

Programming Manual for OMC and TMC



Programming Manual MINILOG

of the OMC and TMC Controllers

TRANSLATION OF THE GERMAN ORIGINAL MANUAL

© 2009

All rights with:

Phytron GmbH

Industriestraße 12

82194 Gröbenzell, Germany

Tel.: +49 8142/503-0

Fax: +49 8142/503-190

Every possible care has been taken to ensure the accuracy of this technical manual. All information contained in this manual is correct to the best of our knowledge and belief but cannot be guaranteed. Furthermore we reserve the right to make improvements and enhancements to the manual and / or the devices described herein without prior notification.

We appreciate suggestions and criticisms for further improvement. Please send your comments to the following E-mail address: doku@phytron.de

Contents

1	Structure of the Minilog Instructions	4
1.1	Instruction Code	4
1.2	Addressing Mode	5
1.3	Conditional Instructions.....	6
1.4	Data and Telegram Format.....	6
2	MINILOG Instructions.....	8
2.1	Outputs	8
2.2	A/D Converter (Option OMC/TMC)	9
2.3	Reset.....	9
2.4	Write Instructions via Serial Interface.....	9
2.5	Input requests	10
2.6	Program Manipulation at Emergency Stop (ONLY PROG).....	12
2.7	Program Interruption	12
2.8	System Adaption during Program Execution	13
2.9	Interpolation Instructions.....	15
2.10	Jump Instructions (ONLY PROG)	16
2.11	Repeating of Program Lines	17
2.12	Ending or Interruption of a Program Call (ONLY PROG).....	18
2.13	Program and Data Management (ONLY PC).....	18
2.14	Register Instructions	22
2.15	System Status (ONLY PC).....	30
2.16	Store Data into Flash EPROM	32
2.17	Time Loops	32
2.18	Subroutines (ONLY PROG)	33
2.19	Terminal Instructions (also by PC in case of terminal connection)	34
2.20	Axes Instructions.....	35
2.21	Function Keys Read Out on Terminal BT24 (also by PC).....	39
3	List of Minilog Instructions	40
4	List of DIN Instructions	45
5	Parameters.....	47
6	About Parameters 30 and 31	52
7	Program Example A/D Converter.....	53
8	Index.....	54

1 Structure of the Minilog Instructions

1.1 Instruction Code

Xrvalue	X	<p>The bold characters represent the instruction code and must be used unchanged.</p> <p>In this example: X represents the motion instruction code for relative positioning of the X-axis.</p> <p>This manual shows only the instruction codes for the single axis controller (OMC), where the axis is generally named “X”.</p> <p>For multiple axis controllers (also Master/slave up to 8 axes) the corresponding characters X, Y, ... or 1, 2, ... must be used.</p>
	r	<p>Small letter require the input of the characters or values which are described in the column <i>Meaning</i>.</p> <p>In this example: r = running direction + or – .</p>
	value	<p>In this example a running distance of 1000 is fed in. The corresponding unit (e.g. steps) of the particular input is defined by parameters. For the specific parameters refer to chap. 5.</p>
Example: X+1000		<p>Relative motion instruction for the X-axis: Go 1000 steps to the + direction.</p>

Important:

- **All characters and signs, belonging to one single instruction, must be written without a blank.**
- **The instructions themselves must be separated by a blank.**
- **Leading zeros in an instruction are ignored (Example: the instruction A001S is realized as A1S)**
- **Instructions, which cannot be used in the program and direct mode, are marked with:**
 1. **Instruction only used in the program (ONLY PROG)**
 2. **Instruction only used in the PC direct mode (ONLY PC)**

Exception: In the instruction group “System Status (ONLY PC)”, chapter 2.13, page 18, the first program name must be separated by a blank from the second program name or the following alphanumeric part of the instruction code.

1.2 Addressing Mode

For instructions, where at least one register is used as an operator, two addressing modes are available: The **Direct Addressing Mode** and the **Indirect Addressing Mode**. In this programming manual the meaning of the basic instruction is always explained for the **Direct Addressing Mode**. The variations of the **Indirect Addressing Mode** are listed for the sake of completeness. The first named register within an instruction code is always the destination register for the result.

Example for Direct Addressing Mode:

<i>Instruction</i>	<i>Meaning</i>
RnnBE_{nn–mm}	The status of the inputs nn to mm is written as a binary value into the register nn.

Example: R1BE1–8

Status of the inputs 1 to 8 is e.g. **1010 0101**. This binary value is written into the register 1. After the Instruction was carried out, the register content is 165 decimal.

Example for Indirect Addressing Mode:

<i>Instruction</i>	<i>Meaning</i>
R[Rnn]BE_{nn–mm}	Indirect Addressing Mode: The status of the inputs nn to mm is written as a binary value into the register which is addressed by the register [Rnn].

Example: **R1S10 [R1]BE1–8**

The addressing register **[R1]** is set to 10. The status of the inputs 1 to 8 is e.g. 1010 0101. This binary value is written into the register 10, which was addressed by the register 1. After the instruction was carried out, the content of the register 10 is 165 decimal.

Addressing with Label

In case of jump calls (page 166) and subroutine calls (page 33) the start or destination line can be set in the instruction code with a label (*la*), which is assigned to this program line. A label is defined between two * and can have up to 6 alphanumeric signs. Max. 100 labels can be used in one program.

Example: *[label name]*

Program name:

Program names [name] in the instruction code can have up to 8 alphanumeric signs.

1.3 Conditional Instructions

The execution of some instructions (e.g. jumps or subroutine calls) can be combined with a condition. Before a conditional jump etc. can be used, the condition byte has to be set, for example by an input request (see page 9) or a register comparison (see page 23).

Possible states of the condition byte:

E = Condition fulfilled **N** = Condition not fulfilled

The state of the condition byte remains stored until it is changed again.

All instructions which set no condition delete the condition request.

1.4 Data and Telegram Format

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

The **send telegram** from PC via RS232 is defined as:

Without check-sum: <STX> | Address | Instruction | <ETX> | <CR> | <LF>

With check-sum: <STX> | Address | Instruction | Separator | Checksum | <ETX> | <CR> | <LF>

The **response telegram** (always for address 0-9, A-F) is defined as:

<STX> | **ACK** | Answer | <ETX> | <CR> | <LF> or
<STX> | **ACK** | <ETX> | <CR> | <LF> or
<STX> | **NAK** | <ETX> | <CR> | <LF>

	Meaning
<STX>	<STX> (Start of Text, 02 _H): It is exclusively used as the start code for a new telegram
Address	Address of the controller, the range of the address byte is 0 to 9 and A to F (30 _H ...39 _H and 41 _H ...46 _H). Additional the Broadcast ¹ address @ (40 _H) is used.
Instruction	MINILOG instruction code
Separator	: Colon (3A _H) as separator, to distinguish between usable data and checksum.
Checksum	Upper byte of the checksum value (see below for the algorithm to calculate the checksum)
	Lower byte of the checksum value (calculation see below)
<ETX>	(End of Text, 03 _H) this code indicates the end of the telegram.
<CR>	(Carriage Return, 0D _H) this code indicates the carriage return.
<LF>	(Line Feed, 0A _H) this code indicates the line feed.
ACK	(Acknowledge 06 _H), the instruction has been confirmed.
NAK	(Negative Acknowledge 15 _H), the instruction has been negatively confirmed.
Answer	Answer as number or string, e.g. E or N

The checksum CS is defined by summing up all bytes, beginning with the address byte and including the separator (:) in an exclusive-OR-operation:

$$CS = \text{address} \oplus \text{data byte 1} \oplus \text{data byte 2} \dots \oplus \text{data byte n} \oplus \text{separator}$$

The checksum is calculated as one 8-bit binary value (00_H to FF_H). This byte is taken apart in its upper and lower byte (nibbles). After the HEX values of the two nibbles have been transferred to the corresponding two ASCII characters (0 to 9 instead of 0_H to 9_H and A to F instead of A_H to F_H, that means to each nibble 30_H or rather 37_H is mathematically added), the checksum is written in the telegram.

The OMC/TMC also calculates (Exclusive OR) the checksum of the received data. The telegram will be rejected if a difference to the received checksum is detected, and the error is confirmed by NAK.

If there is no need to validate the contents of the telegram, the checksum monitoring can be set off. Instead of the checksum bytes, **two X** characters will be accepted, e.g.:

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

¹ Broadcast: All axes receive and evaluate the telegram. To avoid bus-conflicts caused by the response of all axes nearly within the same time, the response of the controllers is suppressed by addressing with “@”.

2 MINILOG Instructions

2.1 Outputs

<i>Instruction</i>	<i>Meaning</i>
	Set Outputs
Annnz	Set one or several outputs at the same time. nnn, mmm, xxx → number (ID) of the output
Annnzmmmmzxxxz	z = S → set z = R → reset Example: A1S2R3S Output 1 and 3 ON, output 2 OUT
	Read Output Status
AGnR	Read the state of the output groupes n. (ONLY PC) Example: AG2R The 2nd output groupe is read Response : <STX><ACK>nnnnnnnn<ETX><CR><LF> (ONLY PC) n = 0 Output OFF n = 1 Output ON
	Set the output group outputs
AGnSzzzzzzzz	Set the output group n=1 or 2, z= 0 or 1. z must always have 8 places Example: AG1S10101001 The 1. output group is set with the information '10101001'
	Read Output Status
ARnnn;mmm;xxx	The state of the outputs nnn, mmm, xxx is read. (ONLY PC) Response : <STX><ACK>nnn<ETX><CR><LF> n = 0 Output OFF n = 1 Output ON Important: Set a ; between the output numbers.

