	35	NP[name]Nnn.....	15

## MINILOG

NP[name]NR[Rnn] .....	15	R[Rnn]=zwert .....	25
NP[name]NRnn .....	15	R[Rnn]>R[Rmm] .....	25
NR[Rnn] .....	15	R[Rnn]>Rmm .....	25
N-R[Rnn] .....	15	R[Rnn]>zwert .....	25
NRnn .....	15	R[Rnn]B^R[Rmm] .....	24
N-Rnn .....	15	R[Rnn]B^Rmm .....	24
NWnn .....	16	R[Rnn]B^zwert .....	23
NWR[Rnn] .....	16	R[Rnn]BA <sub>nn</sub> -mm .....	22
NWRnn .....	16	R[Rnn]BE <sub>nn</sub> -mm .....	22
PA <sub>name</sub> .....	17	R[Rnn]BL <sub>m</sub> .....	22
PA_ .....	17	R[Rnn]BR <sub>m</sub> .....	23
PE .....	18	R[Rnn]BS <sub>zwert</sub> .....	22
PS <sub>name</sub> .....	17	R[Rnn]BT <sub>m</sub> .....	23
PWR .....	17	R[Rnn]BvR[Rmm] .....	24
PWSp .....	17	R[Rnn]BvRmm .....	24
QDP*. * .....	18	R[Rnn]Bv <sub>zwert</sub> .....	24
QDR .....	18	R[Rnn]BXR[Rmm] .....	24
QPE .....	18	R[Rnn]BXRmm .....	24
QP <sub>name</sub> NnnA .....	18	R[Rnn]BX <sub>zwert</sub> .....	24
QP <sub>name</sub> NnnR .....	18	R[Rnn]R .....	27
QP <sub>name</sub> R .....	20	R[Rnn]-R[Rmm] .....	26
QP <sub>name</sub> S <sub>byte</sub> .....	19	R[Rnn]-Rmm .....	26
R[Rnn]:R[Rmm] .....	26	R[Rnn]SAD <sub>y</sub> .....	29
R[Rnn]#R[Rmm] .....	25	R[Rnn]SE <sub>mm</sub> -xx.k .....	28
R[Rnn]:Rmm .....	26	R[Rnn]SN .....	27
R[Rnn]#Rmm .....	25	R[Rnn]SR[mm] .....	27
R[Rnn]:zwert .....	26	R[Rnn]SRmm .....	27
R[Rnn]#zwert .....	25	R[Rnn]SXP <sub>mm</sub> .....	27
R[Rnn]*R[Rmm] .....	26	R[Rnn]SZ .....	28
R[Rnn]*Rmm .....	26	R[Rnn]S <sub>zwert</sub> .....	27
R[Rnn]*zwert .....	26	R[Rnn]-zwert .....	26
R[Rnn]/R[Rmm] .....	26	Rnn:R[Rmm] .....	26
R[Rnn]/Rmm .....	26	Rnn#R[Rmm] .....	25
R[Rnn]/zwert .....	26	Rnn:Rmm .....	26
R[Rnn]+R[Rmm] .....	26	Rnn#Rmm .....	25
R[Rnn]+Rmm .....	26	Rnn:zwert .....	26
R[Rnn]+zwert .....	25	Rnn#zwert .....	25
R[Rnn]<R[Rmm] .....	25	Rnn*R[Rmm] .....	26
R[Rnn]<Rmm .....	25	Rnn*Rmm .....	26
R[Rnn]<zwert .....	25	Rnn*zwert .....	26
R[Rnn]=R[Rmm] .....	25	Rnn.z .....	22
R[Rnn]=Rmm .....	25	Rnn/R[Rmm] .....	26

