Programming interface
for series 300 controllers

OED3

Version 1.0

Doc. no. 212.954/DGB 04.94

Ident. no.: 00441109480

Edition: a000 April 94

Table of contents

Table of contents

Table of contents

1 General description

The OED3 programming interface is utilized to create simple application programs for BERGER LAHR series 300 controllers (e.g. WDP5-318) using a terminal or PC.

Using the OED3 programming interface it is possible to perform the following functions:

– controller parameterizing and programming
– line-by-line execution of an application program (step mode)
– automatic execution of an application program (automatic mode)
– simple programming of manual and teach-in mode using pre-defined flags.

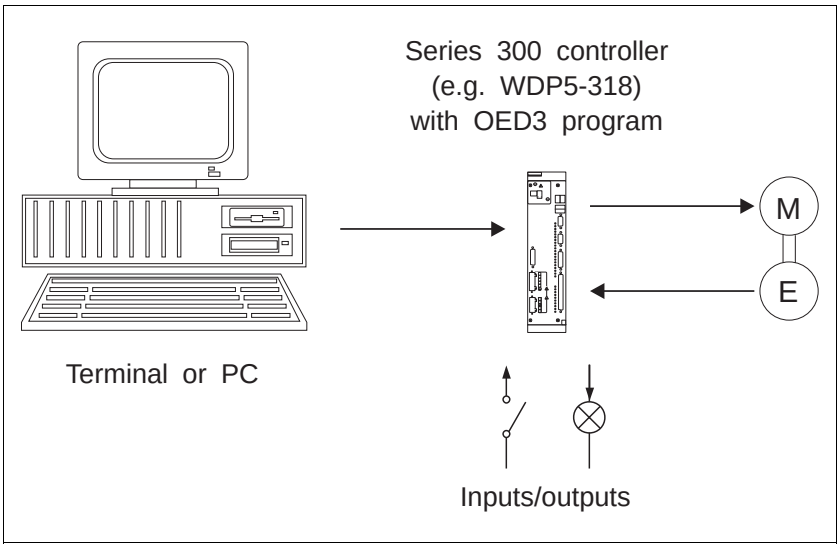


Fig. 1-1 System environment

1.1 Reference documentation

This documentation includes all information necessary for installation and operation of the OED3 programming interface.

The controller manual comprises controller-specific information.

1.2 Contents of software package

The OED3 software package includes the following files:

– Sample programs
– README file
– EXE file for offline support
– BTERM terminal program

1.3 Purpose

The OED3 programming interface is utilized to create application programs for BERGER LAHR series 300 controllers (e.g. WDP5-318) using a terminal or PC.

1.4 Programming and operating a controller

The OED3 program is installed in the BERGER LAHR series 300 controller.

In edit mode, the controller can be programmed using a terminal or PC (with terminal program, e.g. BTERM), see chapter 3.

Application program

The application program created in edit mode is executed in the controller automatic mode after leaving edit mode.
An application program is made up of a sequence program and a PLC program which are executed in parallel (according to time-slice principle) in the controller.

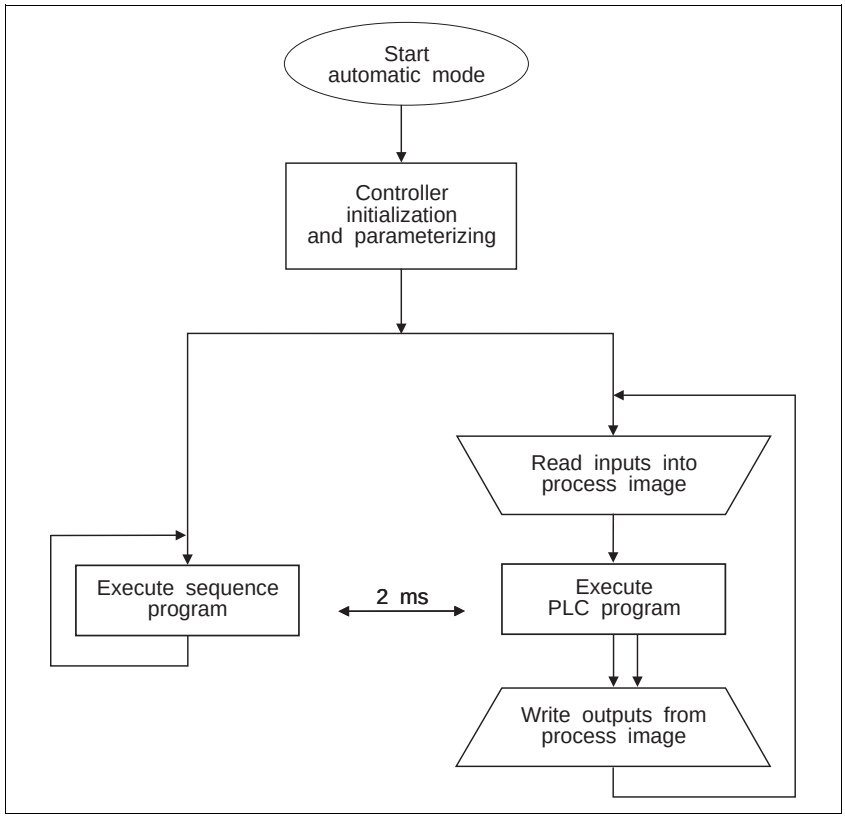


Fig. 1-2 Program execution in automatic mode

PLC program

The following functions can be executed with the PLC program:

- logical operations (AND, OR)
- arithmetic operations (addition, subtraction, multiplication, division)
- transfer operations (loading, saving)
- relational operations (equal, greater than, less than)
- conditional and unconditional jump operations.

The PLC program is cyclically executed when the controller is in automatic mode. The input states are read into the process image (intermediate storage for inputs/outputs) after starting automatic mode. The commands entered in the PLC program are then executed and the output states of the process image updated. Finally, the output states of the process image are output and the cycle starts again (the cycle time is not monitored).

Sequence program

The sequence program offers the following functions:

- axis operating mode setting (point-to-point mode, position following mode)
- execution of positioning operations (absolute or relative) and reference movements
- transmitting and receiving characters and variables through the serial interface
- synchronization with master controller (through synchronizing input/output).

In addition, most of the PLC program functions are available. However, the inputs/outputs are accessed directly, not through the PLC program process image.

NOTE
Communication between PLC and sequence program can be effected using flags or variables.

Teach-in

Positions which have been programmed in the application program can be approached and overwritten in teach-in mode.

Text editor

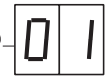
The text editor is used to input character strings which have to be transmitted in the sequence program.

Parameter editor

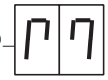
The parameter editor is used to input different parameters for the sequence program.

1.5 Operating modes of a series 300 controller with OED3

The controller standard version has two operating modes:

40 

– Application mode for executing an application program

40 

– Manual mode for manually moving the motor by pressing the respective keys on the front panel.

NOTE
The controller application mode and manual mode are described in the controller manual.

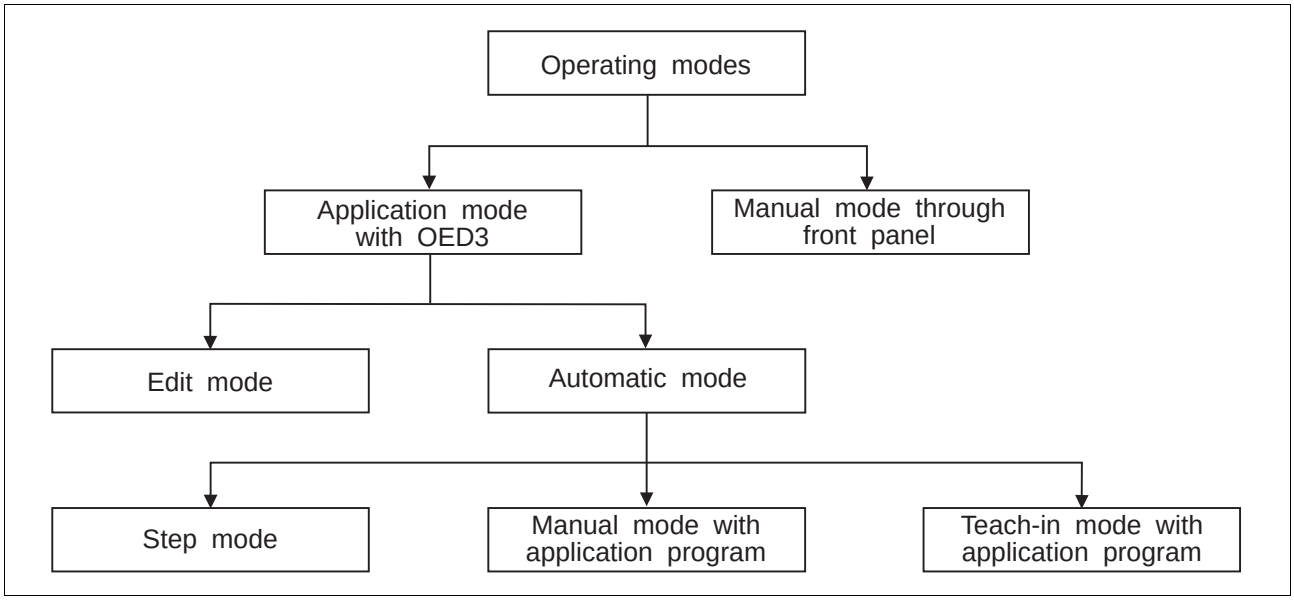
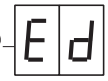


Fig. 1-3 Controller operating modes

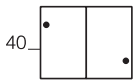
The following operating modes can be selected in application mode.

Edit mode

40 

Edit mode is activated by pressing selector switch 41 on + side.
In edit mode, an application program can be entered and displayed line by line using a terminal or PC. If a PC (with terminal program, e.g. BTERM) is used as the programming device, a complete application program can be loaded from the controller into the PC (upload) or from the PC into the controller (download); see chapter 3.

Automatic mode



The application program, which is made up of PLC program and sequence program, is executed in automatic mode. Automatic mode is activated after leaving edit mode (with <X> and <↵>). If programmed in the application program, the manual, teach-in, or step mode can be selected in automatic mode; see chapter 3.

NOTE
In controller automatic mode, it is possible to enable or disable test output using a flag. During test output, the sequence program is displayed line by line on the screen.

The following table lists the predefined flags.

Flag	Meaning			
m0	Manual movement clockwise			
m1	Manual movement counterclockwise			
m2	Rapid speed			
m3 m4	Multi-axis selection bit 0 Multi-axis selection bit 1	Axis	m4	m3
		1	0	0
		2	0	1
		3	1	0
		4	1	1
m5	Activate teach-in			
m6	Store position			
m7	Do not store position			
m8	Reserved			
m9	Test output; 0 = Off, 1 = On			
m10	Activate manual mode; 0 = Inactive, 1 = Active			
m11	Manual movement active? 0 = No, 1 = Yes			
m12	Test output; 0 = Line number, 1 = Complete line			
m13 to m20	Reserved			
m21 to m1000 m32 to m111 m112 to m191	Freely available; for controller with Interbus-S interface: For extended inputs For extended outputs			
m1001 to m1023	System constants			

NOTE
The separate OED3 software package includes sample programs for manual and teach-in mode.