2.2 A/D Converter (Option OMC/TMC)

<i>Instruction</i>	<i>Meaning</i>
ADnCxSy	A/D converter setting is transmitted. n → A/D converter address: n=0 or n=1 x → A/D converter channel: x=0 to 3 y → transmission method: y=0 bipolar y=1 unipolar
ADnCxR	A/D converter setting is read. Response : <STX><ACK>0<ETX><CR><LF> or <STX><ACK>1<ETX><CR><LF> (ONLY PC)

2.3 Reset

<i>Instruction</i>	<i>Meaning</i>
CR	The controller is reset by the interface.
CT	The display of the terminal is deleted. Response : <STX><ACK><ETX><CR><LF> (ONLY PC)

2.4 Write Instructions via Serial Interface

<i>Instruction</i>	<i>Meaning</i>
	Informations can be carried out via the 3 serial interfaces (X31, X32, X9). The writing is carried out without formatting. s = 1,2 or 3 → interface name 1 → interface RS232 (X31) 2 → interface RS232 (X32) 3 → terminal interface (X9)
Ds <text>	The bracketed expression is carried out.
DsRnn	The content of the register nn is carried out.
DsR[Rnn]	The content of the register which is addressed by register nn is carried out.

MINILOG

<i>Instruction</i>	<i>Meaning</i>
DsxPmm	Parameter mm of the axis x is carried out. mm = 1 to 45 → number (ID) of the parameter x = 1 to 8 or X,Y,Z,W,5,6,7,8 → axis ID Example: D24P10 The parameter 10 of the axis 4 is carried out via the interface X32 .

2.5 Input requests

<i>Instruction</i>	<i>Meaning</i>
	Logical AND
E[^]nnzmmzxxz	The inputs nn, mm, xx are tested as AND condition. Only if the AND condition is fulfilled the condition byte is set. Otherwise the condition byte is reset. nn. mm. xx → input number z = S → input ON z = R → input OFF Example: E[^]1S2R3S The input states 1, 2 and 3 are read out. If input 1 is set, input 2 reset and input 3 set, the AND condition is fulfilled and the condition byte is set. Now a conditional jump or a conditional call of a subprogram can be carried out. Response: <STX><ACK> E <ETX><CR><LF> or <STX><ACK> N <ETX><CR><LF> (ONLY PC)
	Logical OR
Evnnzmmzxx	The inputs nn, mm, xx are tested as OR condition. Only if the OR condition is fulfilled the condition byte is set. Otherwise the condition byte is reset. nn. mm. xx → input number z = S → input ON z = R → input OFF Example: Ev1S2R3S The input states 1, 2 and 3 are read out. If input 1 is set or input 2 reset or input 3 set, the AND condition is fulfilled and the condition byte is set. Now a conditional jump or a conditional call of a subprogram can be carried out. Response: <STX><ACK> E <ETX><CR><LF> or <STX><ACK> N <ETX><CR><LF> (ONLY PC)

<i>Instruction</i>	<i>Meaning</i>
	Wait for Condition Fulfilled
Ennz	Wait for the preset input condition. The program stops until the preset input condition is fulfilled. The condition byte is not affected. (ONLY PROG)
Ennzmmz	When reading the status of several inputs, one input after the other is read out (no AND linking). The condition byte is not affected. (ONLY PROG)
	Example: E1S2R3S
	The status of the inputs 1, 2 and 3 are read. After the input 1 is set, the input 2 is read. After the input 2 is reset, the input 3 is read. After the input 3 set, the reading Instruction is done and the program continues. After the instruction end the inputs1 and 2 can have another state.
	Program start and stop via input (in the master device)
EBSx	Set input for program stop. x=0 to 16(OMC) or 32(TMC). Input stops the program, when it jumps to 0.
EBR	The input is read via interface.
ESSx	The input is set for program start. x=0 to 16(OMC) or 32(TMC). Input starts the program again, when it jumps to 1 and program stop is again on 1.
ESR	Input is read back via interface
	Read input group
EGnR	The input group n is read. (ONLY PC) n=1 to 16
	Response : <STX><ACK>nnnnnnnnnnnnnnnnnn<ETX><CR><LF> n = 0 input is reset n = 1 input is set
ERnn;mm;xx	The Status of the inputs nn, mm, xx is read (ONLY PC).
	Response: <STX><ACK>nnn<ETX><CR><LF> n = 0 input is reset n = 1 input is set
	Important: Set a ; between the input numbers.

2.6 Program Manipulation at Emergency Stop (ONLY PROG)

<i>Instruction</i>	<i>Meaning</i>
FE	Error byte Reset in case of SFI (S tep f ailure i ndication).
FNznr	The program line is defined at which the program has to be continued in the case of an emergency stop.
FN*la**	The program line, at which the program has to be continued in the case of an emergency stop, is defined by a label.
FP[name]	Indicates the program for an emergency stop. In the case of an emergency stop, a jump to the named program is initiated.

2.7 Program Interruption

<i>Instruction</i>	<i>Meaning</i>
H	The program waits here until all axes have stopped. (ONLY PROG)

2.8 System Adaption during Program Execution

<i>Instruction</i>	<i>Meaning</i>
	Number of Axes
IAR	The number of existing axes is requested (ONLY PC). Response: <STX><ACK>n<ETX><CR><LF>
	Automatic Start
IBSname	The name of the start program is written into the Auto Start register. If the REMOTE/LOCAL switch is in the LOCAL position the program execution starts here. Response: <STX><ACK><ETX><CR><LF> or <STX><NAK><ETX><CR><LF> (ONLY PC)
IBR	The name of the Auto Start program is read (ONLY PC). Response: <STX><ACK>name<ETX><CR><LF>
	Read/Set Baudrate (ONLY PC)
ICnSbaud	Set the baudrate for the OMC/TMC interfaces. n = 1 → COM 1, n = 2 → COM 2 of OMC/TMC baud = Baudrate (9600, 19200, 38400, 57600 or 115200 Baud)
ICnR	Baudrate setting of the OMC/TMC interfaces is read out. n = 1 → COM 1, n = 2 → COM 2 of OMC/TMC
	Set power stage monitoring
IESn	n=0 : Power stage monitoring is off n=1: Power stage monitoring is on
	Remote/Local Reversing (ONLY PC)
IFR	The controller is reversed to the Remote function. If a program is running, it is canceled. If the switch is positioned to Local, the position Remote is simulated. Response: <STX><ACK><ETX><CR><LF>
IFL	The controller is reversed to the function Local, if the Remote/Local switch is on the position Local. If the switch is on Remote position, it is not reversed. Response: <STX><ACK><ETX><CR><LF>
	Directory of RAM
IPn	Read out the n program <i>name</i> of the program list from the RAM. If no program name exists, the response NAK is shown (ONLY PC). name (8 characters), number of lines (5 characters)

MINILOG

<i>Instruction</i>	<i>Meaning</i>
	Response: STX><ACK>name number of lines_line(s)<ETX><CR><LF> Response: <STX><NAK><ETX><CR><LF> no name available
	Information of the transmission protocol (effective from OMC/TMC system software V2.71)
ITR	Read out the state of the RS interface transmission protocol
ITSn	Define the RS interface transmission protocol n=0 Instruction transmission without checksum n=1 Instruction transmission with checksum
	Information operator panel
ITTSn	Define type of operator panel n=0 drive without operator panel n=1 drive with operator panel BT5 n=2 drive with operator panel TP11
	Version Request
IVR	The software version of the controller is read (ONLY PC). Response: <STX><ACK>Software Version<ETX><CR><LF>
	Memory Check
IZ	The free memory space for program lines in the EPROM is read (ONLY PC). Response: <STX><ACK>value ligne(s) free<ETX><CR><LF> value = Number of free program lines in the EPROM.

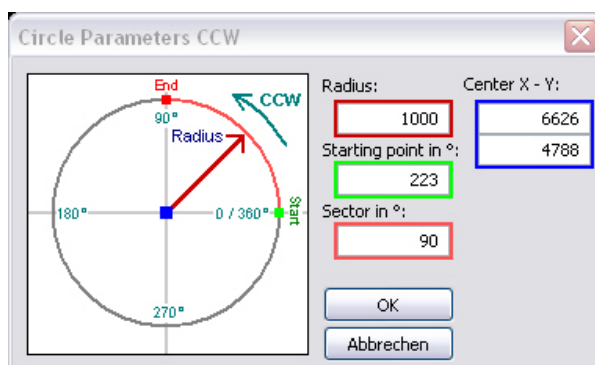
2.9 Interpolation Instructions

Linear interpolation

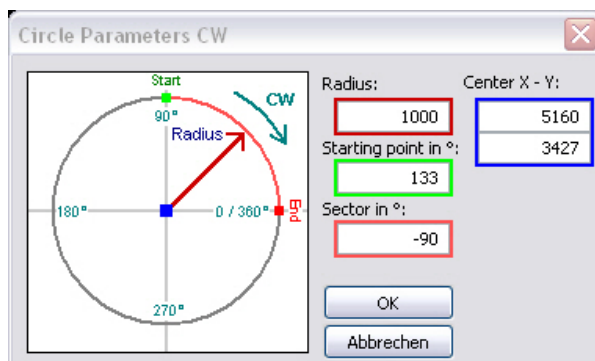
- IR** Linear interpolation is deactivated (reset).
- IS** Linear interpolation is activated (set).
 The instruction will be executed until the instruction IR is to be executed.
 Each pair of positioning instructions which appears in the same program line will be executed with a linear interpolation.

Response: <STX><ACK><ETX><CR><LF> (ONLY PC)

Circular interpolation



counterclockwise (CCW)



clockwise (CW)

- KRn** Set radius n of the circular arc,
 unit and factor of n are defined in P2 and P3 (see chap. 5)
- KS_n** Set starting point n on the circular arc in degree (°)
 n = 0 to 360°
- KW_n** Set the sector n in degree (°) from the starting point
 n = 0 to 360° (CCW)
 n = 0 to -360° (CW)

Important: Write all 3 instructions in **1** program line!