<b>Rnn/Rmm</b> .....	26	<b>RnnST</b> .....	28
<b>Rnn/zwert</b> .....	26	<b>RnnST.z</b> .....	28
<b>Rnn+R[Rmm]</b> .....	26	<b>RnnSTT</b> .....	27
<b>Rnn+Rmm</b> .....	26	<b>RnnSTy</b> .....	29
<b>Rnn+zwert</b> .....	25	<b>RnnSTy.z</b> .....	29
<b>Rnn&lt;R[Rmm]</b> .....	25	<b>RnnSTy;m</b> .....	29
<b>Rnn&lt;Rmm</b> .....	25	<b>RnnSTy;m.z</b> .....	29
<b>Rnn&lt;zwert</b> .....	25	<b>RnnSXPmm</b> .....	27
<b>Rnn=R[Rmm]</b> .....	25	<b>RnnSZ</b> .....	28
<b>Rnn=Rmm</b> .....	25	<b>RnnSzwert</b> .....	27
<b>Rnn=zwert</b> .....	25	<b>RnnTAN</b> .....	26
<b>Rnn&gt;R[Rmm]</b> .....	25	<b>RnnWy</b> .....	29
<b>Rnn&gt;Rmm</b> .....	25	<b>RnnWy.z</b> .....	29
<b>Rnn&gt;zwert</b> .....	25	<b>RnnWy;m</b> .....	29
<b>RnnB^R[Rmm]</b> .....	24	<b>RnnWy;m.z</b> .....	30
<b>RnnB^Rmm</b> .....	24	<b>Rnn-zwert</b> .....	26
<b>RnnB^zwert</b> .....	23	<b>S</b> .....	31
<b>RnnBAnn-mm</b> .....	22	<b>S0</b> .....	32
<b>RnnBEnn-mm</b> .....	22	<b>S1</b> .....	32
<b>RnnBLm</b> .....	22	<b>SA</b> .....	33
<b>RnnBRm</b> .....	23	<b>SB</b> .....	31
<b>RnnBSzwert</b> .....	22	<b>SE</b> .....	31
<b>RnnBTm</b> .....	23	<b>SH</b> .....	32
<b>RnnBvR[Rmm]</b> .....	24	<b>ST</b> .....	32
<b>RnnBvRmm</b> .....	24	<b>SUI</b> .....	32
<b>RnnBvzwert</b> .....	24	<b>TR[Rnn]</b> .....	33
<b>RnnBXR[Rmm]</b> .....	24	<b>TRnn</b> .....	33
<b>RnnBXRmm</b> .....	24	<b>TT&lt;R[Rnn]</b> .....	33
<b>RnnBXzwert</b> .....	24	<b>TT&lt;Rnn</b> .....	33
<b>RnnCOS</b> .....	26	<b>TT&lt;zwert</b> .....	33
<b>RnnQW</b> .....	26	<b>TT=0</b> .....	33
<b>RnnR</b> .....	27	<b>TT&gt;R[Rnn]</b> .....	33
<b>Rnn-R[Rmm]</b> .....	26	<b>TT&gt;Rnn</b> .....	33
<b>RnnRAND</b> .....	27	<b>TT&gt;zwert</b> .....	33
<b>Rnn-Rmm</b> .....	26	<b>TTSR[Rnn]</b> .....	33
<b>RnnSADy</b> .....	29	<b>TTSRnn</b> .....	33
<b>RnnSEmm-xx.k</b> .....	28	<b>TTSzwert</b> .....	33
<b>RnnSIN</b> .....	26	<b>Tzwert</b> .....	33
<b>RnnSN</b> .....	27	<b>U*la</b> .....	34, 35
<b>RnnSR[Rmm]</b> .....	27	<b>UA</b> .....	34
<b>RnnSRmm</b> .....	27	<b>UE</b> .....	34

## MINILOG

---

<b>UE*</b> la* .....	35	<b>X&gt;Rnn</b> .....	36
<b>UE</b> [Rnn] .....	35	<b>X&gt;zwert</b> .....	36
<b>UEnn</b> .....	35	<b>X0-</b> .....	37
<b>UEP</b> [name] .....	35	<b>X0-^I</b> .....	38
<b>UEP</b> [name] <b>N*</b> la* .....	35	<b>X0+</b> .....	37
<b>UEP</b> [name] <b>Nnn</b> .....	35	<b>X0+^I</b> .....	38
<b>UEP</b> [name] <b>NR</b> [Rnn] .....	35	<b>X0+I</b> .....	38
<b>UEP</b> [name] <b>NRnn</b> .....	35	<b>X0-I</b> .....	38
<b>UERnn</b> .....	35	<b>XArR</b> [Rnn] .....	39
<b>UN*</b> la* .....	35	<b>XArR</b> [Rnn] <b>vvEnnz</b> .....	39
<b>Unn</b> .....	34	<b>XArRnn</b> .....	39
<b>UNnn</b> .....	35	<b>XArRnnvvEnnz</b> .....	39
<b>UNP</b> [name] .....	35	<b>XArzwert</b> .....	39
<b>UNP</b> [name] <b>N*</b> la* .....	35	<b>XArzwertvvEnnz</b> .....	39
<b>UNP</b> [name] <b>Nnn</b> .....	35	<b>XC</b> .....	36
<b>UNP</b> [name] <b>NR</b> [Rnn] .....	35	<b>XErR</b> [Rnn] .....	39
<b>UNP</b> [name] <b>NRnn</b> .....	35	<b>XErR</b> [Rnn] <b>vvEnnz</b> .....	39
<b>UNR</b> [Rnn] .....	35	<b>XErRnn</b> .....	39
<b>UNRnn</b> .....	35	<b>XErRnnvvEnnz</b> .....	39
<b>UP</b> [name] .....	34, 35	<b>XErzwert</b> .....	39
<b>UP</b> [name] <b>N*</b> la .....	34, 35	<b>XErzwertvvEnnz</b> .....	39
<b>UP</b> [name] <b>Nnn</b> .....	34	<b>XLr</b> .....	38
<b>UP</b> [name] <b>NR</b> [Rnn] .....	34	<b>XMA</b> .....	37
<b>UP</b> [name] <b>NRnn</b> .....	34	<b>XMD</b> .....	37
<b>UR</b> [Rnn] .....	34	<b>XPmmR</b> .....	37
<b>URnn</b> .....	34	<b>XPmmSR</b> [Rnn] .....	37
<b>X#E</b> .....	36	<b>XPmmSRnn</b> .....	37
<b>X#M</b> .....	36	<b>XPmmSzwert</b> .....	37
<b>X#N</b> .....	36	<b>XrR</b> [Rnn] .....	38
<b>X&lt;R</b> [Rnn] .....	37	<b>XrR</b> [Rnn] <b>vEnnz</b> .....	38
<b>X&lt;Rnn</b> .....	37	<b>XrR</b> [Rnn] <b>vvEnnz</b> .....	38
<b>X&lt;zwert</b> .....	37	<b>XrRnn</b> .....	38
<b>X= I-</b> .....	36	<b>XrRnnvEnnz</b> .....	38
<b>X= I+</b> .....	36	<b>XrRnnvvEnnz</b> .....	38
<b>X=E</b> .....	36	<b>Xrzwert</b> .....	38
<b>X=H</b> .....	36	<b>XrzwertvEnnz</b> .....	38
<b>X=M</b> .....	36	<b>XrzwertvvEnnz</b> .....	38
<b>X=N</b> .....	36	<b>XS</b> .....	39
<b>X&gt;R</b> [Rnn] .....	36	<b>XSN</b> .....	39