– Manual mode

  In manual mode, the shaft can be manually moved at slow or rapid speed. For multi-axis systems, the shaft must be selected prior to the manual movement. Initiating a manual movement or selecting a shaft (for multi-axis systems) is effected in the application program by predefined flags; see figure 1-4.

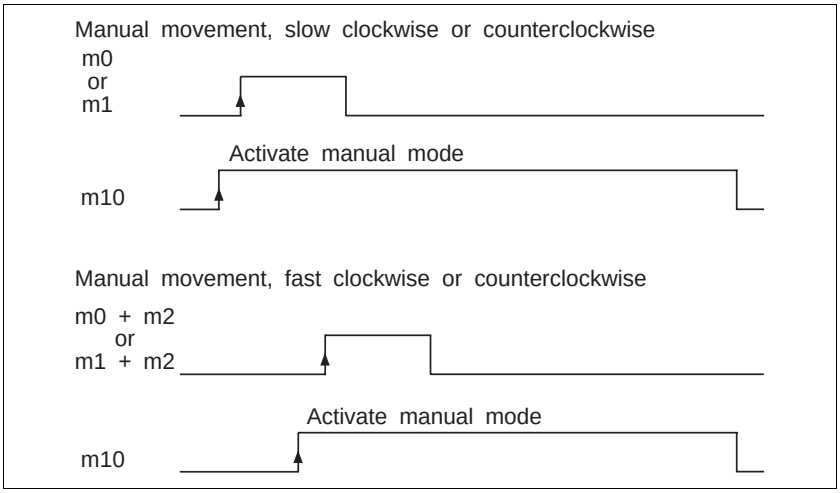
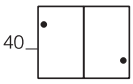


Fig. 1-4 Timing diagram for manual mode

– Teach-in mode



In teach-in mode, positions can be manually approached and stored in the sequence program. For this purpose, the sequence program is normally processed until a positioning command has been executed. Thereafter, the sequence program stops and the programmed position can be changed through teach-in. Thus, it is possible to overwrite the programmed positions one after the other by the teach-in positions. Initiating teach-in mode, or storing or ignoring positions, is performed in the application program by predefined flags; see figure 1-5.
Teach-in is only possible in point-to-point mode.

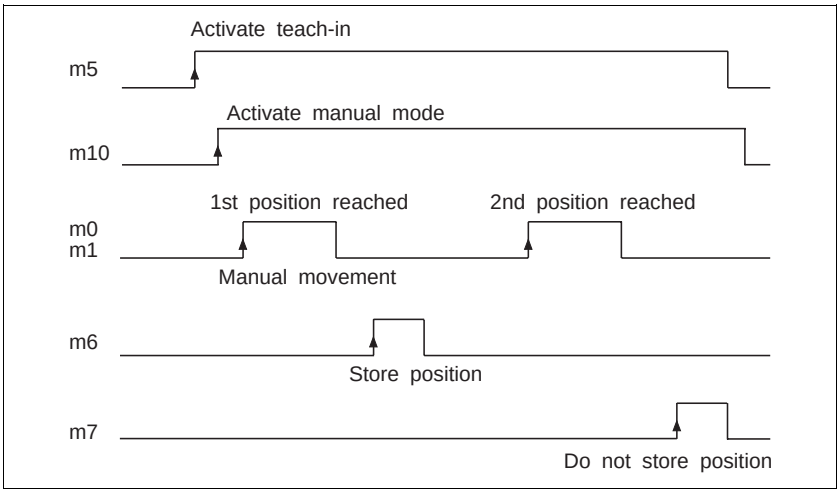


Fig. 1-5 Timing diagram for teach-in mode

– Step mode

In step-mode, the sequence program can be executed line by line (with ENTER key) and displayed on the screen. The PLC program continues to execute during step mode. See chapter 3.3.1 for information on how to activate step mode with parameter 34.

1.6 Movement programming

The following sections explain concepts and interrelations which are involved in programming of movements.

1.6.1 Axis operating modes

The “mode” command can be used to select the following axis operating modes in the sequence program (shaft is at standstill):

– Point-to-point mode
– Position following mode

1.6.1.1 Point-to-point mode

In point-to point mode, a positioning command is used to move from a defined point A to a defined point B. Positioning can be absolute (based on the axis zero position) or relative (based on the current axis position). In addition, you can specify whether the program should wait until positioning has been performed or continue immediately. Position values can be specified in user-defined units, e.g. 100 (mm).

Normalizing factor for position values

The parameters “Normalizing factor numerator” and “Normalizing factor denominator” can be used to convert user-defined units, e.g. 100 mm, to drive units, e.g. 1000 steps (or one motor revolution),

where:

Drive units = User-defined units x Normalizing factor

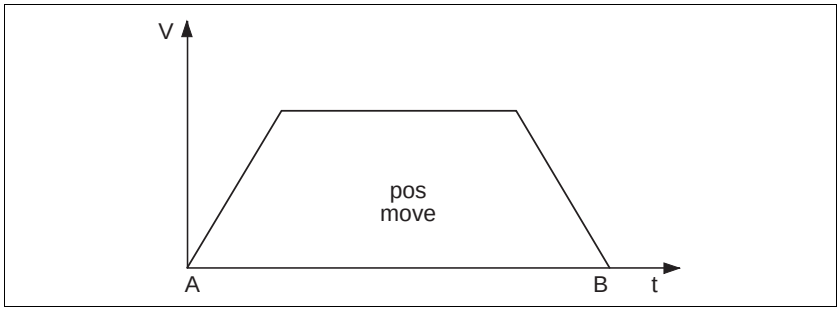


Fig. 1-6 Point-to-point mode

Programmable parameters for point-to-point mode:

– “Active limit switches”
– “Encoder evaluation” (for rotation monitoring)
– “Encoder setting” (for rotation monitoring)
– “Normalizing factor denominator”
– “Normalizing factor numerator”
– “Ramp”
– “Standard acceleration”
– “Standard speed”
– “Start/stop speed”

The parameter meanings and presettings are described in chapter 3.3.1.

Point-to-point mode movement commands:

Command	**Meaning**
“acc”	Select acceleration/deceleration curve
“mode”	Set axis operating mode (point-to-point mode)
“move”	Relative positioning in user-defined units
“movef”	Relative positioning in user-defined units and wait until position has been reached
“pos”	Absolute positioning in user-defined units
“posf”	Absolute positioning in user-defined units and wait until position has been reached
“ref”	Reference movement to limit or reference switch
“reff”	Reference movement to limit or reference switch and wait until reference point has been reached
“stop”	Stop shaft movement
“vel”	Adjust set speed

NOTE
During positioning with a “pos” or “move” command, the program continues to execute, i.e. input signals can be read, output signals can be set, the path or the speed for positioning can be changed.

Reference movement

The “ref” and “reff” commands can be used for reference movements.

NOTE
A reference movement is only possible in point-to-point mode.

With a reference movement, a reference point is approached which is to be used as the zero point for the system of dimensions.
The “type of reference movement” function can be used to determine whether the reference movement addresses the

– negative limit switch,
– positive limit switch or
– reference switch (clockwise or counterclockwise rotation).

The functional principle of the various forms of reference movements are evident in figures 1-7 and 1-8.
A safety distance to the limit switch or reference switch can be programmed with the “clearing distance” parameter.

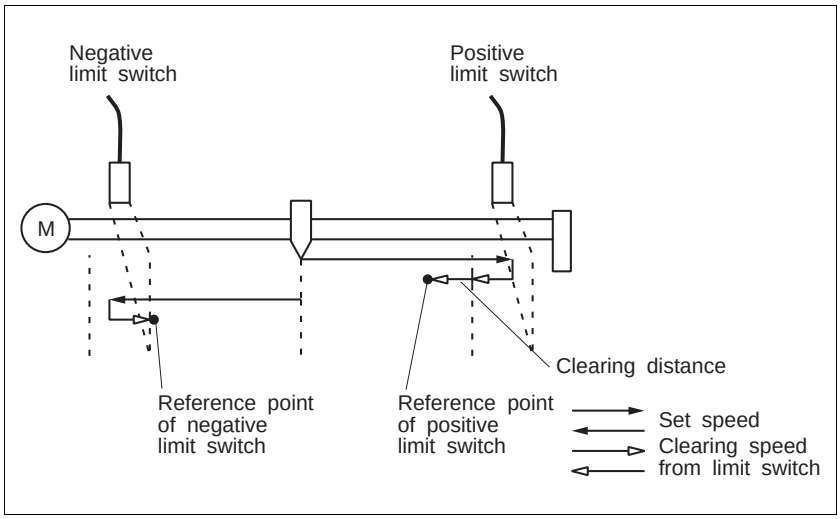


Fig. 1-7 Reference movement with limit switches

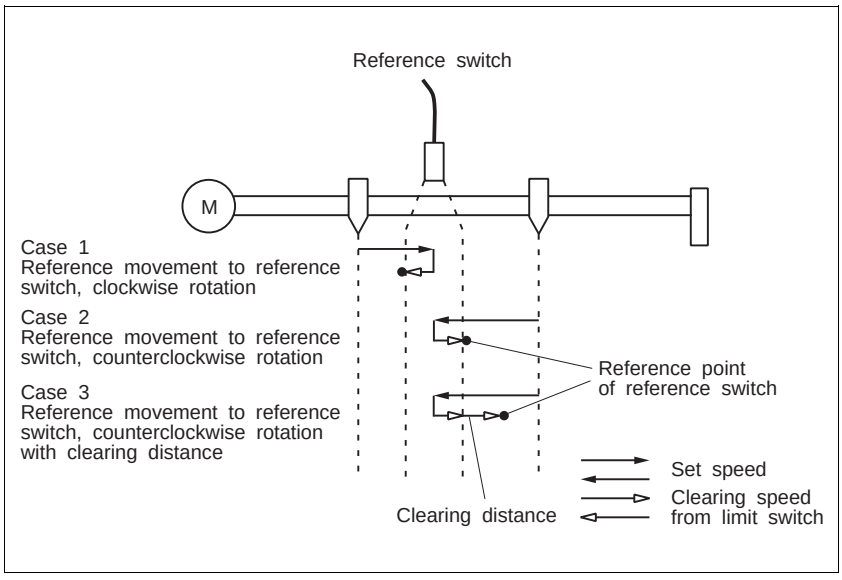


Fig. 1-8 Reference movement with reference switches

1.6.1.2 Position following mode

In position following mode, positions are specified through an encoder or a variable (see chapter 1.7). The setpoint can be modified with a gear ratio so that an electronic gear can be implemented.