Example: KR100 KS90 KW180

2.10 Jump Instructions (ONLY PROG)

<i>Instruction</i>	<i>Meaning</i>
Relative Jump	
N+nn N–nn	Relative jump forward (+) or backward (–). Distance: nn program lines.
N+Rnn N–Rnn N+R[Rnn] N–R[Rnn]	Relative jump forward (+) or backward (–). Distance: number of program lines, defined by the content of register nn.
Absolute Jump	
Nnn	Absolute jump to program line number nn.
N*la*	Absolute jump. Destination program line number: marked by the label *la*.
NRnn NR[Rnn]	Absolute jump. Destination program line number: defined by the content of register nn.
NP[name]	Absolute jump to program <i>name</i> . Program starts at line number 1.
NP[name]Nnn	Absolute jump to program <i>name</i> . Program starts at line number nn.
NP[name]NRnn NP[name]NR[Rnn]	Absolute jump to program <i>name</i> . The program start line number is defined by the content of register nn.
NP[name]N*la*	Absolute jump to program <i>name</i> . The program start line is marked by the label *la*.
Conditional Jump Relative / E = Condition Fulfilled	
NE+nn NE–nn	Relative jump forward (+) or backward (–). Distance: nn program lines.
NE+Rnn NE–Rnn NE+R[Rnn] NE–R[Rnn]	Relative jump forward (+) or backward (–). Distance: number of program lines, defined by the content of register nn.
Conditional Jump Absolute / E = Condition Fulfilled	
NEnn	Absolute jump to program line number nn.
NE*la*	Absolute jump. Destination program line: marked by the label *la*.
NERnn NER[Rnn]	Absolute jump. Destination program line number: defined by the content of register nn.
NEP[name]	Absolute jump to program <i>name</i> . Program starts at line number 1.
NEP[name]Nnn	Absolute jump to program <i>name</i> . Program starts at line number nn.
NEP[name]NRnn NEP[name]NR[Rnn]	Absolute jump to program <i>name</i> . The program start line number is

<i>Instruction</i>	<i>Meaning</i>
	defined by the content of register nn.
NEP[name]N*la*	Absolute jump to program <i>name</i> . The program start line is marked by the label *la* .
Conditional Jump Relative / N = Condition Not Fulfilled	
NN+nn NN–nn	Relative jump forward (+) or backward (–). Distance: nn program lines.
NN+Rnn NN–Rnn NN+R[Rnn] NN–R[Rnn]	Relative jump forward (+) or backward (–). Distance: number of program lines, defined by the content of register nn.
Conditional Jump Absolute / N = Condition Not Fulfilled	
NNnn	Absolute jump to program line number nn.
NN*la*	Absolute jump. Destination program line number: marked by the label *la* .
NNRnn NNR[Rnn]	Absolute jump. Destination program line number: defined by the content of register nn.
NNP[name]	Absolute jump to program <i>name</i> . Program starts at line number 1.
NNP[name]Nnn	Absolute jump to program <i>name</i> . Program starts at line number nn.
NNP[name]NRnn NNP[name]NR[Rnn]	Absolute jump to program <i>name</i> . The program start line number is defined by the content of register nn.
NNP[name]N*la*	Absolute jump to program <i>name</i> . The program start line is marked by the label *la* .

2.11 Repeating of Program Lines

<i>Instruction</i>	<i>Meaning</i>
NWnn	The program line is nn times repeated.
NWRnn NWR[Rnn]	The program line is repeated as often as defined by the content of register nn.
	Response: <STX><ACK><ETX><CR><LF> (ONLY PC)

2.12 Ending or Interruption of a Program Call (ONLY PROG)

<i>Instruction</i>	<i>Meaning</i>
PE	The actual program is ended and the system waits for another changing of the REMOTE/LOCAL switch. If the program was started via computer, the system goes back to the COMPUTER MODE . Axis stop and start
PS	All axes are stopped and the running instruction is set by a flag.
PR	All axes are started, which are stopped by PS .

2.13 Program and Data Management (ONLY PC)

<i>Instruction</i>	<i>Meaning</i>
	Copy Program
QCP <i>name1 name2</i>	The program <i>name 1</i> is copied to a program named <i>name 2</i> .
	Delete Programs and Data
QDP <i>name</i>	The program <i>name</i> is deleted.
QDP *.*	All programs in the RAM are deleted
QDR	All registers in the RAM are set to zero.
	Write Program Line
QP <i>name</i> NnnS <i>instructions</i>	The instructions are written into the line <i>nn</i> of the program <i>name</i> . Maximal 32 characters are admitted.
	Read Program Line
QP <i>name</i> NnnR	The program line <i>nn</i> of the program <i>name</i> is read.
	Program Start by line
QP <i>name</i> NnnA	The program <i>name</i> is started from line <i>nn</i> .
	Program Stop
QPE	If the QPE Instruction is sent by the computer, it causes a jump back to the initial program level from which the program has been started.
	Change Program Name
QRP <i>name1 name2</i>	The program <i>name 1</i> is renamed <i>name 2</i> .

Instruction

Meaning

Program Transmission with Read Out

QPname Sznr

The program *name* is to be transmitted block wise. During the transmission, the whole control sequence must be observed. The name must have 8 characters and each line must contain 32 characters. The line number is not transmitted.

name → maximal 8 characters,

znr → number of lines to be transmitted

1. Computer:

<STX>QPname_Sznr<ETX><CR><LF>

The program transmission sequence is started.

2. Controller response:

STX<ACK>O<ETX><CR><LF>

If the program does not yet exist in the controller, and sufficient RAM space is available.

3. Computer:

<STX><ACK><ETB>name_program<EOT><ETX><CR><LF>

The program name and the program are transmitted without a separating blank.

4. Controller response:

<STX><ACK><ETX><CR><LF>

The program transmission was ended successfully.

5. Computer:

<STX>QPname_Sznr<ETX><CR><LF>

The program transmission sequence is started.

6. Controller response:

<STX><ACK>K<ETX><CR><LF>

If the program does not yet exist in the controller, but there is no sufficient RAM space available.

The transmission is finished.

7. Computer:

<STX>QPname_Sznr<ETX><CR><LF>

The program transmission sequence is started.

8. Controller response:

<STX><ACK>V<ETX><CR><LF>

If the program does already exist on the controller.

<i>Instruction</i>	<i>Meaning</i>
	<p>9. Computer:</p> <p><STX>J<ETX><CR><LF></p> <p>The controller program is to be replaced by the computer program.</p> <p>Continue with item 2.</p>
	<p>10. Computer:</p> <p><STX>QPname_Sznr<ETX><CR><LF></p> <p>The program transmission sequence is started.</p>
	<p>11. Controller response:</p> <p><STX><ACK>V<ETX><CR><LF></p> <p>If the program does already exist on the controller.</p>
	<p>12. Computer:</p> <p><STX>N<ETX><CR><LF></p> <p>The controller program is not to be replaced by the computer program.</p>
	<p>13. Controller response:</p> <p><STX><ACK><ETX><CR><LF></p> <p>The program transmission was finished.</p>
	<p>Read Program with Request</p>
QPname_R	<p>The program <i>name</i> is to be read from the controller unit. The name must have 8 characters and each line must contain 32 characters. The program name and the program are transmitted without a separating blank. The line number is not transmitted.</p>
	<p>Request and send</p>
	<p>1. Computer:</p> <p><STX>QPname_R<ETX><CR><LF></p> <p>The program <i>name</i> is to be read.</p>
	<p>2. Controller response:</p> <p><STX><ACK>Oznr<ETX><CR><LF></p> <p>If the program is available, the controller unit reports the character O and the number of the corresponding program lines.</p>
	<p>3. Computer:</p> <p><STX>J<ETX><CR><LF></p> <p>The computer sends a ready for transmission.</p>

Instruction	Meaning
4. Controller response:	<p><STX><ACK><ETB>name_Programm<EOT><ETX><CR><LF></p> <p>The program name and the program are transmitted continuously without a separating blank and no program line number is sent.</p> <p>The control character EOT marks the end of the program. ETX, CR and LF end the transmission.</p>
Request and breaking-off from the computer	
5. Computer:	<p><STX>QPname_R<ETX><CR><LF></p> <p>The program <i>name</i> is to be read.</p>
6. Controller response:	<p><STX><ACK>Oznr<ETX><CR><LF></p> <p>If the program is available, the controller unit reports the character O and the number of the corresponding program lines.</p>
7. Computer:	<p><STX>N<ETX><CR><LF></p> <p>The computer sends that the program should not be transmitted. The instruction sequence is finished.</p>
Request and breaking-off from the controller	
8. Computer:	<p><STX>QPname_R<ETX><CR><LF></p> <p>The program <i>name</i> is to be read.</p>
9. Controller Response:	<p><STX><ACK>K<ETX><CR><LF></p> <p>The program <i>name</i> does not exist. The instruction sequence is finished.</p>

2.14 Register Instructions

<i>Instruction</i>	<i>Meaning</i>
	Register Value Integer
Rnn.z	The digits after the decimal point of the register nn are deleted without truncation of the value. z = 0 – 6 digits after the decimal point
	Set Outputs with Register Value
RnnBA_{nn–mm} R[Rnn]BA_{nn–mm}	The content of the register nn is set as a binary value to the controller outputs nn to mm.
	Load Register with Input Status
RnnBE_{nn–mm} R[Rnn]BE_{nn–mm}	The status of the inputs nn to mm is written as a binary value into the register nn. Example: R1BE1–8 → Input status: 1010 0101 Result: 165
	Load Register with Hexadecimal Value
RnnBS_{value} R[Rnn]BS_{value}	The register nn is set to the value. The data are fed in hexadecimal Example: R1BS1FA The register 1 is set to the hexadecimal value 1FA. After the instruction was carried out, the content of the register 1 is 506 decimal.
	Shift Register Bit by Bit
RnnBL_m R[Rnn]BL_m	The content of the register nn is shifted the number of m digits to the left (MSB ←). The right side of the register is filled in with zero. m = 1 to 27 → maximal value of the register content. Example: R1S168 R1BL2 The register 1 is set to the decimal value 168, corresponding to the binary value 10101000 . After the register content was shifted the number of 2 digits to the left, the binary value is 1010100000 which corresponds to the decimal value 672. Response: <STX><ACK><ETX><CR><LF> (ONLY PC)

RnnBRm
R[Rnn]BRm

The content of the register nn is shifted the number of m digits to the right (→ LSB). The left side of the register is filled in with zero.

m = 1 to 27 → maximal value of the register content.

Example: R1S168 R1BR2

The register 1 is set to the decimal value 168, corresponding to the binary value **10101000**. After the register content was shifted (R1BL2) the number of 2 digits to the right, the binary value is **101010** which corresponds to the decimal value 42.

Register Bit Check

RnnBTm
R[Rnn]BTm

The content of the register nn is regarded as a binary value. The digit in the position m of the binary value is checked. If the corresponding bit has been set, the condition byte is set. Otherwise the condition byte is reset.

m = 0 to 27 → maximal value of the register content.