## 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.

<b>Befehl</b>	<b>Bedeutung</b>
	<b>G-Befehle (Wegbedingungen)</b>
<b>G00, G0</b>	Punktsteuerungsverhalten Positionierung mit der größtmöglichen Geschwindigkeit (Eilgang) ohne Interpolation mit Parameter 14.
<b>G01, G1</b>	Linearinterpolation setzen
<b>G04Tnn, G4Tnn</b>	Zeitlich vorbestimmte Programmunterbrechungen mit programmierter oder in der Steuerung festgelegter Dauer und automatischer Programmfortsetzung. n= in Sekunden mit Nachkommastellen Abbruch über Eingang 2
<b>G05, G5</b>	Warten auf Achsenstillstand aller Achsen, danach erst weiter im Programm
<b>G20Lnn</b>	Unbedingter Sprung auf Zeile nn
<b>G20L+nn</b>	Unbedingter Sprung um nn Zeilen in Plus-Richtung
<b>G20L-nn</b>	Unbedingter Sprung um nn Zeilen in Minus-Richtung
<b>G20*label*</b>	Unbedingter Sprung auf Label
<b>G20L*label*</b>	Unbedingter Sprung auf Label
<b>G20LP[name]</b>	Unbedingter Sprung ins Programm name in Zeile 1
<b>G21zLnn</b>	Bedingter Sprung auf Zeile nn z = E oder N
<b>G21zL+nn</b>	Bedingter Sprung um nn Zeilen in Plus Richtung z = E oder N
<b>G21zL-nn</b>	Bedingter Sprung um nn Zeilen in Minus Richtung z = E oder N
<b>G21z*label*</b>	Bedingter Sprung auf Label z = E oder N
<b>G21zL*label*</b>	Bedingter Sprung auf Label z = E oder N

## MINILOG

<b>Befehl</b>	<b>Bedeutung</b>
<b>G21zLP</b> [name]	Bedingter Sprung ins Programm name in Zeile 1 z = E oder N
<b>G22L</b> nn	Unterprogrammaufruf des Unterprogramms nn Unterprogramm durch G98Lnn im Programm gekennzeichnet
<b>G22*</b> label*	Unterprogrammaufruf des Unterprogramms *label*
<b>G22P</b> [name]	Unterprogrammaufruf des Programms name
<b>G23L</b> nn	Unterprogramm sofort beenden und Sprung zur Zeile nn
<b>G23*</b> label*	Unterprogramm sofort beenden und Sprung zum Label
<b>G74</b>	Referenzlauf aller Achse minus Richtung
<b>G74 x</b>	Referenzlauf Achse x= X oder Y
<b>G79L</b> nn	Automatischer Unterprogrammaufruf am Zeilenende Unterprogramm durch G98Lxx im Programm gekennzeichnet
<b>G80</b>	Beenden von G79
<b>G90</b>	Positionierung Absolutmaß bezogen auf Referenzpunktzähler Parameter 20
<b>G91</b>	Kettenmaß Positionierung
<b>G92</b>	Nullpunktverschiebung Parameter 20 setzen
<b>G98L</b> nn	Unterprogramm Anfang und Deklaration nn Name des Unterprogramms maximal 6 Zeichen
<b>G99</b>	Unterprogramm Ende
	<b>M-Befehle (Zusatzfunktionen)</b>
<b>M00, M0</b>	Programmierter Halt Fortsetzung des Programms durch Setzen von Eingang 2
<b>M01, M1</b>	Programmierter Halt, wenn Eingang 3 eingeschaltet ist Fortsetzung des Programms durch Setzen von Eingang 2
<b>M02, M2</b>	Programmende