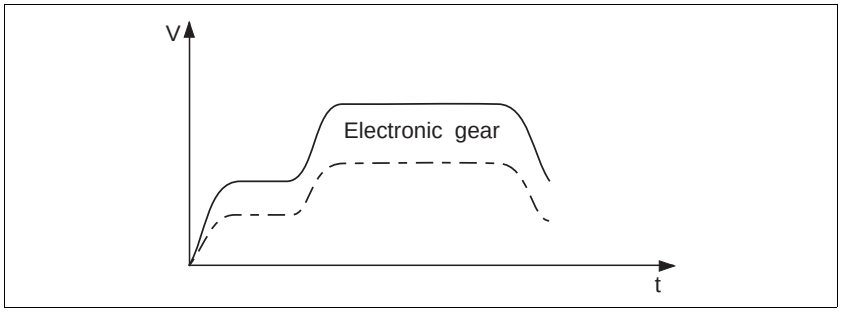


Fig. 1-9 Position following mode

Programmable parameters for position following mode:

– "Active limit switches"
– "Encoder evaluation"
– "Encoder setting"
– "Ramp" (only with position value presetting through a variable)
– "Gear interface signals"
– "Standard acceleration" (only with position value presetting through a variable)

The parameter meanings and presettings are described in chapter 3.3.1.

Position following mode movement commands:

Command	**Meaning**
"gearn"	Set gear ratio denominator
"gearz"	Set gear ratio numerator
"goff"	Input position offset
"mode"	Set axis operating mode (position following mode)
"stop"	Stop shaft movement

Movements are controlled in position following mode by a pulse signal on the encoder connection or by the value of a variable.

The “mode” command is used to determine whether position presetting is to be performed through an encoder connection or a variable.

– Encoder connection
  The encoder connection is selected with the parameter “encoder setting”. The frequency of the pulse signal on the encoder input determines the acceleration and the speed of the shaft. Absence of pulses means shaft standstill.

– Variable
  The value of a variable is interpreted as the absolute shaft position. If this value changes, the shaft moves to the new position using the selected acceleration curve (ramp) and the set speed.

NOTE
The separate OED3 software package includes a sample program for position following mode.

1.7 Encoder programming

Each encoder connection can be used for

– reference variable input (in position following mode) or
– rotation monitoring.

NOTE
The separate OED3 software package includes sample programs for encoder programming.

Figure 1-10 illustrates encoder programming and its effects on the controller.

Encoder evaluation

If an encoder is used for reference variable input or rotation monitoring, the encoder resolution (500 or 1000 marks) as well as the internal evaluation factor (single, dual or quadruple) must be communicated to the controller using the parameter “encoder evaluation”,

where:

$$\text{Motor revolutions} = \text{Encoder pulses} \times \frac{\text{Evaluationfactor}}{\text{Encoderresolution}}$$

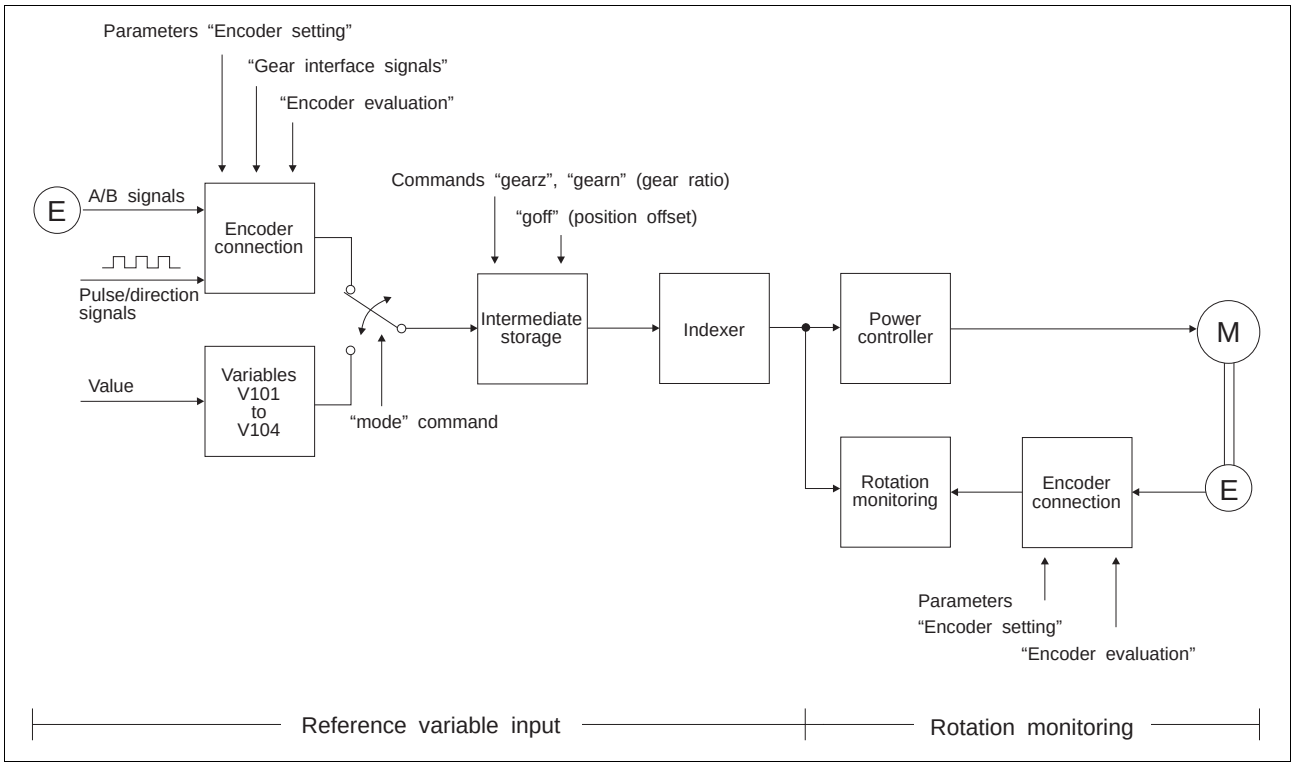


Fig. 1-10 Encoder programming

Reference variable input (position following mode)

In position following mode, the reference variable (position) is preset by

– an encoder input or
– a variable.

Electronic gear

The reference variable is multiplied by a gear ratio which means that an electronic gear can be implemented.
In addition, the reference variable can be adjusted with a position value using the “goff” comand (position offset).

If the reference variable changes too quickly, the programmed acceleration curve is used to accelerate or decelerate. An intermediate storage feature ensures that no pulses are lost.

NOTE
If the drive is stopped or another axis operating mode selected, any incoming pulses (positions) continue to be collected and intermediately stored. After switching back to position following mode these pulses are read out, which results in a relative positioning of the drive.

The following parameters and commands are important for reference variable input:

Parameter	**Meaning**
“Encoder evaluation”	This parameter is used to set single, dual or quadruple evaluation of the encoder signals (see chapter 3.3.1)
“Encoder setting”	This parameter is used to set: 0 No rotation monitoring Encoder connection 1 for position following mode 1 Rotation monitoring via encoder connection 1 Encoder connection 2 for position following mode 2 Rotation monitoring via encoder connection 2 Encoder connection 1 for position following mode
“Gear interface signals”	This parameter is used for setting: 0 Pulse/direction signals 1 A/B signals

Command	**Meaning**
“gearn”	Gear ratio denominator in user-defined units
“gearz”	Gear ratio numerator in user-defined units
“goff”	Position offset in user-defined units
“mode”	Set axis operating mode (position following mode)

NOTE
The separate OED3 software package includes a sample program for position following mode.

Rotation monitoring

Position errors can be prevented by activating rotation monitoring. This means that an encoder detects the actual position of the motor and compares it to the setpoint. If the difference exceeds a certain limit, the motor is decelerated.

The following parameters are important for rotation monitoring programming:

Parameter	**Meaning**
"Encoder evaluation"	This parameter is used to set single, dual or quadruple evaluation of the encoder signals (see chapter 3.3.1)
"Encoder setting"	This parameter is used to set: 0 No rotation monitoring Encoder connection 1 for position following mode 1 Rotation monitoring via encoder connection 1 Encoder connection 2 for position following mode 2 Rotation monitoring via encoder connection 2 Encoder connection 1 for position following mode

Response in case of rotation error

1. The motor decelerates at the set ramp.
2. The power controller switches off.

ATTENTION
The motor does not have any holding torque now.

3. The error message "12" is displayed in the status display 40.
4. The controller changes to STOP status.
5. All outputs are disabled.
6. Eliminate the cause of the fault (e.g. mechanical blocking, dirt).
7. Since the position has not been stored, restart program with a reference movement.

1.8 Interpolation with multi-axis positioning units

Several axes can be controlled simultaneously using series 300 multi-axis positioning units (e.g. WPM-311). The axes can be controlled independently of each other or with a defined dependence (interpolation). Linear interpolation is possible with two or three axes.

NOTE
Linear interpolation can be effected with absolute (position data relating to the axis zero point) or relative (position data relating to the current axis position) values, with or without waiting (the program is temporarily stopped until interpolation has been finished).

The following interpolation functions are possible with multi-axis positioning units:

Command	**Meaning**
“linpos**”**	Starting a linear interpolation to an absolute target position
“linposf”	Starting a linear interpolation to an absolute target position and wait until target position has been reached
“linmove”	Starting a linear interpolation to a relative target position
“linmovef”	Starting a linear interpolation to a relative target position and wait until target position has been reached
“setipos”	For each axis to be involved in linear interpolation a target position is preset in user-defined units, e.g.: setipos x1 1000 setipos x2 2000 setipos x3 2000

The “stopa” comand can be used to stop linear interpolation.

The following points must be observed for linear interpolation:

– The axes involved in interpolation must be set to point-to-point mode (see “mode” command).
– The speed and acceleration values of the axes involved must be entered before the interpolation is performed (see “vel” and “acc” commands).
– The target positions, speed and acceleration values of the axes involved cannot be changed during an interpolation process.
– Only one interpolation process at a time can be performed.
– The electronic gear may be affected by an interpolation process (it may be necessary to re-initialize the electronic gear after the interpolation process).

Principle of linear interpolation

Figure 1-11 shows the principle of linear interpolation by way of an example:

A movement from point A to point B is to be carried out with 2 axes (x1 and x2) using linear interpolation.

When the interpolation function is called (e.g. “setipos x1 500” and “setipos x2 200” commands), the axis setpoints are passed and the interpolation is invoked using the “linmove” command.

The linear interpolator calculates the speed and acceleration values required for interpolation from the setpoints specified and controls the individual axes. It ensures that the preset speed and acceleration values of the individual axes are not exceeded.

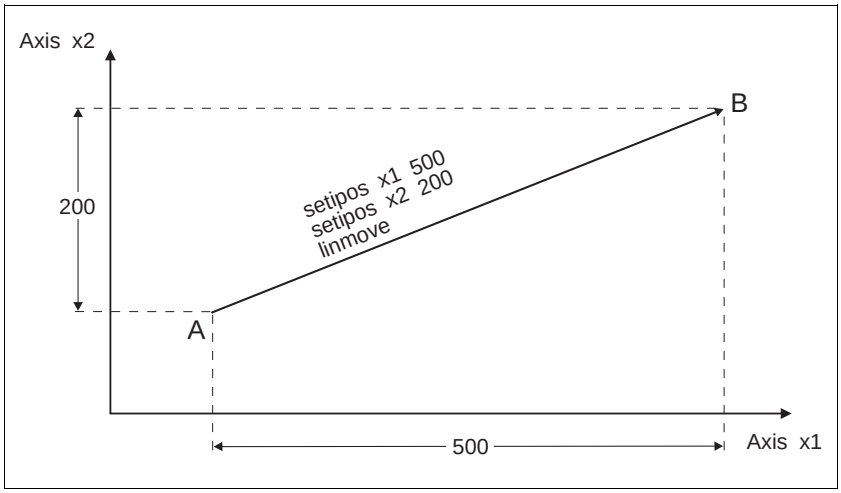


Fig. 1-11 Principle of linear interpolation

1.9 Position control with WDP3-337 and WDP3-338

For the WDP3-337 and WDP3-338 controllers, the actual position of the axis is detected via the resolver connection and passed to the internal position controller (fig. 1-12). The resolver is automatically set to quadruple evaluation for this purpose. For a resolver with a resolution of 1024 increments, this results in an internal resolution of 4096 encoder units (increments) per revolution.
The contouring error limit standard setting is 4096 encoder units. If the following error (the difference between set position and actual position) exceeds the contouring error limit, the motor is deenergized.