Example: R1S168 R1BT4

The register 1 is set to the decimal value 168, corresponding to the binary value **10101000**. The Instruction R1BT4 checks the 4th digit from the right side (m ← LSB) of the binary value. The condition byte is set, because the 4th digit has the value 1.

Response: <STX><ACK> E <ETX><CR><LF> or

<STX><ACK> N <ETX><CR><LF> (ONLY PC)

Logical Register Operations

Logic AND

RnnB^value
R[Rnn]B^value

A logical **AND** operation is carried out with the content of the register nn and the hexadecimal value. The condition byte is set if the result is zero. Otherwise it is reset.

Example: R1BS2A8 R1B^1A0

The register 1 is set to the hexadecimal value 2A8 (= 680 decimal). After the instruction **R1B^1A0** has been carried out, the content of the register 1 is 160 decimal.

	Decimal	Hex	Binary
	680	2A8	1010101000
	416	1A0	0110100000
Result:	160	0A0	0010100000

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

A logical **AND** operation is carried out with the content of the register nn and the content of register mm. The condition byte is set if the result is zero. Otherwise it is reset.

Logic OR

RnnBvvalue
R[Rnn]Bvvalue

A logical **OR** operation is carried out with the content of the register nn and the hexadecimal value. The condition byte is set if the result is zero. Otherwise it is reset.

Example: R1BS2A8 R1Bv1A0

The register 1 is set to the hexadecimal value 2A8 (= 680 decimal). After the instruction **R1Bv1A0** has been carried out, the content of the register 1 is 936 decimal.

	Decimal	Hex	Binary
	680	2A8	1010101000
	416	1A0	0110100000
Result:	936	3A8	1110101000

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

A logical **OR** operation is carried out with the content of the register nn and the content of register mm. The condition byte is set if the result is zero. Otherwise it is reset.

Response: <STX><ACK><ETX><CR><LF> (ONLY PC)

Logical Exclusive OR

RnnBXvalue
R[Rnn]BXvalue

A logical **XOR** operation is carried out with the content of the register nn and the hexadecimal value. The condition byte is set if the result is zero. Otherwise it is reset.

Example: R1BS2A8 R1BX1A0

The register 1 is set to the hexadecimal value 2A8 (= 680 decimal). After the instruction **R1BX1A0** has been carried out, the content of the register 1 is 776 decimal.

	Decimal	Hex	Binary
	680	2A8	1010101000
	416	1A0	0110100000
Result:	776	308	1100001000

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

A logical **XOR** operation is carried out with the content of the register nn and the content of register mm. The condition byte is set if the result is zero. Otherwise it is reset.

Compare Register Content with Number Values

Rnn=value
R[Rnn]=value

The content of register nn is compared with a number (value). The condition byte is set if equality has been detected. Otherwise it is reset.

Rnn#value
R[Rnn]#value

The content of register nn is compared with a number (value). The condition byte is set if inequality has been detected. Otherwise it is reset.

Rnn>value
R[Rnn]>value The content of register nn is compared with a number (value). The condition byte is set if the register value is higher. Otherwise it is reset.

Rnn<value
R[Rnn]<value The content of register nn is compared with a number (value). The condition byte is set if the register value is lower. Otherwise it is reset.

Compare Register Content with an Axis Parameter

Rnn=XPmm
R[Rnn]=XPmm The content of register nn is compared with the axis parameter mm of axis X. The condition byte is set if equality has been detected. Otherwise it is reset.

mm = parameter number

Rnn#XPmm
R[Rnn]#XPmm The content of register nn is compared with the axis parameter mm of axis X. The condition byte is set if inequality has been detected. Otherwise it is reset.

mm = parameter number

Rnn>XPmm
R[Rnn]>XPmm The content of register nn is compared with the axis parameter mm of axis X. The condition byte is set if the register value is higher. Otherwise it is reset.

mm = parameter number

Rnn<XPmm
R[Rnn]<XPmm The content of register nn is compared with the axis parameter mm of axis X. The condition byte is set if the register value is lower. Otherwise it is reset.

mm = parameter number

Compare Register Content

Rnn=Rmm
R[Rnn]=Rmm
Rnn=R[Rmm]
R[Rnn]=R[Rmm] The content of register nn is compared with the content of register mm. The condition byte is set if equality has been detected. Otherwise it is reset.

Rnn#Rmm
R[Rnn]#Rmm
Rnn#R[Rmm]
R[Rnn]#R[Rmm] The content of register nn is compared with the content of register mm. The condition byte is set if inequality has been detected. Otherwise it is reset.

Rnn>Rmm
R[Rnn]>Rmm
Rnn>R[Rmm]
R[Rnn]>R[Rmm] The content of register nn is compared with the content of register mm. The condition byte is set if the value of register nn is higher. Otherwise it is reset.

MINILOG

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

The content of register nn is compared with the content of register mm. The condition byte is set if the value of register nn is lower. Otherwise it is reset.

Response for all relations:

$\langle \text{STX} \rangle \langle \text{ACK} \rangle \text{ E } \langle \text{ETX} \rangle \langle \text{CR} \rangle \langle \text{LF} \rangle$ or
 $\langle \text{STX} \rangle \langle \text{ACK} \rangle \text{ N } \langle \text{ETX} \rangle \langle \text{CR} \rangle \langle \text{LF} \rangle$ (ONLY PC)

Arithmetical Register Operations

Addition

$R_{nn} + \text{value}$
 $R[R_{nn}] + \text{value}$

The value is added to the content of register nn.

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

The content of register mm is added to the content of register nn.

Subtraction

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

The value is subtracted from the content of the register nn.

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

The content of register mm is subtracted from the content of the register nn.

Multiplication

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

The content of register nn is multiplied by the value.

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

The content of the register nn is multiplied by the content of register mm.

Division

$R_{nn} : \text{value}$
 $R[R_{nn}] : \text{value}$

The content of register nn is divided by the value.

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

The content of register nn is divided by the content of register mm.

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

The content of register nn is divided by the value.

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

The content of register nn is divided by the content of register mm.

Trigonometric Functions

RnnSIN
RnnCOS
RnnTAN

Sinus, Cosinus or Tangant is evaluated from the value of the register nn and the result is written back to the register nn.

Square Root

RnnQW

The square root is evaluated from the value of the register nn and written back to the register nn.

Random Number

RnnRAND

The register nn is set with the random number in the range 0 to 32767.

Read Register

RnnR
R[Rnn]R

The content of register nn is read (ONLY PROG).
Response: <STX><ACK>value<ETX><CR><LF>

Response for all arithmetical operations:

<STX><ACK><ETX><CR><LF> (ONLY PC)

Write Register

with Decimal Values:

RnnSvalue
R[Rnn]Svalue

Register nn is set to the value.

with Register Values

RnnSRmm
R[Rnn]SRmm
RnnSR[Rmm]
R[Rnn]SR[mm]

Register nn is set to the value of register mm.

with Parameter Values

RnnSXPmm
R[Rnn]SXPmm

Register nn is set to the parameter mm of axis x.

with line number emergency stop

RnnSN
R[Rnn]SN

The register nn is set with the line number, in which an emergency stop started.

Example : **Lnr 001 FN10**
 Lnr 002 X+1000
 Lnr 003 X-1000 H N2
 Lnr 010 R1SN

In this example an emergency stop program is defined in line 10. The x-axis drives 1000 steps in + and – direction. In case of an emergency stop during an axis drive, the program continues in line 10 and the axis is stopped. With the instruction **R1SN** the register 1 is set with the line number, in which the emergency stop

started. Then, it is possible to interpret at which positioning the emergency stop started.

with the Timer Ticker Value

RnnSTT

The register nn is written with the timer ticker value.

with the Program Line Number

RnnSZ

The program line number at which this instruction is called is written into the register nn.

R[Rnn]SZ

The program line number at which this instruction is called is written into the register which is addressed by the register nn.

Example : **Lnr 041 R1S10**
 Lnr 042 R[R1]SZ

In this example the register 1 is written with the value 10. The register 10 is written with the instruction **R[R1]SZ** by the actual line number. The register content 10 is now 42. This instruction can be used for automatic start functions.

Write Register via Inputs

RnnSEmm-xx.k
R[Rnn]SEmm-xx.k

A BCD value is written via the inputs mm to xx into the register nn. k = number of digits after the decimal point.

Example: R1SE1-8.1

The inputs 1 to 8 have e.g. the status: **1001 0011**. The result is 9.3.

Change Register with Terminal

RnnST

Register nn is displayed in line 4 from position 10. New data are input at the cursor position. The register nn is rewritten by pressing the ENTER key of the terminal.

Example : **R41ST**

In this example the register 41 is displayed in the 4th line of the terminal display from the 10th position and is ready for editing.

Response: <STX><ACK><ETX><CR><LF> if terminal available
<STX><NAK><ETX><CR><LF> if no terminal available
 (ONLY PC)

RnnST.z

Register nn is displayed in line 4 with z digits after the decimal point (z=0 to 6). New data are input at the cursor position. The register nn is rewritten by pressing the ENTER key of the terminal.

Example : **R2ST.6**

The register 2 is displayed with 6 digits after the decimal point on the terminal and is rewritten.

RnnSTy

Register nn in line y is displayed (y=1 to 4) and is rewritten after new input with ENTER key.

Example : R2ST3

The register 2 is displayed in the 3rd line on the terminal from position 1 and is rewritten.

RnnSTy.z

Register nn in line y (y=1 to 4) is displayed with z digits after the decimal point (z=0 to 6) and is rewritten after new input with ENTER key.

Example : R2ST3.4

The register 2 is displayed in the 3rd line with 4 digits after the decimal point from position 1 on the terminal and is rewritten.

RnnSTy;m

Register nn in line y (y=1 to 4) is displayed from position m (m=1 to 20) and is rewritten after new input with ENTER key.

Example : R1ST3;6

The register 1 is displayed in the 3rd line from position 6 and is newly written.

RnnSTy;m.z

Register nn in line y (y=1 to 4) is displayed with z digits after the decimal point (z=0 to 6) from position m (m=1 to 20) and is rewritten after new input with ENTER key.