<b>Befehl</b>	<b>Bedeutung</b>
<b>M03, M3</b>	Einschalten der Spindeldrehung im Rechtslauf Ausgang 1 ein; Ausgang 2 aus
<b>M04, M4</b>	Einschalten der Spindeldrehung im Linkslauf Ausgang 1 aus; Ausgang 2 ein
<b>M05, M5</b>	Schnellstmögliches Anhalten der Spindeldrehung Ausgang 1 aus; Ausgang 2 aus
<b>M07, M7</b>	Kühlmittel 2 ein Ausgang 3 aus; Ausgang 4 ein
<b>M08, M8</b>	Kühlmittel 1 ein Ausgang 3 ein; Ausgang 4 aus
<b>M09, M9</b>	Kühlmittel aus Ausgang 3 aus; Ausgang 4 aus
<b>M10</b>	Klemmung ein; Ausgang 5 ein
<b>M11</b>	Klemmung aus; Ausgang 5 aus
<b>M68</b>	Werkstück spannen ; Ausgang 6 ein
<b>M69</b>	Werkstück entspannen ; Ausgang 6 aus



## 5 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 MiniLog-Comm 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 ein X bzw. Y eingefügt werden (1 oder 2 auch möglich).
- Beispiel: XP15 ist die Beschleunigungsrampe für die X-Achse.
- Auch innerhalb des Programms ist es möglich, Parameter zu ändern.
- Parameter können entweder beschrieben oder ausgelesen werden.
- P48 und P49 können nur ausgelesen werden.
- P19 bis P22 sind Zähler, die vom Programm bei Verfahren der Achsen laufend aktualisiert werden.


## 5.1 Parameterliste

Nr.	Bedeutung	Auslieferungszustand
<b>P01</b>	<p>Art der Bewegung</p> <p>0 = rotatorisch Rundtisch, 1 Endschalter für Initialisierung</p> <p>1 = linear Lineartisch, 2 Endschalter: Initialisierung und Begrenzung –Richtung Begrenzung +Richtung</p>	0
<b>P02</b>	<p>Maßeinheit der Bewegung</p> <p>1 = Schritt 2 = mm 3 = Zoll 4 = Grad</p>	1
<b>P03</b>	<p>Umrechnungsfaktor Spindelsteigung</p> <p>1 Schritt entspricht ...</p> <p>Bei P03 = 1 (Schritte) ist der Umrechnungsfaktor 1</p> <p>Berechnung des Umrechnungsfaktors:</p> $\text{Umrechnungsfaktor} = \frac{\text{Spindelsteigung}}{\text{Motorschrittzahl pro Umdrehung}}$ <p>Beispiel: 4 mm Spindelsteigung 200-schrittiger Motor = 400 Schritte/U im Halbschrittbetrieb</p> $\text{Umrechnungsfaktor} = \frac{4}{400} = 0,01$	1
<b>P04</b>	<p>Start-/Stoppfrequenz</p> <p>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.</p> <p>Eingabe der Frequenz in Hz</p>	400
<b>P05</b> <b>P06</b>	nicht belegt	
<b>P07</b>	<p>Achsenrampe für Nothalt</p> <p>Eingabe in 4000-Hz/s-Schritten</p>	100 000

Nr.	Bedeutung	Auslieferungszustand
<b>P08</b>	$f_{\max}$ MØP, Fahrfrequenz beim Initialisieren Eingabe in Hz (ganzzahlige Werte)	4000
<b>P09</b>	Rampe MØP für Initialisierung, zugehörig zu Parameter P08 Eingabe in 4000-Hz/s-Schritten	4000
<b>P10</b>	$f_{\min}$ MØP, Fahrfrequenz beim Verlassen der Endschalter Eingabe in Hz	400
<b>P11</b>	MØP Offset für Endschalter Plusrichtung Abstand des mechanischen Nullpunkts MØP (Referenzpunkt) vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt	0
<b>P12</b>	MØP Offset für Endschalter Minusrichtung Abstand des mechanischen Nullpunkts MØP vom Schaltpunkt des Endschalters. Einheit: wie in Parameter P02 festgelegt	0
<b>P13</b>	Beruhigungszeit MØP Wartezeit bei Initialisierung Eingabe in ms	20
<b>P14</b>	$f_{\max}$ Lauffrequenz bei Positionierbefehlen Eingabe in Hz (ganzzahlige Werte) (max. 40 000)	4000
<b>P15</b>	Rampe für Lauffrequenz (P14) Eingabe in 4000-Hz/s-Schritten (4000 bis 500 000 Hz/s)	4000
<b>P16</b>	Beruhigungszeit Position Wartezeit nach Ausführung eines Fahrbefehls Eingabe in ms	20