ATTENTION
A deenergized motor does not have a holding torque. This may cause mechanical damage. Protect vertical loads from falling down (use motor with brake).

A PID position controller is integrated in the WDP3-337 and WDP3-338 controllers. See the controller manual for control parameter setting.

NOTE
The values for the control parameters can be determined with the ONLINE3 setup software. The parameters thus determined must be entered into the OED3 parameter editor. You cannot change any parameters during positioning.

The encoder connection can be used for reference variable input in position following mode (see chapter 1.7).

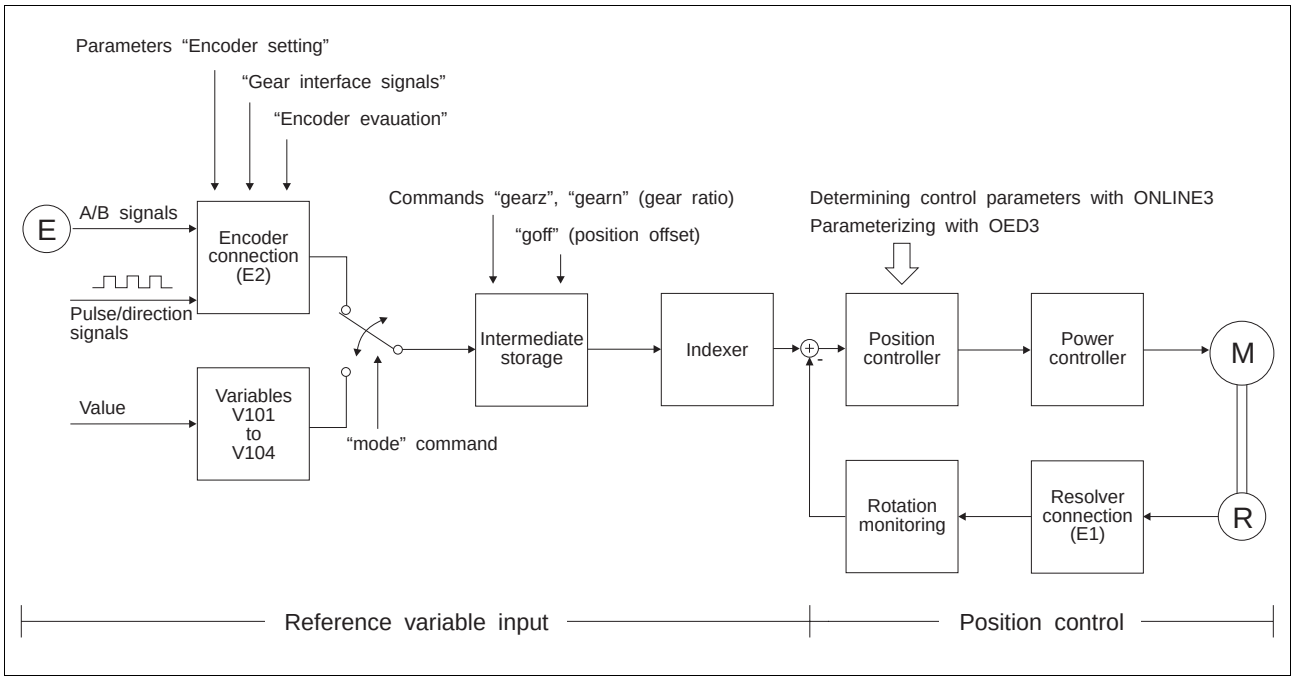


Fig. 1-12 Position control

1.10 Interface programming

The controller can be programmed and operated through the c1 or c2 serial interface, e.g. using the operating terminal FT 2000. The operator dialog can be independently programmed. The following values are preset for the c1 or c2 interface:

Baud rate	9600
Data bits	7
Parity	Even
Stop bits	1
Handshake	XON/XOFF

The parameter “control unit” can be used to determine how programming and operation through the serial interfaces are to be performed.

Serial interface c1

The parameter “control unit” means for a controller provided with one serial interface:

Parameter	Meaning
“Control unit”	0 Programming with OED3 or operator dialog with application program and multiple-line output 1 Programming with OED3 or operator dialog with application program and two-line error output (e.g. on FT 2000)

Serial interfaces c1 and c2

The parameter “control unit” means for a controller provided with two serial interfaces:

Parameter	Meaning
“Control unit”	0 Programming with OED3 or operator dialog with application program and multiple-line output through interface c2. Interface c1 without function. 1 Operator dialog with application program and two-line output (e.g. on FT 2000) through interface c1. Programming with OED3 through interface c2.

Character transfer through serial interface

Individual characters can be transferred. For this purpose, the following commands are available:

Commands	Meaning
“rec_char”	Receive character
“rec_var”	Receive variable
“snd_str”	Send string
“snd_var”	Send variable

NOTE
The string which is to be transferred with the “snd_str” command must be entered using the OED3 text editor, see chapter 3.6.

Character interpretation:

Characters in the controller input/output buffer	Direction of transmission	Characters through interface	
ASCII characters (16#20 - 16#7E)	↔	ASCII characters (16#20 - 16#7E)	
Other characters (16#7F - 16#FF)	↔	16#7F - 16#FF	
$$	↔	$ (16#24)	*
$'	↔	' (16#27)	*
$L	↔	<LF> (16#0A) Line Feed	*
$N	↔	(16#0A) New Line	*
$P	↔	<FF> (16#0C) Form Feed	*
$R	↔	<CR> (16#0D) Carriage Return	*
$T	↔	<TAB> (16#09) Tabulator	*
$00 - $1F	←	16#00 - 16#1F	*
$00 - $FF	→	16#00 - 16#FF	*

* When transferring these characters, a character conversion is carried out.

NOTE
The separate OED3 software package includes a sample program for FT 2000.

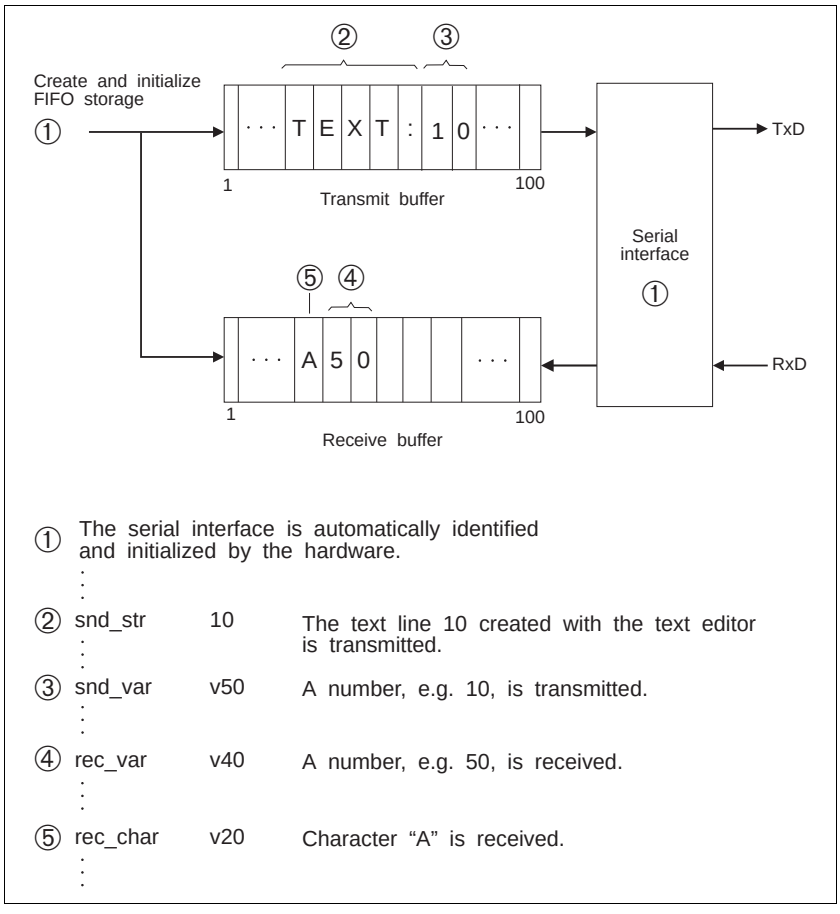


Fig. 1-13 Example for interface programming

1.11 Synchronization with master controller

Using the “handshake” command, synchronization with the master controller can be programmed through any input and output. This means that the synchronizing output indicates that a programmed position has been reached in the application program. The application program is stopped and waits for the synchronizing input to be energized (= 1). As soon as input = 1, the application program continues and the synchronizing output is reset.

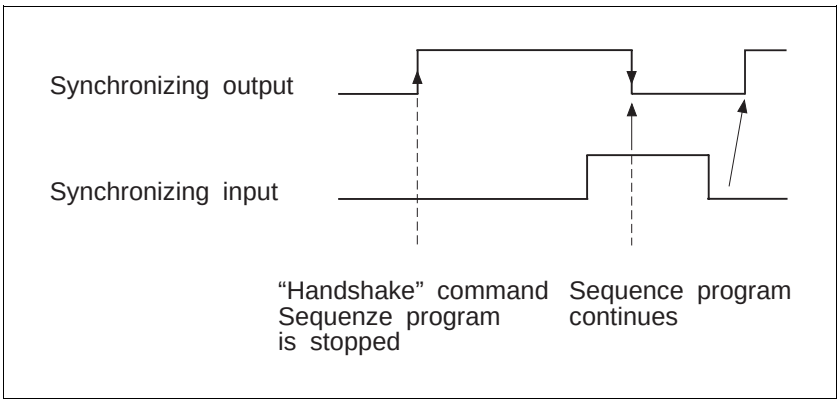


Fig. 1-14 Synchronization

1.12 Operation with input output card MP 926

The MP 926 input/output card is provided with 16 inputs and 16 outputs which can be addressed from a series 300 positioning unit with RS 485 HS interface.

The parameter 35 is used for setting the number of connected input/ output cards (5 cards max.).

The inputs are assigned the flags m32 to m111, the outputs are assigned the flags m112 to m191 (see fig. 1-15).
The flags can be read and set in the sequence editor and the PLC editor.