Example : R2ST2;2.6

The register 2 is displayed in the 2nd line from position 2 with 6 digits after the decimal point from position 1 on the terminal and is rewritten.

with A/D converter values (Option OMC/TMC)

**RnnSADxCy
R[Rnn]SADxCy**

Register nn is written by the A/D converter value.
x =0 or 1: A/D converter address,
y=1 to 4: A/D converter channel

Display Register with Terminal

RnnWy

The value of register nn is displayed in line y from position 1 (y=1 to 4).

Example : R2W2

Register 2 is displayed in line 2 from position 1.

RnnWy.z

The value of register nn is displayed in line y from position 1 (y=1 to 4) with z digits after the decimal point (z=0 to 6).

Example : R1W4.6

The register 1 is displayed in the 4th line from the 1st position with 6 digits after the decimal point.

RnnWy;m

The value of register nn is displayed in line y from position m (y=1 to 4, m= 1 to 20).

Example : R2W3;5

The register 2 is displayed in the 3rd line from the 5th position.

RnnWy;m.z The value of register nn is displayed in line y from position m (y=1 to 4, m= 1 to 20, z=0 to 6) with z digits after the decimal point.

Example : **R7W2;5;3**

The register 7 is displayed in the 2nd line from the 5th position with 3 digits after the decimal point.

2.15 System Status (ONLY PC)

Instruction *Meaning*

System Status General

S Axes check and request for the number of axes.

Response:<STX><ACK>n IO <ETX><CR><LF>

n = number of axes

System Status Binary

SB Read system status in binary format ($d_B = 0$ or 1).

Response: <STX><ACK> d_{B8} d_{B1} <ETX><CR><LF>

$d_B 1 = 1 \rightarrow$ Program Run

$d_B 2 = 1 \rightarrow$ Software Remote

$d_B 3 = 1 \rightarrow$ Emergency limit switch of an axis

$d_B 4 = 1 \rightarrow$ Power stage failure of an axis

$d_B 5 = 1 \rightarrow$ Error programming (reset after status request)

$d_B 6 = 1 \rightarrow$ Terminal is activated

$d_B 7 = 1 \rightarrow$ SRQ has been set

$d_B 8 = 1 \rightarrow$ Computer call

System Status Extended

SE Read system status in hexadecimal code. Two bytes (4 hexadecimal digits d_H) are available per axis: 1. + 2. byte for the x-axis, 3. + 4. byte for the y-axis.

Response:<STX><ACK> $d_{HX}d_{HX}d_{HX}d_{HX}d_{HY}d_{HY}d_{HY}d_{HY}$ <ETX><CR><LF>

Bit 0 = 1 \rightarrow Power stage overcurrent

Bit 1 = 1 \rightarrow Power stage under current

Bit 2 = 1 \rightarrow Power stage overtemperature

Bit 3 = 1 \rightarrow Power stage is activated

Bit 4 = 1 \rightarrow Initiator – is activated (emergency stop)

Bit 5 = 1 \rightarrow Initiator + is activated

Bit 6 = 1 \rightarrow Step failure (only with option SFI = Step Failure Indication)

Bit 7 = 1 \rightarrow Encoder error

Bit 8 = 1 \rightarrow Motor stands still

<i>Instruction</i>	<i>Meaning</i>
	Bit 9 = 1 → Reference point is driven and OK (is reset at stop by initiator) (Bit 10 to Bit 15 not reserved) If Bit 0 to Bit 2 are set at the same time, no power stage is connected. Otherwise, only one error is possible at the time.

Error Message Slave

SF1	Error message Slave is activated. In case of error in the Slave, the ERROR message is sent to the Master. All axis are stopped.
------------	---

SF0	Error message Slave is deactivated.
------------	-------------------------------------

SH	System Status Axes
-----------	---------------------------

Axis test with status axes output.

Response:<STX><ACK> E <ETX><CR><LF>, if all axes are stopped.

<STX><ACK> N <ETX><CR><LF>, if any axis is running.

System Status Decimal

ST	Read system status as decimal number.
-----------	---------------------------------------

Response: <STX><ACK>value <ETX><CR><LF>

value = number between 0 and 255

0 = End of program in the LOCAL MODE

1 = Program run

2 = Software Remote

4 = Emergency limit switch of an axis

8 = Power stage failure of an axis

16 = Error programming (reset after status request)

32 = Terminal is activated

64 = SRQ has been set

128 = Computer Mode

Initiators

SUI	Read status of initiators (limit switches).
------------	---

Response:<STX><ACK>I=n <ETX><CR><LF>

n = 0 → Axis is free, no initiator has reacted

n = + → Initiator + direction has reacted

n = - → Initiator - direction has reacted

n = 2 → Both initiators have reacted (that means: wrong polarity of the initiators, broken wire or no 24 V supply voltage)

Synchronous Start

S1	Prepare the synchronous start of the axes
-----------	---

S0	Execute the synchronous start of the axes.
-----------	--

2.16 Store Data into Flash EPROM

<i>Instruction</i>	<i>Meaning</i>
	Store programs and axis parameters (ONLY PC)
SA	Axis parameters are stored into the EPROM.
SP*.*	Programs are stored into the EPROM.

2.17 Time Loops

<i>Instruction</i>	<i>Meaning</i>
Tvalue	The value for time loops (value, content of register nn or register [Rnn]) is preset in ms.
TRnn	
TR[Rnn]	The program waits here until the preset time has run out. Response: <STX><ACK><ETX><CR><LF> (ONLY PC)
TTSvalue	The timer is loaded with a time (ms) value (value, content of register nn or register [Rnn]).
TTSRnn	
TTSR[Rnn]	The timer counts down to zero. The program is not interrupted. Response: <STX><ACK><ETX><CR><LF> (ONLY PC)
TT=0	The timer is compared with zero. If the timer is equal to zero the condition byte is set. Otherwise it is reset. Timer = 0: the preset time is passed.
TT>value	The timer is compared with the preset value (value, content of register nn or register [Rnn]). If the timer value is higher/lower than the preset value (condition fulfilled) the condition byte is set.
TT>Rnn	
TT>R[Rnn]	
TT<value	Otherwise it is reset.
TT<Rnn	
TT<R[Rnn]	
	Response: <STX><ACK>E<ETX><CR><LF> or <STX><ACK>N<ETX><CR><LF> (ONLY PC)

2.18 Subroutines (ONLY PROG)

<i>Instruction</i>	<i>Meaning</i>
	Break Off Subroutine
UA	Break off all subroutines and set stack. The program can be continued with a jump instruction.
	End of Subroutine
UE	The subroutine is finished and the program is continued at the program line where this subroutine has been called.
	Call of Subroutine
U_{nn}	The subroutine with the start line <i>nn</i> is called. The subroutine can be ended by means of the instruction UE.
UR_{nn} UR[R_{nn}]	The register <i>nn</i> or [R _{nn}] contains the start line of the called subroutine. The subroutine is ended with the instruction UE.
U*<i>la</i>*	The subroutine starts at that line which is indicated by the label * <i>la</i> *. The subroutine is ended by the instruction UE.
UP[<i>name</i>]	The subroutine <i>name</i> (start line number 1) is called. The subroutine is ended by the instruction UE.
UP[<i>name</i>]_{nn}	The subroutine <i>name</i> (start line number <i>nn</i>) is called. The subroutine is ended by the instruction UE.
UP[<i>name</i>]_{NR_{nn}} UP[<i>name</i>]_{NR[R_{nn}]}	The subroutine <i>name</i> starts at that program line which is stored in the register <i>nn</i> or [R _{nn}]. The subroutine is ended by the instruction UE.
UP[<i>name</i>]_{N*<i>la</i>*}	The subroutine <i>name</i> starts at that line which is indicated by the label * <i>la</i> *. The subroutine is ended by the instruction UE.
	Conditional Subroutine Call
	All instruction variants described above are available for the conditional subroutine call. The instruction call is only completed by the letter „E“ for condition fulfilled or „N“ for condition not fulfilled.
	"E" = Condition fulfilled
U_{enn}	see U_{nn} , above
UE_{R_{nn}} UE[R_{nn}]	
UE*<i>la</i>*	see U*<i>la</i>* , above
UEP[<i>name</i>]	see UP[<i>name</i>] , above

MINILOG

<i>Instruction</i>	<i>Meaning</i>
UEP[name]Nnn	
UEP[name]NRnn	
UEP[name]NR[Rnn]	
UEP[name]N*la*	see UP[name]N*la* , page 33
	"N" = Condition not fulfilled
Unnn	see Unn , page 33
UNRnn	
UNR[Rnn]	
UN*la*	see U*la* , page 33
UNP[name]	see UP[name] , page 33
UNP[name]Nnn	
UNP[name]NRnn	
UNP[name]NR[Rnn]	
UNP[name]N*la*	see UP[name]N*la* , page 33

2.19 Terminal Instructions (also by PC in case of terminal connection)

<i>Instruction</i>	<i>Meaning</i>
<>Wy	Erase text on line y (y=1 to 4)
<text>Wy	Display the text in line y from position 1 (y=1 to 4)
<text>Wy;m	Display the text in line y from position m (y=1 to 4; M=1 to 20)
	Response: <STX><ACK><ETX><CR><LF>

2.20 Axes Instructions

<i>Instruction</i>	<i>Meaning</i>
XC	Reset x-axis
YC	Reset y-axis
nC	Reset n-axis n=1 to 8
Response: <STX><ACK><ETX><CR><LF> (ONLY PC)	
Axis Status Request	
X=E	Axis request on power stage error.
X#E	Check (=) if a power stage error has occurred or check (#) if the power stage is operating normally. The error message "Failure" is requested.
X=H	Axis request on stillstand.
X#H	Check (=) if the axis is in standstill or check (#) if the axis is in motion. The condition byte is set when the condition is fulfilled. Otherwise it is reset.
X=I+	Axis request on initiator status.
X=I-	The condition byte is set when the axis has come to a standstill at the initiator or the initiator is not connected. Otherwise it is reset.
X=M	Axis request on power stage error.
X#M	Check power stage (=), if a Step failure has occurred or has not (#) occurred. The condition byte is set, when the condition is fulfilled. Otherwise it is reset. This instruction applies only to control units with optional Step Failure Indication (SFI) board.
X=N	Axis request on emergency stop.
X#N	Check (=) if the axis has come to a standstill (or not (#)) at an emergency switch. The condition byte is set when the condition is fulfilled. Otherwise it is reset.
Response: <STX><ACK>E<ETX><CR><LF> or <STX><ACK>N<ETX><CR><LF> (ONLY PC)	
Wait until Set Point is reached	
X>value	The axis X is positioned and the program waits until the value of the counter XP21 is higher than the preset value (value, content of
X>Rnn	register nn or register [Rnn]). If the XP21 value is higher or the axis
X>R[Rnn]	has come to a standstill the program is continued.
Example:	Inr 005 XP21S0 XP14S2000 XL+ Inr 006 X>5000 XP14S1000 Inr 007 X>10000 XS XP14S2000