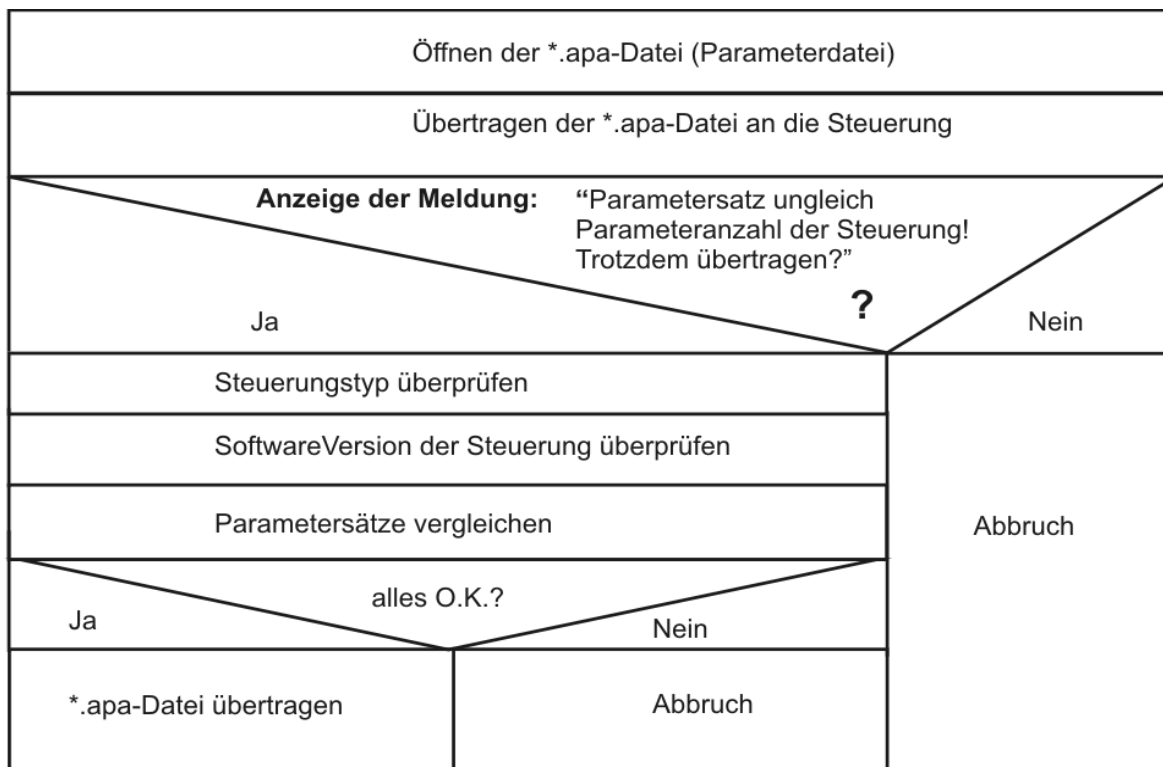
Nr.	Bedeutung	Auslieferungszustand
<b>P17</b>	<p>Boost (definiert in P42)</p> <p>0 = aus 1 = ein während der Motor fährt 2 = ein bei Hochlauf und Absenkung der Fahrfrequenz (Rampe)</p> <p><u>Anmerkungen:</u></p> <p>Der Booststrom kann in Parameter P42 programmiert werden.</p> <p>Mit dem Parameter P17 wird festgelegt, wann die Steuerung auf Booststrom umschaltet.</p> <p>P17 = 1 bedeutet, dass bei fahrendem Motor immer der Booststrom fließt. Bei Stillstand des Motors wird auf Stoppstrom umgeschaltet.</p>	0
<b>P18</b>	nicht belegt	
<b>P19</b>	<p>Elektronischer-Nullpunkt-Zähler</p> <p>Dient zur Bestimmung von Arbeitspunkten, kann bei Achsenstillstand gesetzt und ausgelesen werden.</p>	0
<b>P20</b>	<p>Mechanischer-Nullpunkt-Zähler</p> <p>Zählt Impulse bezogen auf den mechanischen Nullpunkt. Kann bei Achsenstillstand ausgelesen werden. Am MØP wird P20 automatisch null gesetzt.</p>	0
<b>P21</b>	<p>Absolutwertzähler</p> <p>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.</p>	0
<b>P22</b>	<p>Encoderzähler</p> <p>Gibt die aktuelle Encoderposition an.</p>	0
<b>P23</b>	<p>Achsenbegrenzung pos. Richtung +</p> <p>Bei Erreichen dieser Schrittzahl wird der Lauf in +Richtung abgebrochen.</p> <p>0 = keine Begrenzung</p>	0