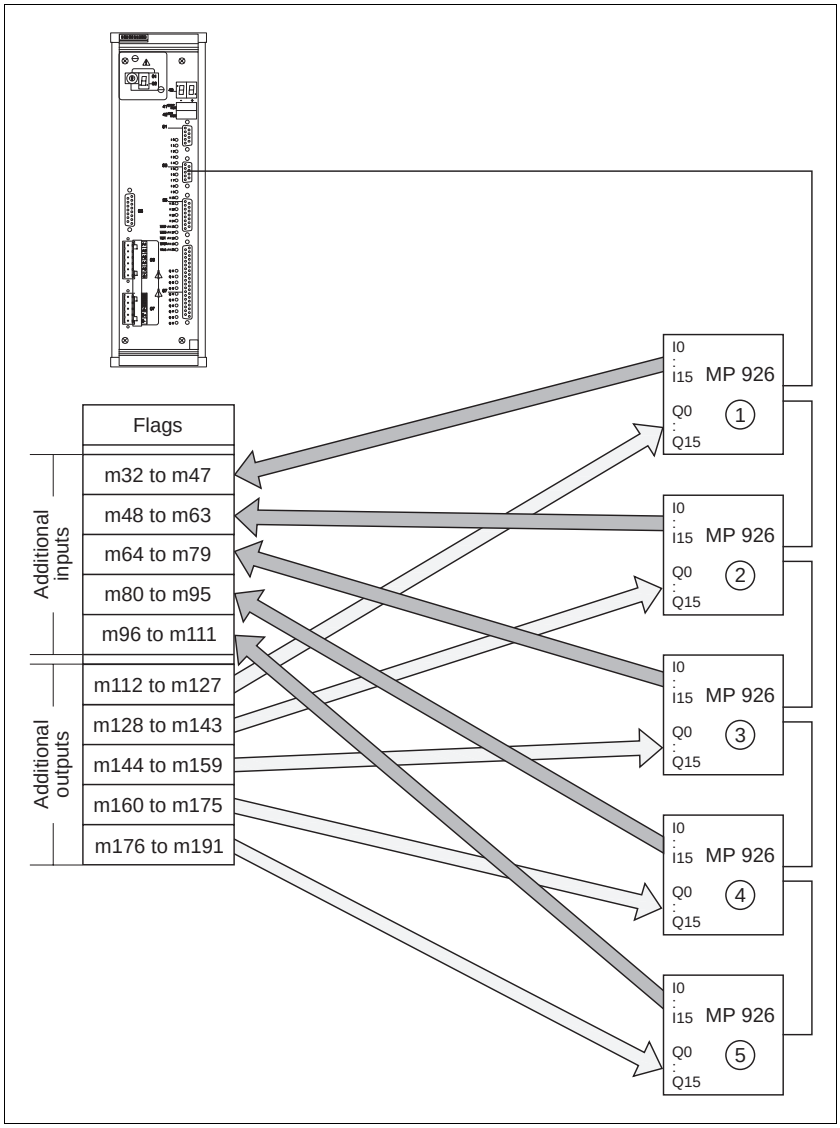


Fig. 1-15 Flag ranges for external inputs/outputs

2 Installation

2.1 Scope of supply

Check the scope of supply for completeness.

The scope of supply comprises:

Qty.	Designation
1	Series 300 controller (e.g. WDP5-318)
1	3½" disk with OED3 sample programs and BTERM
1	5¼" disk with OED3 sample programs and BTERM
1	Controller manual
1	OED3 documentation
1	BTERM documentation

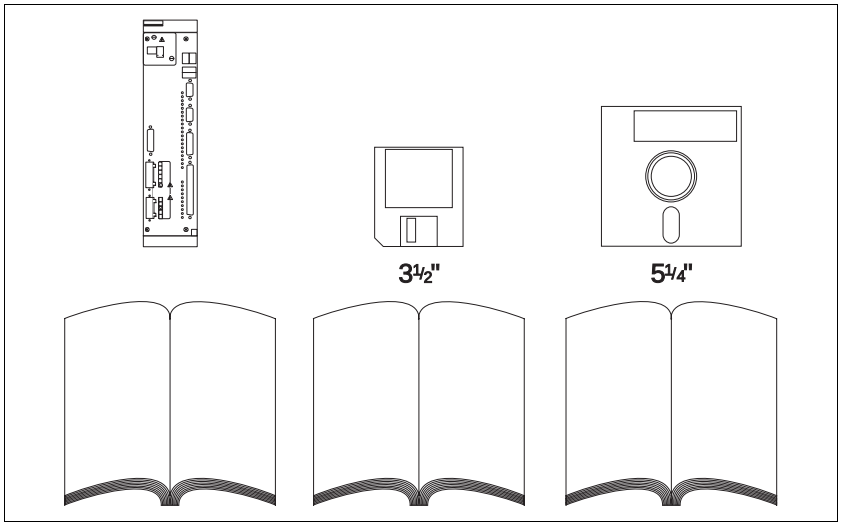


Fig. 2-1 Scope of supply

2.2 Accessories

The following optional accessories (refer to controller manual, chapter 6.3 for a description of accessories) are available:

– Interface cable male/female
– Interface cable male/male
– Interface converter MP 923 (RS 485/RS 232)

2.3 System requirements

System requirements for operating the programming interface:

- Terminal or PC with terminal program, e.g. BTERM, see scope of supply
- Interface cable for communication with the controllers
- If required, interface converter RS 485/RS 232 (e.g. MP 923)
- Series 300 controller with OED3

2.4 Connection

Communication between PC and controller is performed by serial data transmission through interface c1 or c2. Figure 2-2 shows how to connect the controller to the PC.

NOTE
The series 300 controllers which are provided with one serial interface are programmed through the c1 interface.
The controllers which are provided with two interfaces are programmed through the c2 interface. The c1 interface can be used for other purposes (see chapter 1.10).
The interface wiring is described in chapters 2 and 6 of the controller manual.

ATTENTION
When wiring, you first have to determine whether the controller is provided with an RS 485 interface (female connector) or an RS 232 interface (male connector).
If the controller is provided with an RS 485 interface and the PC with an RS 232 interface, an interface converter (e.g. MP 923) must be used.

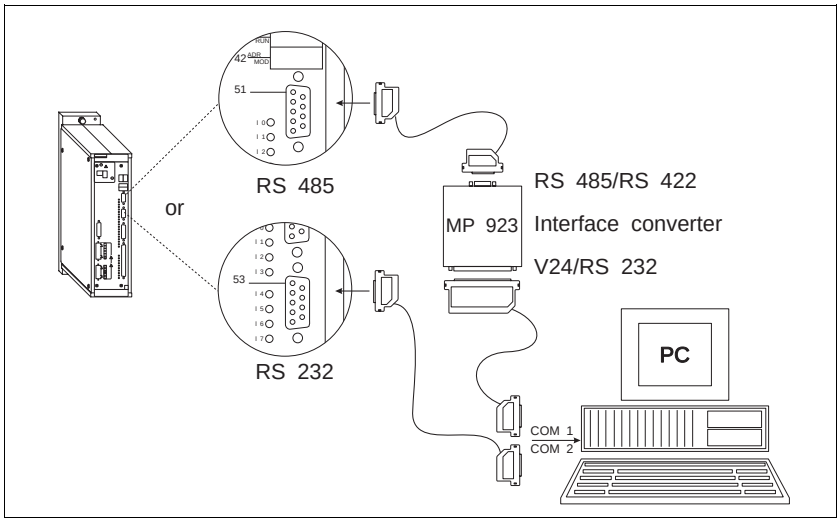


Fig. 2-2 Connection of controller, e.g. WDP5-318

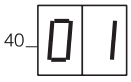
3 Programming

3.1 Starting OED3

Programming and operation with OED3 are only possible if installation has been performed according to chapter 2.

1. For controllers provided with two interfaces, connect the terminal or PC to the c2 serial interface, for controllers with only one interface to the c1 serial interface.
2. Switch on the terminal or PC. Start the terminal program BTERM for operation with PC.
3. Set the terminal or PC to the following interface parameters:

Baud rate	9600
Data bits	7
Parity	Even
Stop bits	1
Handshake	XON/XOFF



4. Switch on the controller and wait until “01” is displayed on the status display 40.

NOTE
If an application program is available on the controller, the program can be started by pressing selector switch 41 on + side (automatic mode).

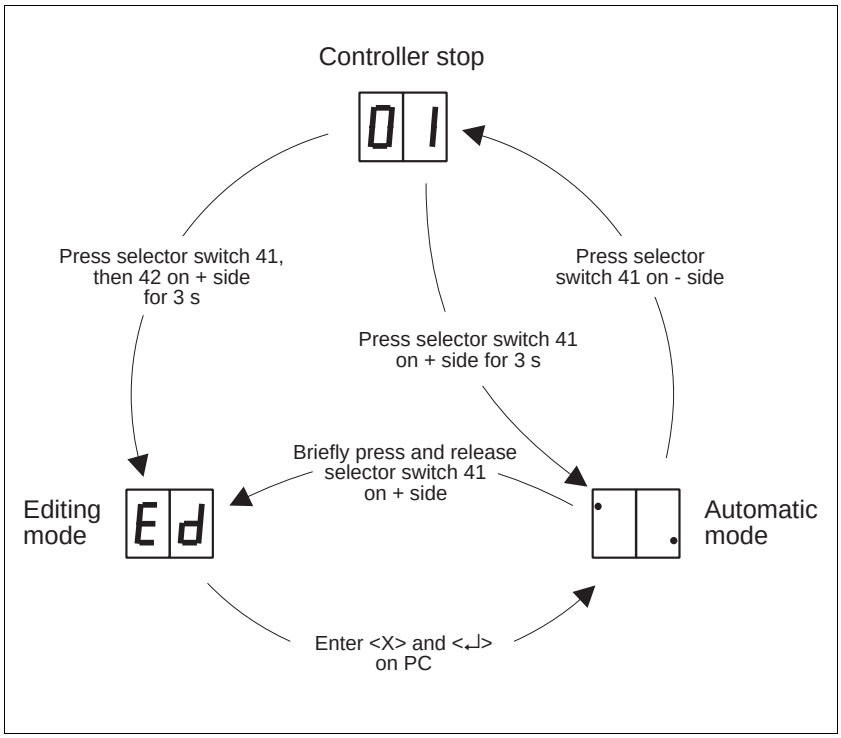
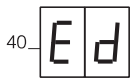


Fig. 3-1 Status changes



5. Press selector switch 41 on + (RUN) side and, within 3 s, also press selector switch 42 on + (MOD) side until “Ed” is displayed on the status display 40. The controller is in edit mode. Now it is possible to enter or edit an application program.

NOTE
If an application program is executed, selector switch 41 must be pressed on + (RUN) side for activating edit mode.

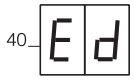
The following screen is displayed:

```
BERGER LAHR(R)     OED3  PR444.1 V1.01      SIG Positec
Copyright(C)    BERGER LAHR 1993.  All rights reserved.

EDIT>
```

Edit mode

In edit mode, an application program is created and edited using the following functions:



Parameter editor	see chapter 3.3
PLC editor	see chapter 3.4
Sequence editor	see chapter 3.5
Text editor	see chapter 3.6

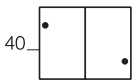
The functions

Upload	see chapter 3.7
Download	see chapter 3.8

are used to load an application program from the controller into the PC or from the PC into the controller, respectively.

Automatic mode

It is possible to leave edit mode with <X> and <↵>. After leaving edit mode, the application program is started (automatic mode). Two points are displayed on status display 40.