<i>Instruction</i>	<i>Meaning</i>
	The axis is to be moved 10000 steps with 2000 Hz. After 5000 steps, the frequency is lowered to 1000 Hz and is set to 2000 Hz again after the standstill of the axis. At the instruction X>5000 the program is stopped and will be continued after the position 5000 is reached or the axis has been stopped by an emergency stop.
X<value X<Rnn X<R[Rnn]	The axis x is positioned and the program waits until the value of the counter xP21 is lower than the preset value (value, content of register nn or register [Rnn]). If the xP21 value is lower or the axis has come to a standstill the program is continued. Response: <STX><ACK><ETX><CR><LF> (ONLY PC), if the axis has come to a standstill or the position condition is fulfilled. Otherwise the programs waits.
	Switching Power Stages
	Activate
XMA	The power stage of axis X is activated.
	Deactivate
XMD	The power stage of axis X is deactivated.
	Axis parameter
XPmmR	The parameter mm of axis x is read out. (Only PROG) Response : <STX><ACK>value<ETX><CR><LF> mm = Parameter ID (ONLY PROG)
XPmmSvalue XPmmSRnn XPmmSR[Rnn]	The parameter mm of axis x is loaded with the preset value (value, the content of register nn or register [Rnn]). mm = Parameter ID
	Initialisation/Reference Search Run
	To initialize an axis, a reference search run has to be carried out. The initiators, also called limit switches, serve as reference point. The axis moves to an initiator. When the initiator signal is identified, the motor stops and moves as long in the opposite direction until there is no more initiator signal. In case of initiator offset setting the offset distance is run and the axis is stopped. This point is called MØP (mechanical zero point) or reference point.
X0-	The axis moves to the initiator of the - direction.
X0+	The axis moves to the initiator of the + direction.
X0-I	The axis moves in - direction and stops with the zero pulse of the incremental encoder. Only incremental, no SSI Encoder!

X0+I The axis moves in + direction and stops with the zero pulse of the incremental encoder. Only incremental, no SSI Encoder!

Response: <STX><ACK><ETX><CR><LF> (ONLY PC)

X0-^I The axis moves to the initiator of the – direction. After the offset distance the axis moves again until the zero impulse of the Incremental encoder stops the axis. Only incremental, no SSI Encoder!

X0 +^I The axis moves to the initiator of the + direction. After the offset distance the axis moves again until the zero impulse of the Incremental encoder stops the axis. Only incremental, no SSI Encoder!

Free Running

XLr The axis is started and runs as long as it is stopped by the instruction xS or by a limit switch.

r = + or – running direction

Relative Positioning

Xrvalue
XrRnn
XrR[Rnn] The axis runs the distance relatively which is preset by value, the content of register Rnn or register [Rnn].

r = + or – running direction

with stop instruction via input

XrvaluevEnnz
XrRnnvEnnz
XrR[Rnn]vEnnz The axis runs relatively with its creep speed the distance which is preset by value, the content of Rnn or register [Rnn]. The axis stops prematurely if the input nn gets the status z or a limit switch stops the positioning.

r = + or – running direction

z = S → input set

z = R → input reset

XrvaluevvEnnz
XrRnnvvEnnz
XrR[Rnn]vvEnnz The axis runs relatively with its high speed the distance which is preset by value, the content of Rnn or register [Rnn]. The axis stops prematurely if the input nn gets the status z or a limit switch stops the positioning.

r = + or – running direction

z = S → input set

z = R → input reset

Absolute Positioning Related to the MØP

XArvalue
XArRnn
XArR[Rnn] The axis runs, in relation to the mechanical zero point MØP (XP20) to the absolute position, which is preset by value, the content of Rnn or register [Rnn].

r = + or – running direction

with stop instruction via input

XArvaluevvEnnz
XArRnnvvEnnz
XArR[Rnn]vvEnnz The axis runs with high speed, in relation to the mechanical zero point MØP to the absolute position, which is preset by value, the content of Rnn or register [Rnn]. The axis stops prematurely if the input nn gets the status z or a limit switch stops axis run.

r = + or – running direction

z = S → input set

z = R → input reset

Absolute Positioning Related to the ELØP

XErvalue
XErRnn
XErR[Rnn] The axis runs, in relation to the electrical zero point (ELØP) to the absolute position, which is preset by value, the content of Rnn or register [Rnn].

r = + or – running direction

z = S → input set

z = R → input reset

with stop instruction via input

XErvaluevvEnnz
XErRnnvvEnnz
XErR[Rnn]vvEnnz The axis runs with high speed, in relation to the electrical zero point (ELØP) to the absolute position, which is preset by value, the content of Rnn or register [Rnn]. The axis stops prematurely if the input nn gets the status z or a limit switch stops axis run.

r = + or – running direction

z = S → input set

z = R → input reset

Axis Stop

XS All running instructions are cut off. The axis stops with the preset ramp.

XSN The axis stops with the preset emergency stop ramp (parameter P7).

2.21 Function Keys Read Out on Terminal BT24 (also by PC)

<i>Instruction</i>	<i>Meaning</i>
	Conditional Keyboard Read Out
#vFn	If the function key n is being depressed, the conditional byte is set. Otherwise it is reset. n = function key F1 to F6
#vnmx	If the key n or m or x is being depressed, the conditional byte is set. Otherwise it is reset. n, m, x = 0 to 9 (key 0 to 9) n, m, x = L (key CURSOR LEFT) n, m, x = R (key CURSOR RIGHT) n, m, x = U (key CURSOR UP) n, m, x = D (key CURSOR DOWN) n, m, x = H (key CURSOR HOME) n, m, x = B (key SCROLL) n, m, x = C (key CLEAR) n, m, x = E (key ENTER) n, m, x = P (key PRINT) n, m, x = ? (key ?) n, m, x = + (key +) n, m, x = - (key -) n, m, x = . (key .)

Example: ZNR 005 #vH1? NN-0

The BT24 keyboard is scanned as long as the key **H**, **1** or **?** is being depressed. The conditional byte is reset, if the keys **HOME**, **1** or **?** are not depressed. By the instruction **NN-0** the program jumps to the line hold of line 5.

Important: The key ENTER is not defined for a read out.

Response: <STX><ACK> E <ETX><CR><LF> or
<STX><ACK> N <ETX><CR><LF> (ONLY PC)

3 List of Minilog Instructions

#vFn.....	39	IPn.....	13
#vnmX.....	39	IR.....	15
<>Wy.....	34	IS.....	15
<text>Wy.....	34	ITR.....	14
ADnCxR.....	9	ITSn.....	14
ADnCxSy.....	9	ITTSn.....	14
AGnR.....	8	IVR.....	14
AGnSZZZZZZZ.....	8	IZ.....	14
Annnz.....	8	KRn.....	15
Annnzmmmzxxxz.....	8	KSn.....	15
ARnnnmmmxxx.....	8	KWn.....	15
CR.....	9	<text>Wy.....	34
CT.....	9	N*la*.....	16
D1.....	9	N+nn.....	16
D2.....	9	N+R[Rnn].....	16
D3.....	10	N+Rnn.....	16
E^nnzmmzxxz.....	10	nC.....	35
EBR.....	11	NE*la*.....	16
EBSx.....	11	NE+nn.....	16
EGnR.....	11	NE+R[Rnn].....	16
Ennz.....	11	NE+Rnn.....	16
Ennzmmz.....	11	NEnn.....	16
ERnnmmxx.....	11	NE-nn.....	16
ESR.....	11	NEP[name].....	16
ESSx.....	11	NEP[name]N*la*.....	17
Evnnzmmzxx.....	10	NEP[name]Nnn.....	16
FE.....	12	NEP[name]NR[Rnn].....	16
FN*la*.....	12	NEP[name]NRnn.....	16
FNznr.....	12	NER[Rnn].....	16
FP[name].....	12	NE-R[Rnn].....	16
H.....	12	NERnn.....	16
IAR.....	13	NE-Rnn.....	16
IBR.....	13	NN*la*.....	17
IBSname.....	13	NN+nn.....	17
ICnR.....	13	NN+R[Rnn].....	17
ICnSbaud.....	13	NN+Rnn.....	17
IESn.....	13	Nnn.....	16
IFL.....	13	N-nn.....	16
IFR.....	13	NNnn.....	17