## MINILOG

Nr.	Bedeutung	Auslieferungszustand
<b>P24</b>	Achsenbegrenzung neg. Richtung - Bei Erreichen dieser Schrittzahl wird der Lauf in –Richtung abgebrochen. 0 = keine Begrenzung	0
<b>P25</b>	Spielausgleich Gibt die Schrittzahl an, um die die Sollposition in der gewählten Richtung überfahren und anschließend in umgekehrter Richtung angefahren wird. 0 = kein Spielausgleich	0
<b>P26</b>	nicht belegt	
<b>P27</b>	Initiatorotyp 0 = PNP-Öffner 1 = PNP-Schließer	0
<b>P28 bis P33</b> nicht belegt		
<b>P34</b>	Encodertyp 0 = keiner 1 = inkrementell 2 = serielle Schnittstelle SSI Binär Code 3 = serielle Schnittstelle SSI Gray Code   <b>Achtung:</b> Encodertyp <b>korrekt</b> eingeben! Falsches Parametrieren führt zu Beschädigungen!	0
<b>P35</b>	Auflösung bei Absolut-Encoder (SSI) Eingabe: maximale Auflösung in Bit (max. 31 Bit)	10
<b>P36</b>	Encoderfunktion 0 = Zähler	0
<b>P37</b>	nicht belegt	
<b>P38</b>	Encoder Vorzugsdrehrichtung 0 = plus 1 = minus	0
<b>P39</b>	Encoder Umrechnungsfaktor 1 Inkrement entspricht ...	1

Nr.	Bedeutung			Auslieferungszustand
		<b>MCC-2/MCC-1 in Stufen von 0,1 A</b>	<b>MCC-2 LIN in Stufen von 0,04 A</b>	<b>MCC-1 bzw MCC-2 / MCC-2 LIN</b>
<b>P40</b>	<b>Stoppstrom</b> Bereich Stufen(Eingabe)	0 bis 2.5 A 0 bis 25	0 bis 1 A 0 bis 25	2 / 2
<b>P41</b>	<b>Laufstrom</b> Bereich Stufen(Eingabe)	0 bis 2.5 A 0 bis 25	0 bis 1 A 0 bis 25	6 / 6
<b>P42</b>	<b>Booststrom</b> Bereich Stufen(Eingabe)	0 bis 2.5 A 0 bis 25	0 bis 1 A 0 bis 25	10 / deaktiviert
<b>P43</b>	Stoppstromüberhöhungszeit in ms			20
<b>P44</b>	nicht belegt			
<b>P45</b>	Schrittauflösung 1 bis 256 1 = Vollschritt                      10 = 1/10 Schritt 2 = Halbschritt                      16 = 1/16 Schritt 4 = 1/4 Schritt                      128 = 1/128 Schritt 8 = 1/8 Schritt                      256 = 1/256 Schritt			4
<b>P46</b>	Stromformung (CS= Current Shaping) 0 = Aus    1 = Ein Empfohlene Einstellung: P46 = 1    (siehe Kap. 8)			1
<b>P47</b>	Chopperfrequenz 0 = niedrig    1 = hoch Der Wert der Chopperfrequenz hängt von P46 ab: Wenn P46 = 0, dann gilt: P47 = 0:    16 kHz P47 = 1: 22.5 kHz Wenn P46 = 1, dann gilt: P47 = 0:    50 kHz P47 = 1:    75 kHz Empfohlene Einstellung: P47 = 1			1
<b>P48</b>	Endstufentyp (nur lesen) 0 = Linear    1 = Chopper			(nur lesen)
<b>P49</b>	Endstufentemperatur in °C (nur lesen) (nur bei linearem Endstufentyp)			(nur lesen)