NOTE
When switching on the controller, the same mode is active which was active when switching off.

3.2 Programming interface

This section describes the general operating functions of the OED3 programming interface.

```
BERGER LAHR(R)     OED3  PR444.1 V1.01      SIG Positec
Copyright(C)    BERGER LAHR 1993.  All rights reserved.

EDIT> H

S PLC editor
T TXT editor
A SEQ editor
P PAR editor
U Upload
D Download
X Exit

EDIT> _
```

Fig. 3-2 Programming interface

- The header comprises program and version number.
- The flashing cursor indicates the input line. All inputs are acknowledged with <↵>.
- <H> is used to call the help function.
- <X> is used to exit the OED3 program.
- <ESC> is used to leave the current editor.

Line editor:

- Delete current line with <Ctrl> <Y>.
- Move to beginning of line with <Ctrl> <A>.
- Move to end of line with <Ctrl> <F>.
- Move one character to the right with <Ctrl> <D>.

NOTE
By entering “EEPROM”, the application program can be stored in the optional controller EEPROM (if installed).

3.2.1 Editors and functions

The following editors and functions are available:

PLC editor A PLC program is created or edited with the PLC editor.

Text editor The text editor is used to select the texts to be output in the application program on a display unit (e.g. FT 2000) using the “snd_str” command.

Sequence editor The sequence program is created or edited with the sequence editor. The inputs/outputs are directly read and written.

Parameter editor The parameter editor is used to set e.g. motor parameters, movement parameters and system parameters for the sequence program.

Upload Using the upload function all application data (e.g. sequence program, PLC program, text, parameters) are read out of the controller and displayed on the screen through the serial interface. The data can be saved on the PC and edited with a PC text editor.
In addition, comments can be entered in the program (a comment must start with a semicolon).

Download Using the download function, all data saved with the upload function are loaded into the controller and submitted to a validity check.
If an error is detected, the loading operation into the controller is interrupted and the remaining data are discarded.

3.3 Parameter editor

Select parameter editor	<P> <↵>
Call function selection	<H> <↵>

The following lines and sections are shown by way of example.

3.3.1 List parameters

List required line	<L3> <↵>
List required section	<L1-17> <↵>

```
    1: Start/stop speed                          100
    2: Standard speed                            1000
    3: Standard acceleration                     125
    4: Manual speed, slow                        200
    5: Manual speed, fast                        2000
    6: Inching distance for manual mode          10
    7: Normalizing factor numerator x1           1000
    8: Normalizing factor numerator x2 1)        1000
    9: Normalizing factor numerator x3 1)        1000
   10: Normalizing factor numerator x4 1)        1000
   11: Normalizing factor denominator x1         1000
   12: Normalizing factor denominator x2 1)      1000
   13: Normalizing factor denominator x3 1)      1000
   14: Normalizing factor denominator x4 1)      1000
   15: Ramp x1                                   0
   16: Ramp x2 1)                                0
   17: Ramp x3 1)                                0
PAR edit> _
```

Fig. 3-3 Lines 1 to 17 of parameter list

```
   18: Ramp x4 1)                                0
   19: Active limit switches x1                  3
   20: Active limit switches x2 1)               3
   21: Active limit switches x3 1)               3
   22: Active limit switches x4 1)               3
   23: Type of reference movement x1             0
   24: Type of reference movement x2 1)          0
   25: Type of reference movement x3 1)          0
   26: Type of reference movement x4 1)          0
   27: Clearing speed from limit switch          200
   28: Clearing distance from limit switch       0
   29: Encoder setting 2)                        0
   30: Encoder evaluation E1 2)                  0
   31: Encoder evaluation E2                     0
   32: Gear interface signals                    1
   33: Control unit                              0
   34: Step-by-step processing                   0
PAR edit> _
```

Fig. 3-4 Lines 18 to 34 of parameter list

1) only displayed for multi-axis positioning units
2) not displayed for WDP3-33X positioning units

```
  27: Clearing speed from limit switch        200
  28: Clearing distance from limit switch     0
  29: Encoder setting1)                       0
  30: Encoder evaluation E11)                 0
  31: Encoder evaluation E2                   0
  32: Gear interface signals                  1
  33: Control unit                            0
  34: Step-by-step processing                 0
  35: Interbus-S I/O modules2)                0
  36: Proportional component KP3)             0
  37: Integral component KI3)                 0
  38: Differential component KD3)             0
  39: Speed offset KF3)                       0
  40: Acceleration offset KA3)                0
  41: Manipulated variable offset3)           0
  42: Sampling time TA3)                      1

PAR-Edit> _
```

Fig. 3-5 Lines 27 to 42 of parameter list

1) not displayed for WDP3-33X positioning units
2) only displayed for positioning units with RS 485 HS interface
3) only displayed for WDP3-33X positioning units

The following table describes the sequence program parameters in alphabetical order.

Parameter	Default value	Setting range/Description
"Acceleration offset KA"[1)]	0	±16383 The acceleration offset compensates the acceleration-related following error.
"Active limit switches x1 to x4"[2)]	3	0 = No limit switch active 1 = Negative limit switch active 2 = Positive limit switch active 3 = Both limit switches active After reaching a limit switch, the drive is set to standstill using the defined ramp. The controller is then in error condition.
"Clearing distance from limit switch"	0 steps	0 to 100 steps After successful completion of reference movement, the motor is moved away from the reached limit or reference switch using this value.
"Control unit"	0	0 = ASCII terminal with 24 lines and 80 characters/line 1 = Two-line terminal (e.g. FT 2000)
"Differential component KD"[1)]	0	±16383 The differential component determines the response characteristics to control deviation changes.
"Encoder evaluation E1 or E2"[3)]	0	0 = 500-mark encoder, single evaluation 1 = 500-mark encoder, dual evaluation 2 = 500-mark encoder, quadruple evaluation 3 = 1000-mark encoder, single evaluation 4 = 1000-mark encoder, dual evaluation 5 = 1000-mark encoder, quadruple evaluation

Parameter	Default value	Setting range/Description
"Encoder setting"[3)]	0	0 = No rotation monitoring Encoder connection 1 for position following mode 1 = Rotation monitoring through encoder connection 1 Encoder connection 2 for position following mode 2 = Rotation monitoring through encoder connection 2 Encoder connection 1 for position following mode
"Gear interface signals"	1	0 = Pulse/direction signals 1 = A/B signals Input signals on encoder connection for position following mode.
"Inching distance for manual mode"	10 steps	0 to 100 steps Number of steps after status change 0 → 1 of flags m0 and m1.
"Integral component KI"[1)]	0	0 to 16383 The integral component balances a constant control deviation.
"Interbus-S I/O modules"[4)]	0	0 to 5 Number of MP 926 input/output cards connected to the RS 485 HS interface (c2) (0 = No card connected)  *NOTE* *The value must correspond to the actual number of input/output cards.*
"Manipulated variable offset"[1)]	0	±2147483647 Trimming parameter for preventing motor offset drift.
"Manual speed, fast"	2000 Hz	1 to 10000 Hz Frequency used for fast movements in manual mode.
"Manual speed, slow"	200 Hz	1 to 10000 Hz Frequency used for movements in manual mode.
"Normalizing factor denominator x1 to x4"[2)]	1000	±2147483647 The normalizing factor is used to convert user-defined units (e.g. mm) to drive units (steps or increments) in point-to-point mode.
"Normalizing factor numerator x1 to x4"[2)]	1000	
"Proportional component"[1)]	0	0 to 16383 The proportional component defines the rigidity of a drive.
"Ramp x1 to x4"[2)]	1	0 = Linear ramp 1 = Exponential ramp 2 = Sin^2 ramp Curve to be used to accelerate or decelerate.
"Sampling time TA"[1)]	1	1 = 1 ms, 2 = 2 ms, 3 = 4 ms, 4 = 8 ms The interval in which set position and actual position are compared and the PID position controller calculates a new setpoint.
"Speed from limit switch"	200 Hz	1 to 10000 Hz Speed used to move away from the reached limit switch.
"Speed offset KF"[1)]	0	±16383 The speed offset compensates the speed-related following error.

Parameter	Default value	Setting range/Description
"Standard acceleration"	125 Hz/ms	1 to 2000 Hz/ms If no acceleration is specified in the application program, all axes x1 to x4 are acccelerated or decelerated using this acceleration.
"Standard speed"	10000 Hz	1 to 100000 Hz If no other speed is specified in the application program, all active shafts x1 to x4 are moved using this maximum frequency.
"Start/Stop speed"	100 Hz	1 to 10000 Hz Speed used to start or stop a shaft movement.
"Step-by-step processing"	0	0 to 1000 0 = Sequence program execution without stop 1 to 1000 = Step-by-step execution of sequence program starting with the specified line When entering 0, the sequence program is executed without stop. A line number is entered in the sequence program, indicating where the step mode is to be started. One line is executed with each <↵> entered through the serial interface. *NOTE* *The program lines are output when the flags m9 and m12 are set.* Step mode is reset upon starting the sequence program and initiated upon reaching the selected line.
"Type of reference movement x1 to x4"[2)]	0	0 = To negative limit switch 1 = To positive limit switch 2 = To reference switch, counterclockwise rotation 3 = To reference switch, clockwise rotation Specifies the axis which is to address a specific limit switch.

1) only displayed for WDP3-33X positioning units
2) "x2 to x4" is only displayed for multi-axis positioning units
3) not displayed for WDP3-33X positioning units
4) only displayed for positioning units with RS 485 HS interface

3.3.2 Edit parameters

Call parameter line	<E1> <↵>
Scroll through parameter list	<↵>
End editing	<ESC>

NOTE
Changed parameters remain in storage after turning off.

```
PAR edit> E1
    1: Start/stop speed                          100
    2: Standard speed                            10000
    3: Standard acceleration                     125
    4: Manual speed, slow                        200
    5: Manual speed, fast                        2000
    6: Inching distance for manual mode          10
```

Fig. 3-6 Editing parameter line 6

Example:

Change inching distance for manual mode to “20”:

PAR edit>E6 <↵> <20> <↵> <ESC>

3.4 PLC editor

Select PLC editor	<S> <↵>
Call function selection	<H> <↵>

Insert	see chapter 3.4.3
Delete	see chapter 3.4.4
List	see chapter 3.4.2
Edit	see chapter 3.4.5
Command list	see chapter 3.4.1

The following lines and sections are shown by way of example.

```
BERGER LAHR(R)     OED3  PR444.1 V1.01      SIG Positec
Copyright(C)    BERGER LAHR 1993.  All rights reserved.

EDIT> S
PLC edit> H

I Insert
D Delete
L List
E Edit
B Command list

PLC edit> _
```

Fig. 3-7 PLC editor

3.4.1 Command list

Call command list <B> <↵>

```
PLC edit> B

nop               add     <kv>      and       <iqm>
andn    <iqm>     ld      <kviqmxb> ldn       <iqmb>
or      <viqm>    orn     <viqm>    r         <qm>
s       <qm>      st      <viqmx>   stn       <iqm>
sub     <kv>      eq      <kviqm>   gt        <kv>
lt      <kv>      jmp     <lk>      jmpc      <lk>
jmpn    <lk>      mul     <kv>      div       <kv>
label   <l>       stimer  <t>       end

PLC edit> _
```

Fig. 3-8 PLC editor command list

A command in the command list always appears as follows:

	Operator	<possible operands>
	ld	<k>
e.g.	ld	100

The subsequent tables describe possible operators and operands.