NN –nn.....	17	R [Rnn]#value.....	24
NNP [name].....	17	R [Rnn]#XPmm.....	25
NNP [name] N *la*.....	17	R [Rnn]* R [Rmm].....	26
NNP [name] N nn.....	17	R [Rnn]*Rmm.....	26
NNP [name] NR [Rnn].....	17	R [Rnn]*value.....	26
NNP [name] NR nn.....	17	R [Rnn]/ R [Rmm].....	26
NNR [Rnn].....	17	R [Rnn]/Rmm.....	26
NN – R [Rnn].....	17	R [Rnn]/value.....	26
NN –Rnn.....	17	R [Rnn]+ R [Rmm].....	26
NNR nn.....	17	R [Rnn]+Rmm.....	26
NP [name].....	16	R [Rnn]+value.....	26
NP [name] N *la*.....	16	R [Rnn]< R [Rmm].....	26
NP [name] N nn.....	16	R [Rnn]<Rmm.....	26
NP [name] NR [Rnn].....	16	R [Rnn]<value.....	25
NP [name] NR nn.....	16	R [Rnn]<XPmm.....	25
NR [Rnn].....	16	R [Rnn]= R [Rmm].....	25
N – R [Rnn].....	16	R [Rnn]=Rmm.....	25
NR nn.....	16	R [Rnn]=value.....	24
N –Rnn.....	16	R [Rnn]=XPmm.....	25
NW nn.....	17	R [Rnn]> R [Rmm].....	25
NWR [Rnn].....	17	R [Rnn]>Rmm.....	25
NWR nn.....	17	R [Rnn]>value.....	25
PE	18	R [Rnn]>XPmm.....	25
PR	18	R [Rnn] B [^] R [Rmm].....	23
PS	18	R [Rnn] B [^] Rmm.....	23
QCP name1 name2.....	18	R [Rnn] B [^] value.....	23
QDP *.*.....	18	R [Rnn] BA nn–mm.....	22
QDP name.....	18	R [Rnn] BE nn–mm.....	22
QDR	18	R [Rnn] BL m.....	22
QPE	18	R [Rnn] BR m.....	23
QP name N nn A	18	R [Rnn] BS value.....	22
QP name N nn R	18	R [Rnn] BT m.....	23
QP name N nn S instructions.....	18	R [Rnn] Bv R [Rmm].....	24
QP name S znr.....	19	R [Rnn] Bv Rmm.....	24
QP name_ R	20	R [Rnn] Bv value.....	24
QRP name1 name2.....	18	R [Rnn] BXR [Rmm].....	24
R [Rnn]: R [Rmm].....	26	R [Rnn] BXR mm.....	24
R [Rnn]# R [Rmm].....	25	R [Rnn] BX value.....	24
R [Rnn]:Rmm.....	26	R [Rnn] R	27
R [Rnn]#Rmm.....	25	R [Rnn]– R [Rmm].....	26
R [Rnn]:value.....	26	R [Rnn]–Rmm.....	26

MINILOG

R[Rnn]SADxCy	29	RnnBA ⁿⁿ -mm	22
R[Rnn]SE ^{mm} -xx.k	28	RnnBE ⁿⁿ -mm	22
R[Rnn]SN	27	RnnBL ^m	22
R[Rnn]SR[mm].....	27	RnnBR ^m	23
R[Rnn]SR ^{mm}	27	RnnBS ^{value}	22
R[Rnn]S ^{value}	27	RnnBT ^m	23
R[Rnn]SXP ^{mm}	27	RnnBvR[R ^{mm}]	24
R[Rnn]SZ.....	28	RnnBvR ^{mm}	24
R[Rnn]-value.....	26	RnnBv ^{value}	24
Rnn:R[R ^{mm}]	26	RnnBXR[R ^{mm}].....	24
Rnn#R[R ^{mm}].....	25	RnnBXR ^{mm}	24
Rnn:R ^{mm}	26	RnnBX ^{value}	24
Rnn#R ^{mm}	25	RnnCOS.....	27
Rnn:value	26	RnnQW	27
Rnn#value.....	24	RnnR.....	27
Rnn#XP ^{mm}	25	Rnn-R[R ^{mm}].....	26
Rnn*R[R ^{mm}]	26	RnnRAND	27
Rnn*R ^{mm}	26	Rnn-R ^{mm}	26
Rnn*value	26	RnnSADxCy.....	29
Rnn.z.....	22	RnnSE ^{mm} -xx.k	28
Rnn/R[R ^{mm}].....	26	RnnSIN	27
Rnn/R ^{mm}	26	RnnSN	27
Rnn/value.....	26	RnnSR[R ^{mm}]	27
Rnn+R[R ^{mm}].....	26	RnnSR ^{mm}	27
Rnn+R ^{mm}	26	RnnST	28
Rnn+value.....	26	RnnST.z	28
Rnn<R[R ^{mm}].....	26	RnnSTT	28
Rnn<R ^{mm}	26	RnnSTy	28
Rnn<value.....	25	RnnSTy.m	29
Rnn<XP ^{mm}	25	RnnSTy.m.z	29
Rnn=R[R ^{mm}].....	25	RnnSTy.z	29
Rnn=R ^{mm}	25	RnnS ^{value}	27
Rnn=value.....	24	RnnSXP ^{mm}	27
Rnn=XP ^{mm}	25	RnnSZ.....	28
Rnn>R[R ^{mm}].....	25	RnnTAN	27
Rnn>R ^{mm}	25	Rnn-value.....	26
Rnn>value.....	25	RnnWy	29
Rnn>XP ^{mm}	25	RnnWy.m	29
RnnB^R[R ^{mm}].....	23	RnnWy.m.z	30
RnnB^R ^{mm}	23	RnnWy.z	29
RnnB^value.....	23	S.....	30

S0	31	UNP [name] Nnn	34
S1	31	UNP [name] NR [Rnn].....	34
SA	32	UNP [name] NRnn	34
SB	30	UNR [Rnn]	34
SE	30	UNRnn	34
SF0	31	UP [name].....	33, 34
SF1	31	UP [name] N *la.....	33, 34
SH	31	UP [name] Nnn	33
SP *,*	32	UP [name] NR [Rnn]	33
ST	31	UP [name] NRnn	33
SUI	31	UR [Rnn]	33
TR [Rnn]	32	URnn	33
TRnn	32	X#E	35
TT < R [Rnn].....	32	X#M	35
TT < Rnn	32	X#N	35
TT <value.....	32	X < R [Rnn]	36
TT =0	32	X < Rnn	36
TT > R [Rnn].....	32	X <value.....	36
TT > Rnn	32	X = I -.....	35
TT >value.....	32	X = I +	35
TTSR [Rnn]	32	X = E	35
TTSRnn	32	X = H	35
TTS value	32	X = M	35
T value	32	X = N	35
U *la	33, 34	X > R [Rnn]	35
UA	33	X > Rnn	35
UE	33	X >value	35
UE *la*	33	X0 -.....	36
UE [Rnn].....	33	X0 - ^I	37
UEnn	33	X0 +.....	36
UEP [name]	33	X0 + ^I	37
UEP [name] N *la*.....	34	X0 + I	37
UEP [name] Nnn	34	X0 - I	36
UEP [name] NR [Rnn].....	34	XAr [Rnn]	38
UEP [name] NRnn	34	XAr [Rnn] vvEnnz	38, 45, 46
UERnn	33	XArnn	38
UN *la*	34	XArnnvvEnnz	38
Unn	33	XAr value	38
UNnn	34	XAr value vvEnnz	38, 45, 46
UNP [name].....	34	XC	35
UNP [name] N *la*.....	34	XEr [Rnn]	38

MINILOG

XErR[Rnn]vvEnnz	38	XrR[Rnn]	37
XErRnn	38	XrR[Rnn]vEnnz	37
XErRnnvvEnnz	38	XrR[Rnn]vvEnnz	37
XErvalue	38	XrRnn	37
XErvaluevvEnnz	38	XrRnnvEnnz	37
XLr	37	XrRnnvvEnnz	37
XMA	36	Xrvalue	37
XMD	36	XrvaluevEnnz	37
XPmmR	36	XrvaluevvEnnz	37
XPmmSR[Rnn]	36	XS	38
XPmmSRnn	36	XSN	38
XPmmSvalue	36		

4 List of DIN Instructions

The controller program can also be defined by the DIN instructions for process conditions and special functions. These standard instructions DIN 66025 can be used in one program with all the MINILOG instructions.

<i>Instruction</i>	<i>Meaning</i>
	G Instructions (Process Conditions)
G00, G0	Coordinate setting course Positioning with the speed as high as possible (high speed activation) without interpolation by parameter 14.
G01, G1	Set the linear interpolation
G04Tnn, G4Tnn	Program interrupts with term, programmed or defined in the controller The program continues automatically. n= in seconds with digits after the decimal point Abortion via input 2
G05, G5	Halt: the program waits for standstill of all axes and after that is continued
G20Lnn	Unconditional jump to line nn
G20L+nn	Unconditional jump by nn lines in + direction
G20L-nn	Unconditional jump by nn lines in – direction
G20*label*	Unconditional jump to *label*
G20L*label*	Unconditional jump to *label*
G20LP[name]	Unconditional jump to program name in line 1
G21zLnn	Conditional jump to line nn z=E or N
G21zL+nn	Conditional jump by nn lines in + direction z = E or N
G21zL-nn	Conditional jump by nn lines in – direction z = E or N
G21z*label*	Conditional jump to label z = E oder N
G21zL*label*	Conditional jump to label z = E or N
G21zLP[name]	Conditional jump to program name in line 1 z = E or N
G22Lnn	Call the subroutine program nn Subroutine is marked by G98Lnn in the program
G22*label*	Call the subroutine program *label*
G22P[name]	Call the subroutine program [name]

MINILOG

G23Lnn	Stop the subroutine at once and return to line nn
G23*label*	Stop the subroutine at once and return to *label*
G74	Initialisation of all axes – direction
G74 x	Initialisation of one axis x= X or Y
G79Lnn	Automatic subroutine call at the end of the program line Subroutine is marked by G98Lxx in the program
G80	End of the automatic subroutine call G79
G90	Positioning absolut value in relation to the reference counter parameter 20
G91	Incremental positioning
G92	Set the absolute counter (zero offset) parameter 20
G98Lnn	Subroutine beginning and declaration nn Subroutine name maximum 6 characters
G99	Subroutine end
	M instructions (Additional Functions)
M00, M0	Programmed halt The program is continued by setting input 2
M01, M1	Programmed halt, if input 3 is ON The program is continued by setting input 2
M02, M2	End of program
M03, M3	Spindle ACTIVATED, clockwise rotation output 1 on; output 2 off
M04, M4	Spindle ACTIVATED, counterclockwise rotation output 1 off; output 2 on
M05, M5	Spindle quick STOP output 1 off; output 2 off
M07, M7	Cooling 2 on output 3 off; output 4 on
M08, M8	Cooling 1 on output 3 on; output 4 off
M09, M9	Cooling off output 3 off; output 4 off
M10	Tool holder on; output 5 on
M11	Tool holder off; output 5 off
M68	Clamp component; output 6 on
M69	Unclamp component ; output 6 off