## 5.2 Übertragen des Parametersatzes in die Steuerung



## 6 Programmierbeispiele

### 6.1 Allgemein

Zeilennr.	Programm	Kommentar
ZNR1	E^1R2R NN+1 X=H NE+1 XS H A1R2R	Einlesen von 2 Eingängen, wenn beide 0 und Motor läuft, dann Motor stoppen sonst weiter mit nächster Zeile. Wenn Motor steht, Ausgang 1 und 2 rücksetzen.
ZNR2	E^1S2R NN+1 X=H NN+1 XL+ A1S	Wenn erster Eingang 1 und Motor steht, dann Lauf in positive Richtung starten und Ausgang 1 setzen.
ZNR3	E^1R2S NN+1 X=H NN+1 XL- A2S	Eingang 2 = Lauf in negative Richtung und Ausgang 2 setzen, wenn Motor läuft.
ZNR4	E^3S NN+1 X=H NN+1 N+3	Wenn Eingang 3=1 und Motor steht, dann Referenzfahrt auf Initiator, danach Programmfortsetzung Zeile 1.
ZNR5	E^4S NN-4 X=H NN-4 N+3	Wenn Eingang 4=1 und Motor steht, dann Positionierung relativ.
ZNR6	N1	Rücksprung zu Zeile 1
ZNR7	X0- A3S H A3R N1	Referenzfahrt auf Initiator Minus Richtung ausführen und warten bis Motor steht, danach Rücksprung Zeile 1. Beim Referenzlauf Ausgang 3 setzen
ZNR8	X+1000 A4S	Positionierung 1000 Schritte in plus Richtung. Bei der Positionierung den Ausgang 4 setzen.
ZNR9	E^5S1 NN+1 XS H A4R N1	Hier warten bis Eingang 5=1, dann Motor stoppen und zurück zum Programmmanfang, wenn Positionierung beendet
ZNR10	X=H NN-1 A4R N1	Positionierung beendet ? Wenn ja, dann Ausgang 4 rücksetzen und Rücksprung zu Zeile 1

### 6.2 Programmbeispiel A/D Wandler

Programm	Kommentar
*START*	
R2SAD1	Lädt das Register 2 mit AD Karte Ch 1
R3SAD2	Lädt das Register 3 mit AD Karte Ch 2
R2W2	Schreiben des AD-Wertes in die Anzeige
R3W3	Schreiben des AD-Wertes in die Anzeige
N*START*	Rücksprung zum Start



### 7 Speicherung der Programme, Parameter und Register

---

Programme und Parameter werden in MiniLog-Comm editiert, zur Steuerung überspielt und gespeichert. Während des Programmablaufs werden Register und/oder Zähler vom Programm beschrieben. Solange die Steuerung eingeschaltet ist, bleiben diese Daten erhalten. Was mit den Daten nach Ausschalten der Steuerung geschieht, hängt ab vom Typ der eingebauten Speicherbausteine:

<b>Flash-EPROM Speicher</b>	<p>Vom Programm beschriebene Register oder Zähler werden bei Ausschalten der Steuerung <u>nicht</u> gespeichert.</p> <p>Falls diese Daten weiter benötigt werden, müssen sie vor Ausschalten der Steuerung in MiniLog-Comm abgespeichert und zur Steuerung zurück gesendet werden.</p>
<b>RAM Speicher</b>	<p>Die ersten 100 Register werden in ein flüchtiges RAM gespeichert: Vorteil: Schnellerer Zugriff Nachteil: Daten gehen bei Ausschalten verloren.</p> <p>Ab Register 101 werden die Daten in ein serielles RAM (<b>SRAM</b>) gespeichert: Vorteil: Daten gehen bei Ausschalten nicht verloren und stehen bei erneutem Einschalten zur Verfügung. Nachteil: Langsamer Zugriff</p>

## 8 Stromformung CS

Mit Stromformung (CS= Current Shaping) wird hier eine schaltungstechnische Maßnahme beschrieben.

CS bewirkt, dass der tatsächliche Phasenstromverlauf über einen großen Drehzahlbereich der vorgewählten Stromkurve entspricht. Wird ein Schrittmotor ohne CS betrieben, so kommt es schon bei relativ niedrigen Drehzahlen zu Abweichungen der Stromkurve vom Sollwert.

Für 1/20-Sinusbetrieb ergeben sich bei mittleren Drehzahlen die in folgender Abbildung dargestellten Veränderungen:

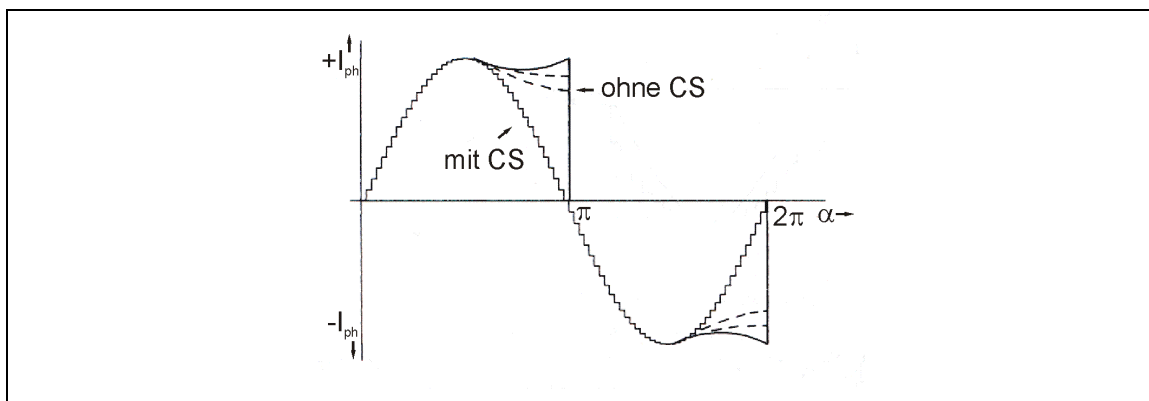


Abb. 1: Stromformung CS

Diese typischen Verformungen treten auch bei allen anderen Kurvenformen auf. Die Verformungen werden verursacht durch die Motorinduktivität und die mit der Drehzahl zunehmende generatorische Rückwirkung des Schrittmotors.