Available operands:

Operand	Meaning		Range of values
b	**Boolean variable**		0 or 1
i	**Inputs**		0 or 1
	i0 to i15	For WDP3-33X, WDP5-318, WP-311	
	i0 to i20	For WPM-311	
k	**Constants** (value)		±2 x 10^9
l	**Labels** l1 to l50		
m	**Flags**		0 or 1
	m0	Manual movement to right	
	m1	Manual movement to left	
	m2	Rapid speed	
	m3 m4	Multi-axis selection bit 0 Multi-axis selection bit 1 Axis / m4 / m3: 1 / 0 / 0 2 / 0 / 1 3 / 1 / 0 4 / 1 / 1	
	m5	Activate teach-in	
	m6	Store position	
	m7	Do not store position	
	m8	Reserved	
	m9	Test output: 0 = Off, 1 = On	
	m10	Activate manual mode: 0 = Inactive, 1 = Active	
	m11	Manual mode active? 0 = No, 1 = Yes	
	m12	Test output: 0 = Line number, 1 = Complete line	
	m13 to m20	Reserved	
	m21 to m1000 m32 to m111 m112 to m191	Freely available; for controllers with RS 485 HS or Interbus-S interface: For extended inputs For extended outputs	
	m1001	Axis x1: 0 = Axis stopped, 1 = Axis positioning	
	m1002	Axis x2: 0 = Axis stopped, 1 = Axis positioning	
	m1003	Axis x3: 0 = Axis stopped, 1 = Axis positioning	
	m1004	Axis x4: 0 = Axis stopped, 1 = Axis positioning	
	m1005 to m1023	Reserved	
q	**Outputs** q0 to q9		0 or 1
t	**Timer** t1 to t5 (100 ms resolution)		0 to 2 x 10^9
v	**Variables**		±2 x 10^9
	v0 to v100	Freely available	
	v101	Position following variable, axis x1	
	v102	Position following variable, axis x2	
	v103	Position following variable, axis x3	
	v104	Position following variable, axis x4	
	v105 to v114	Reserved	
x	**Axis**		
	x1	For WDP3-33X, WDP5-318, WP-311	
	x1 to x4	For WPM-311	

Available operators for PLC commands:

Operator	Meaning
“add”	Arithmetic operation: addition
“and”	Logical operation: AND operation
“andn”	Logical operation: NAND operation
“div”	Arithmetic operation: division
“end”	End PLC program
“eq”	Relational operation: =
“gt”	Relational operation: >
“jmp”	Jump to label
“jmpc”	Conditional jump, if CR = TRUE (1)
“jmpn”	Conditional jump, if CR = FALSE (0)
“label”	Label
“ld”	Transfer operation
“ldn”	Transfer operation
“lt”	Relational operation: <
“mul”	Arithmetic operation: multiplication
“nop”	End PLC program
“or”	Logical operation: OR operation
“orn”	Logical operation: NOR operation
“r”	Reset boolean operand
“s”	Set boolean operand
“st”	Transfer operation
“stimer”	Timer
“stn”	Transfer operation
“sub”	Arithmetic operation: subtraction

Current result CR

The current result is contained in an intermediate storage feature (accumulator) of the controller which is used for arithmetic operations, logical operations, and data transfer.

Example	Meaning
ld 15 add 10 st v10	Load value 15 into CR. Add value 10 to CR. Store result as v10 variable.

Description of PLC commands

add

Syntax: `add` `<kv>`

The “add” command is used to carry out addition with an operand and the CR.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`add`	`300`	Add value 300 to CR contents (value 100).
`st`	`v5`	Store result as variable v5.
:		

and

Syntax: `and` `<iqm>`
`andn` `<iqm>`

The “and” command is used to carry out a logical operation with a boolean operand and the CR.
A boolean negation of the operand can be performed prior to the logical operation by adding the suffix “n”. The suffix “n” must be appended to the command, e.g.“andn”.

Example 1:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 in CR.
`and`	`i2`	AND operation with inputs i1 and i2. CR = i1 and i2
`st`	`m15`	Store CR contents as flag m15.
:		

Example 2:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 into CR.
`andn`	`i3`	NAND operation with input i3 and CR. CR = CR andn i3
`st`	`m11`	Store CR contents as flag m11.
:		

div

Syntax: `div <kv>`

The “div” command is used to perform division with an operand and the CR.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`div`	`20`	Divide CR contents (value 100) by 20.
`st`	`v4`	Store result as variable v4.
:		

end

Syntax: `end`

The “end” command is used to end a PLC cycle. The process image is updated and a new cycle is started, beginning with line 1.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`add`	`300`	Add value 300 to CR contents (value 100).
`st`	`v5`	Store result as variable v5.
`end`		End of cycle.

eq

Syntax: `eq` `<kviqm>`

The “eq” command is used to perform a comparison (=) with an operand and the CR. After comparison, the CR always reads:

0 or FALSE, if the comparison is false or

1 or TRUE, if the comparison is true.

Example:

Operator	Operand	Meaning
:		
`ld`	`v100`	Load variable v100 into CR.
`eq`	`100`	Perform comparison. Is CR = 100? CR = 1 if comparison is true, if not, CR = 0.
:		

gt

Syntax: `gt` `<kv>`

The “gt” command is used to perform a comparison (>) with an operand and the CR. After comparison, the CR always reads:

0 or FALSE, if the comparison is false or

1 or TRUE, if the comparison is true.

Example:

Operator	Operand	Meaning
:		
`ld`	`v100`	Load variable v100 into CR.
`gt`	`1500`	Perform comparison with CR > 1500. CR = 1 if the comparison is true, if not, CR = 0.
:		

jmp

Syntax:

```
jmp     <lk>
jmpc    <lk>
jmpn    <lk>
```

The “jmp” command is used to carry out conditional and unconditional jump operations. Using the suffixes “c” and “n”, conditional jump operations can be performed. “k” specifies a relative jump by k lines.
Jump instructions provided with the suffix “c” are only executed if the value in the CR is not equal to 0.
Jump instructions provided with the suffix “n” are only executed if the value in the CR equals 0.

Example 1:

Operator	Operand	Meaning
:		
`label`	`l1`	Assign label l1.
`ld`	`v100`	Load variable v100 into CR.
`add`	`100`	Add value 100 to CR.
`st`	`v100`	Store result as variable v100.
`jmp`	`l1`	Jump to label l1.
:		

Example 2:

Operator	Operand	Meaning
:		
`ld`	`i12`	Load input i12 into CR.
`jmpc`	`l10`	if i12 ≠ 0, go to label l10.
:		
`label`	`l10`	Assign label l10.

Example 3:

Operator	Operand	Meaning
:		
`ld`	`i10`	Load input i10 into CR.
`jmpn`	`-1`	If i10 = 0, go back by one line.

label

Syntax: `label <l>`

The “label” command is used to assign a label for jump operations and subroutine calls.

Example:

Operator	Operand	Meaning
:		
`label`	`l1`	Assign label l1.
`ld`	`v100`	Load variable v100 into CR.
`add`	`100`	Add value 100 to CR.
`st`	`v100`	Store result as variable v100.
`jmp`	`l1`	Jump to label l1.
:		

ld

Syntax: `ld <kviqmxb>`
`ldn <iqmb>`

The “ld” command is used to load an operand into the CR.
A boolean negation of the operand can be performed using the “n” suffix.

Example 1:

Operator	Operand	Meaning
:		
`ld`	`i1`	Load input i1 into CR.
`st`	`m15`	Store CR contents (input i1) as flag m15.
:		

Example 2:

Operator	Operand	Meaning
:		
`ldn`	`i1`	Load negated input i1 into CR.
`ld`	`i2`	Load input i2 into CR.
`st`	`m16`	Store CR contents as flag m16.
:		

lt

Syntax: `lt` `<kv>`

The “lt” command is used to perform a comparison (<) with an operand and the CR. After comparison, the CR always reads:

0 or FALSE, if the comparison is false or

1 or TRUE, if the comparison is true.

Example:

Operator	Operand	Meaning
:		
`ld`	`v110`	Load variable v110 into CR.
`lt`	`1800`	Perform comparison with CR < 1800. CR = 1 if the comparison is true, if not, CR = 0.
:		

mul

Syntax: `mul` `<kv>`

The “mul” command is used to perform multiplication with an operand and the CR.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`mul`	`20`	Multiply CR contents (value 100) by 20.
`st`	`v3`	Store result as variable v3.
:		

nop

Syntax: `nop`

The “nop” command is used to end the PLC program.

or

Syntax:
```
or      <viqm>
orn     <viqm>
```

The “or” command is used to carry out a logical operation with a boolean operand and the CR.
A boolean negation of the operand can be performed prior to the logical operation by adding the suffix “n”. The suffix “n” must be appended to the command, e.g. “orn”.

Example 1:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 into CR.
`or`	`i2`	OR operation with inputs i1 and i2. CR = i1 or i2
`st`	`m17`	Store CR contents as flag m17.

Example 2:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 into CR.
`and`	`i2`	AND operation with inputs i1 and i2. CR = i1 and i2.
`orn`	`i3`	NOR operation with CR and input i3. CR = CR orn i3
`st`	`m11`	Store CR contents as flag m11.

r

Syntax: `r` `<qm>`

The “r” command is used to reset an output or flag to 0, depending on the current result (CR).

Example:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 into CR.
`and`	`i2`	AND operation with inputs i1 and i2. CR = i1 and i2.
`st`	`m11`	Store CR contents as flag m11.
`r`	`q5`	Output q5 is set to 0, if CR = 1.

s

Syntax: `s` `<qm>`

The “s” command is used to set an output or flag to 1, depending on the current result (CR).

Example:

Operator	Operand	Meaning
`ld`	`i1`	Load input i1 into CR.
`and`	`i2`	AND operation with inputs i1 and i2. CR = i1 and i2.
`st`	`m13`	Store CR contents as flag m13.
`s`	`q5`	Output q5 is set to 1, if CR = 1.

st

Syntax: `st <viqmx>`
`stn <iqm>`

The “st” command is used to store an operand in the CR.
A boolean negation of the operand can be performed using the suffix “n”.

Example 1:

Operator	Operand	Meaning
:		
`ld`	`i1`	Load input i1 into CR.
`st`	`m13`	Store CR contents (input i1) as flag m13.
:		

Example 2:

Operator	Operand	Meaning
:		
`ld`	`i1`	Load input i1 into CR.
`stn`	`m14`	Store CR contents (negation of input i1) as flag m14.
:		

stimer

Syntax: `stimer <t>`

Timers are special variables, the values of which change as a function of the time. Five timers are defined, t1 to t5. The “stimer” command is used for loading a timer with a time value and activate it. The time resolution is 100 ms (time = time x 100 ms). Specifying “0” for a timer stops the timer. You can set a new time at any time.