5 Parameters

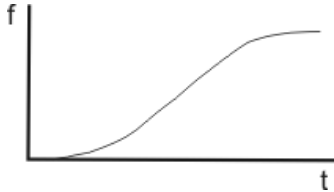
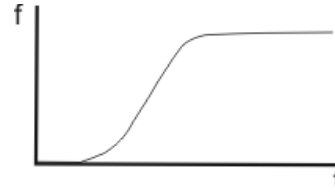
List of Parameters		
No.	Meaning	Default
P01	Type of movement 0 rotational Rotating table, 1 limit switch for mechanical zero (referencing) 1 linear, for XY tables or other linear systems, 2 limit switches: 1. Mechanical zero and limit direction – 2. Limit direction +	0
P02	Measuring units of movement 1 step 2 mm 3 inch 4 degree	1
P03	Conversion factor for the thread 1 step corresponds to ... Computing the conversion factor: $\text{Conversion factor} = \frac{\text{Thread}}{\text{Number of steps per revolution}}$ <u>Example:</u> 4 mm thread pitch 200-step motor = 400 steps/rev. in the half step mode $\text{Conversion factor} = \frac{4}{400} = 0.01$	1
P04	Start/stop frequency The start/stop frequency is the maximum frequency to start or stop the motor without ramp. At higher frequencies, step losses or motor stop would be the result of a start or stop without ramp. The start/stop frequency depends on various factors: type of motor, load, mechanical system, power stage. The frequency is programmed in Hz.	400
P07	Emergency stop ramp The frequency is programmed in Hz/s.	50 000
P08	f_{\max} MØP (mechanical zero point) Run frequency during initializing (referencing) Enter in Hz	4000

MINILOG

List of Parameters		
P09	Ramp MØP Ramp during initializing, associated to parameter P08 Enter in Hz/s	25 000
P10	f_{\min} MØP Run frequency for leaving the limit switch range Enter in Hz	400
P11	MØP offset for limit switch direction + Distance between reference point MØP and limit switch activation Unit: is defined in parameter P02	0
P12	MØP offset for limit switch direction – Distance between reference point MØP and limit switch activation Unit: is defined in parameter P02	0
P13	Recovery time MØP Time lapse during initialization Enter in ms	20
P14	f_{\max} Run frequency during program operation Enter in Hz (integer value) see chap. 6	4000
P15	Ramp for run frequency (P14) Input in Hz/s (integer value)	25 000
P16	Recovery time Position Time lapse after positioning Input in ms	20
P17	Boost (defined in P42) 0 off 1 on during motor run 2 on during acceleration and deceleration ramp <u>Remarks:</u> 1. The boost current can be set in parameter P42. 2. With parameter P17 you can select in which situations the controller switches to boost current. 3. P17 = 1 means, the boost current always is switched on during motor run. During motor standstill the controller switches to stop current.	0

List of Parameters		
The following four parameters are counters normally set by the program. In the MC-COMM parameter editor these parameters are not displayed!		
P19	Electronical zero counter Used for setting operating points. At standstill of the axis, P19 can be read or programmed during program execution.	
P20	Mechanical zero counter This counter contains the number of steps referred to the mechanical zero (MØP). Can be read at axis standstill. If the axis reaches the MØP, P20 will be set to zero.	
P21	Absolute counter Indicates the true position. At any time P21 can be asked, programmed or modified. P21 is not automatically set to zero when the MØP is reached.	
P22	Encoder counter Indicates the true encoder position.	
Parameters 26 to 45 are specially adapted to the OMC and TMC controllers:		
P26	Divider for SSI encoder Data transfer rate 10 to 80 (= 100 to 800 kHz)	0
P27	Proximity switch type 0 = + and – are PNP normally closed contacts (NCC) 1 = + is a normally open contact (NOC), - is a NCC 2 = + is a NCC, - is a NOC 3 = + and – are PNP NOC	0
P28	Output motor brake Define the output number for the motor brake Input for OMC: 1 - 8 for TMC: 1 - 16 Example: output number = 4 The output 4 is set in case of an error in the power stage.	0
P29	Delay time for enabling motor brake The delay time after switch-on for releasing the brake Input in sec	0

MINILOG

List of Parameters		
P30	<p>Frequency band setting</p> <p>0 = manual 1 = automatic</p> <p>Remark: See chap. 6 It is recommended to work with the automatic setting mode. For each run frequency (P14) and ramp (P15) the controller automatically selects suitable settings.</p>	1
P31	<p>Frequency and ramp predivider (only if P30 = 0, manual)</p> <p>See chap. 6</p> <p>Predivider values: 3 or 5 (OMC: 5, TMC: 3) This parameter can be used for individual settings for special applications.</p>	3
P32	<p>Positioning ramp shape</p> <p>0 = s-shape 1 = linear ramp</p> <p>Remark: The s-shape ramp can be modified with P33 parameter.</p>	1
P33	<p>Arc value setting for s-shape ramp</p> <p>Values: OMC: 1 to 8191 TMC: 1 to 32767</p> <div style="display: flex; justify-content: space-around; align-items: flex-end;"> <div style="text-align: center;">  <p>P33: low value</p> </div> <div style="text-align: center;">  <p>P33: high value</p> </div> </div>	1
P34	<p>Encoder type</p> <p>-1 = no encoder 0 = incremental 5.0 V 1 = incremental 5.5 V 2 = serial interface SSI binary Code 5.0 V 3 = serial interface SSI binary Code 5.5 V 4 = serial interface SSI Gray Code 5.0 V 5 = serial interface SSI Gray Code 5.5 V</p>	0
P35	<p>Encoder resolution for SSI encoder</p> <p>Enter max. encoder resolution in Bit (max. 32 Bit)</p>	10
P36	<p>Encoder function</p> <p>0 = counter 1 = SFI dynamical step failure indication</p>	0

List of Parameters		
P37	Encoder tolerance for SFI Enter tolerance value for SFI evaluation	0
P38	Encoder preferential direction of rotation 0 = + (positive) 1 = - (negative)	0
P39	Encoder conversion factor 1 increment corresponds to ...	1
P40	Stop current in 0.1 A steps Values: 0 to 6.3 A Input: 0 to 63	2
P41	Run current in 0.1 A steps Values: 0 to 6.3 A Input: 0 to 63	4
P42	Boost current as absolute value in 0.1 A steps which is not added to the run current. Values: 0 to 6.3 A Input: 0 to 63	4
P43	Current delay time in ms	20
P44	Control pulses change over 0 = power stage X (Y) on controller X (Y) 1 = power stage X (Y) on controller Y (X) 2 = external control pulses on power stage X (Y)	0
P45	Step resolution 0 = Full step 3 = 1/5 step 1 = Half step 4 = 1/10 step 2 = 1/4 step 5 = 1/20 step 6 = 1/8 step	4

6 About Parameters 30 and 31

The motor controller is parameterized by several chip internal values as frequency and predivider. In the automatic mode (P30 = 1) these two parameters are automatically derived by software from the physical units indicated by the MINILOG parameters P14 (run frequency) and P15 (ramp). Due to the binary chip design, only certain combinations of frequencies and ramps can be used.

Controller	Predivider	f_{\max} [Hz]	$f_{\text{resolution}}$ [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

Predivider/frequencies/resolution

Predivider: Defines a frequency range.

f_{\max} : Maximum frequency in this range

$f_{\text{resolution}}$: Minimum distance between two accessible frequencies

Remark: Frequency and destination position can be changed during the run!

Snap regula for manual frequency band setting:

Little frequency – very exactly setting:

Select a low predivider value.

In this case the ramp will be flat. Steep ramps are not possible with a low predivider value.

Steep acceleration ramp: (nearly rectangle profile)

Select a high predivider.

In this case you'll get a coarse frequency grid.

7 Program Example A/D Converter

Program	Comment
START	
R1SAD0C0	Register 1 is set with AD card 0 Ch 0
R2SAD0C1	Register 2 is set with AD card 0 Ch 1
R3SAD0C2	Register 3 is set with AD card 0 Ch 2
R4SAD0C3	Register 4 is set with AD card 0 Ch 3
R1W1	The value of register 1 is displayed in line 1
R2W2	The value of register 2 is displayed in line 2
R3W3	The value of register 3 is displayed in line 3
R4W4	The value of register 4 is displayed in line 4
N*START*	Jump back to Start

8 Index

A

A/D converter 9, 29, 53
Addressing mode
 direct 5
 indirect 5
 with label 5
Adressing
 with label 5
Axis Instructions
 Free Running 37
 Initialization 36
 Power stages 35
 Read/load parameter 36
 Status request 35
 Stop 18, 38
 Wait 35

B

Baudrate
 read 13
 set 13
Broadcast 7

C

Checksum 7
Circular interpolation 15
Counter 49

D

DIN instructions 45
Display instruction 34

E

ELØP 38
Error message Slave 31

F

Frequency band setting 49, 50

I

Inputs
 Conditional link 11
 Logical AND 10
 Read status 11
Instruction code 4

J

Jump instructions
MA 1174-A009 GB

conditional 17
relative 16

L

Label 5
Linear interpolation 15

M

Megacard
 Memory check 14
MØP 37

O

Outputs
 read 8
 Reading 8
 set 8

P

Parameter list 47
Positioning
 absolute 38
 in relation to ELØP 38
 in relation to MØP 38
 relative 37
Positioning ramp shape 50
Predivider 52
Predivisor 50
Process conditions 45
Program
 Start 11
 Stop 11

Program and data management
 copy 18
 delete 18
 read program 20

Program Call
 Ending 18

Programname 5

R

RAM
 Contents read 13
Reference search run 36
Register
 Shifting 22
Register instructions
 Arithmetical operations

Cosinus 27
Random number 27
Sinus 27
Square root 27
Tangent 27

write with A/D converter values 29
write with decimal value 27
write with line number 28
write with line number emergency stop 27

Reset Controller 9

S

Send telegram 7

Slave 31

S-shape ramp 50

Standard functions 45

Start-/Stop frequency 47

Subroutines

Break-off 33

Call 33

conditional call 33

End 33

Synchronous start 31

System Status (only computer mode)

decimal 31

T

Time loops 32

V

Version request 14

W

Write instruction via serial interface 9

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