Der auftretende „Stromschwanz“ macht eine exakte Phasenstromregelung nur mit Current Shaping (CS= 4-Quadrantenstromregelung) möglich. Die Amplitude des „Stromschwanzes“ ist über eine Umdrehung beträchtlichen Schwankungen unterworfen und kann den Motor zu Resonanzen anregen, die zu Schrittverlusten oder „Außertrittfallen“ führen können.

Mit zugeschalteter Stromformung tritt der „Stromschwanz“ nicht auf, die ideale Stromkurve bleibt erhalten.

Wir empfehlen deshalb, CS im hohen Strom- und Drehzahlbereich zu verwenden.

Die Stromformung CS ist durch den Parameter P46 einschaltbar (siehe Kap. 5.1).

## 9 Stichwortverzeichnis

---

### A

A/D-Wandler 30  
Achsenbefehle  
    Endstufe 38  
    Freier Lauf 39  
    Initialisierung 38  
    Parameter lesen/laden 38  
    Status abfragen 37  
    Stop 40  
    Warten 37  
Adressierung  
    Direkt 6  
    Indirekt 7  
    mit Label 7  
Anzeige schalten 36, 37  
Ausgänge  
    Lesen 10, 12  
    MCC-1 13  
    Schalten 10  
Automatikstart 14

### B

Baudrate  
    lesen 14  
    setzen 14  
Bedienterminal 15  
Bedingungsbyte 7  
Befehlscode 5  
Broadcast 8

### C

Chopper 54  
Current Shaping 58

### D

DIN-Befehle 46

### E

Eingänge  
    MCC-1 13  
    ODER Verknüpfung abfragen 12  
    Status lesen 13  
    UND Verknüpfung abfragen 12  
    Warten bis Zustand erfüllt 12  
ELØP 40  
Endschalter 50, 51

### F

Flash-EPROM 57

### L

Label 7  
Linear 50, 54

### M

Mechanischer Nullpunkt 38

### P

Parameter 5, 6, 49  
Parameterliste 50  
Passwort  
    Freigabezustand lesen 18  
    Freigabezustand setzen 18  
Positionierung  
    absolut 40  
    bezogen auf ELØP 40  
    bezogen auf MØP 40  
    relativ 39  
Programm  
    Programmzeilen wiederholen 17  
    Unterbrechung 14  
Programm- u. Dateiverwaltung (nur ü. Rechner)  
    Progr. Abbruch 19  
    Progr. lesen mit Abfrage 21  
    Progr. Start ab Zeile 19  
    Progr. Übertragung mit Abfrage 20  
    Progr. Zeile lesen 19  
Programmname 7  
Programmzeile 6

### R

RAM 57  
    Inhaltsverz. auslesen 15  
Referenzsuchlauf 38  
Register 6, 57  
Registerbefehle  
    Ausgänge setzen 23  
    Auslesen 28  
    beschreiben mit A/D-Wandlerwerten 30  
    beschreiben mit Dezimalwert 28  
    beschreiben mit Zeilenr. 28, 29  
    beschreiben mit Zeilenr. Nothalt 28  
    beschreiben über Eingänge 29  
    Bit testen 24  
    Bitweise schieben 23  
    Inhalte vergleichen 26

Rechenoperationen

*Addieren* 26  
*Cosinus* 27  
*Dividieren* 27  
*Multiplizieren* 27  
*Quadratwurzel* 27  
*Sinus* 27  
*Subtrahieren* 27  
*Tangens* 27  
*Zufallszahl* 28

Reset Steuerung 11

**S**

Schreibausgabe über serielle Schnittstelle 11

Speicherung 57

Spielausgleich 53

Sprungbefehle

absolut 16  
 absolut E = Bedingung erfüllt 16  
 absolut N = Bedingung nicht erfüllt 17  
 relativ 16  
 relativ E = Bedingung erfüllt 16  
 relativ N = Bedingung nicht erfüllt 17

SRAM 57

Start-/Stopfrequenz 50

Stromformung 58

Synchronstart 33

Systemanpassung im Programmablauf  
 Automatikstart 14

Systemstatus lesen

allgemein 32  
 binär 32  
 dezimal 33

**U**

Unterprogramm

abbrechen 35  
 aufrufen 35  
 bedingter Aufruf 35  
 beenden 35

**V**

Versionsabfrage 15

**W**

Wegbedingungen 46

**Z**

Zeitschleifen 34

Zusatzfunktionen 46