Example:

Operator	Operand	Meaning
:		
`ld`	`10`	Load time value, 1 second in this case.
`st`	`t1`	Store time value in timer 1.
`stimer`	`t1`	Activate timer 1.
:		
`ld`	`t1`	
`eq`	`0`	If timer 1 = 0,
`st`	`q1`	set output 1.

sub

Syntax: `sub` `<kv>`

The “sub” command is used to perform subtraction with an operand and the CR.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`sub`	`30`	Subtract value 30 from CR contents (value 100).
`st`	`v6`	Store result as variable v6.
:		

3.4.2 List PLC program

List PLC program <L> <↵>
List required section <L1-17> <↵>

```
PLC edit> l1-17

 1:    ld       i1
 2:    and      i2
 3:    st       m5
 4:    s        q5
 5:    ldn      i1
 6:    orn      i2
 7:    r        q5
 8:    ld       v100
 9:    gt       1500
10:    s        q6
11:    ld       17
12:    ld       true
13:    st       q7
14:    jmp      l1
15:    ld       false
16:    st       q7
17:    end

PLC edit> _
```

Fig. 3-9 PLC sample program

3.4.3 Insert line

Insert line, e.g. from line 5 <I5> <↵>
End inserting and editing <ESC>

Example:

```
PLC edit>I5 <↵>
PLC edit-I0005>LD V15 <↵>
PLC edit-I0006>ADD 100 <↵>
PLC edit-I0007> <ESC>
```

3.4.4 Delete line(s)

Delete line, e.g. line 5	<D5> <↵>
Delete lines, e.g. lines 1 to 5	<D1-5> <↵>

NOTE
The delete operation is immediately carried out and the entire PLC program is updated.

Example:

```
PLC edit>D5 <↵>
```

NOTE
After entering <↵>, *OK is displayed.*

3.4.5 Edit line

Call required line, e.g. line 11	<E11> <↵>
End editing	<ESC>

Example:

```
PLC edit>E11 <↵>
PLC edit-0011>LD V100 <↵>
PLC edit-0012>LD V110 <↵>
PLC edit-0013>GT 1500 <↵>
PLC edit-0014> <ESC>
```

3.5 Sequence editor

Select sequence editor	<A>	<↵>
Call function selection	<H>	<↵>

Insert	see chapter 3.5.3
Delete	see chapter 3.5.4
List	see chapter 3.5.2
Edit	see chapter 3.5.5
Command list	see chapter 3.5.1

The following lines and sections are shown by way of example.

```
BERGER LAHR(R)     OED3  PR444.1 V1.01      SIG Positec
Copyright(C)    BERGER LAHR 1993.  All rights reserved.


EDIT> A
SEQ edit> H

I Insert
D Delete
L List
E Edit
B Command list

SEQ edit> _
```

Fig. 3-10 Sequence editor

3.5.1 Command list

Call command list <B> <↵>

```
SEQ edit> B

nop                acc     <x><kv>    add        <kv>
and      <iqm>     andn    <iqm>      div        <kv>
eq       <kviqm>   gearn   <x><kv>    gearz      <x><kv>
goff     <x><kv>   gt      <kv>       handshake  <i><q>
jmp      <lk>      jmpc    <lk>       jmpn       <lk>
ld       <kviqmxb> lt      <kv>       move       <x><kv>
movef    <x><kv>   mul     <kv>       or         <viqm>
orn      <viqm>    pos     <x><kv>    posf       <x><kv>
mode     <x><k>    r       <qm>       rec_var    <v>
rec_char<v>        ref     <x>        reff       <x>
s        <qm>      snd_str <kv>       snd_var    <v>
st       <viqmx>   stop    <x>        sub        <kv>
vel      <x><kv>   wait    <kv>       cal        <l>
label    <l>       ret                end

SEQ edit> _
```

Fig. 3-11 Command list

A command in the command list is always described as follows:

	Operator	<possible operands>
	`ld`	`<k>`
e.g.	`ld`	`100`

The following tables describe the available operators and operands.

Available operands:

Operand	Meaning		Range of values
b	**Boolean variable**		0 or 1
i	**Inputs**		0 or 1
	i0 to i15	For WDP3-33X, WDP5-318, WP-311	
	i0 to i20	For WPM-311	
k	**Constants** (value)		$\pm 2 \times 10^9$
l	**Labels** l1 to l50		
m	**Flags**		0 or 1
	m0	Manual movement to right	
	m1	Manual movement to left	
	m2	Rapid speed	
	m3 m4	Multi-axis selection bit 0 Multi-axis selection bit 1	Axis: 1, 2, 3, 4 m4: 0, 0, 1, 1 m3: 0, 1, 0, 1
	m5	Activate teach-in	
	m6	Store position	
	m7	Do not store position	
	m8	Reserved	
	m9	Test output: 0 = Off, 1 = On	
	m10	Activate manual mode: 0 = Inactive, 1 = Active	
	m11	Manual mode active? 0 = No, 1 = Yes	
	m12	Test output: 0 = Line number, 1 = Complete line	
	m13 to m20	Reserved	
	m21 to m1000 m32 to m111 m112 to m191	Freely available; for controllers with RS 485 HS or Interbus-S interface: For extended inputs For extended outputs	
	m1001	Axis x1: 0 = Axis stopped, 1 = Axis positioning	
	m1002	Axis x2: 0 = Axis stopped, 1 = Axis positioning	
	m1003	Axis x3: 0 = Axis stopped, 1 = Axis positioning	
	m1004	Axis x4: 0 = Axis stopped, 1 = Axis positioning	
	m1005 to m1023	Reserved	
q	**Outputs** q0 to q9		0 or 1

Operand	Meaning		Range of values
v	**Variables**		$\pm 2 \times 10^9$
	v0 to v100	Freely available	
	v101	Position following variable, axis x1	
	v102	Position following variable, axis x2	
	v103	Position following variable, axis x3	
	v104	Position following variable, axis x4	
	v105 to v114	Reserved	
x	**Axis**		
	x1	For WDP3-33X, WDP5-318, WP-311	
	x1 to x4	For WPM-311	

Available operators for sequence commands:

Operator	Meaning
“acc”	Select acceleration/deceleration curve
“add”	Arithmetic operation: addition (see chapter 3.4.1)
“and”	Logical operation: AND operation (see chapter 3.4.1)
“andn”	Logical operation: NAND operation (see chapter 3.4.1)
“cal”	Call subroutine
“div”	Arithmetic operation: division
“end”	End sequence program
“eq”	Relational operation: = (see chapter 3.4.1)
“gearn”	Set gear ratio denominator
“gearz”	Set gear ratio numerator
“goff”	Position offset for reference variable
“gt”	Relational operation: > (see chapter 3.4.1)
“handshake”	Synchronization with master controller
“jmp”	Jump to label (see chapter 3.4.1)
“jmpc”	Conditional jump (see chapter 3.4.1)
“jmpn”	
“label”	Label (see chapter 3.4.1)
“ld”	Transfer operation (see chapter 3.4.1)
“linmove”[1)]	Start relative linear interpolation in user-defined units
“linmovef”[1)]	Start relative linear interpolation in user-defined units and wait until target position has been reached
“linpos”[1)]	Start absolute linear interpolation in user-defined units
“linposf”[1)]	Start absolute linear interpolation in user-defined units and wait until target position has been reached

Operator	Meaning
“lt”	Relational operation: < (see chapter 3.4.1)
“mode”	Set axis operating mode
“move”	Relative positioning in user-defined units
“movef”	Relative positioning in user-defined units and wait until position has been reached.
“mul”	Arithmetic operation: multiplication
“nop”	End program (see chapter 3.4.1)
“or”	Logical operation: OR operation (see chapter 3.4.1)
“orn”	Logical operation: NOR operation (see chapter 3.4.1)
“pos”	Absolute positioning in user-defined units
“posf”	Absolute positioning in user-defined units and wait until position has been reached
“r”	Reset boolean operand (see chapter 3.4.1)
“rec_char”	Read character from receive buffer
“rec_var”	Read variable from receive buffer
“ref”	Perform reference movement to limit or reference switch
“reff”	Perform reference movement and wait until reference point has been reached
“ret”	Return to main program
“s”	Set boolean operand (see chapter 3.4.1)
“setipos”1)	Preset position in user-defined units and prepare axes involved in linear interpolation
“snd_str”	Write character string in transmit buffer
“snd_var”	Write variable in transmit buffer
“st”	Transfer operation (see chapter 3.4.1)
“stop”	Stop shaft movement or linear interpolation
“stopa”1)	Stop shaft movements or linear interpolation
“sub”	Arithmetic operation: subtraction
“vel”	Adjust set speed
“wait”	Wait command for sequence program

1) These commands only apply to multi-axis positioning units.

Description of sequence commands

NOTE
The sequence editor commands are described below. The available PLC commands are described in chapter 3.4.1.

acc

Syntax: `acc` `<x><kv>`

The “acc” command is used to enter the maximum acceleration which is used to calculate the selected acceleration and deceleration curve (“ramp” parameter see chapter 3.3). During all subsequent movements, the respective axis x1 to x4 is accelerated or decelerated using this ramp.

Example:

Operator	Operand		Meaning
`ld`	`0`		Load value 0 into CR.
`st`	`x1`		Store CR contents as axis x1 position.
`acc`	`x1`	`125`	Ramp of axis x1 is accelerated or decelerated at a maximum frequency of 125 Hz/ms.
:			

cal

Syntax: `cal` `<l>`

The “cal” command is used to call a subroutine at a predefined label. Subroutines may be nested up to a maximum of three levels.

Example:

Operator	Operand	Meaning
:		
`cal`	`l10`	
:		
`label`	`l10`	Execute subroutine from label l10.
:		
`ret`		Return from subroutine.

div

Syntax: `div` `<kv>`

The “div” command is used to perform division with the CR and an operand.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`div`	`20`	Divide CR contents (value 100) by 20.
`st`	`v4`	Store result as variable v4.
:		

end

Syntax: `end`

The “end” command is used to end the sequence program. The program pointer jumps to the program beginning. The program starts again.

Example:

Operator	Operand	Meaning
`ld`	`0`	Load value 0 into CR.
`st`	`x1`	Store CR contents as axis x1 position.
:		
`end`		Program end. Return to program start.

gearn

Syntax: `gearn` `<x><kv>`

The “gearn” command is used to set the gear ratio denominator for axes x1 to x4 in position following mode. This can be used for implementing an electronic gear, where:

$$\text{Pulse} = \text{Encoder units} \times \frac{\text{gearn}}{\text{gearz}} + \text{goff}$$

The encoder units depend on the “Encoder evaluation E1 or E2” parameter setting. The gear ratio is set with the “gearz” command.

Example:

Operator	Operand		Meaning
`gearn`	`x1`	`3`	Multiply the position value input at the encoder input with the gear ratio 3/4.
`gearz`	`x1`	`4`	
`mode`	`x1`	`1`	Set position following mode via encoder input.

gearz

Syntax: `gearz` `<x><kv>`

The “gearz” command is used to set the gear ratio numerator for axes x1 to x4 in position following mode. This can be used for implementing an electronic gear, where:

$$\text{Pulses} = \text{Encoder units} \times \frac{\text{gearn}}{\text{gearz}} + \text{goff}$$

The encoder units depend on the “Encoder evaluation E1 or E2” parameter setting. The gear ratio is set with the “gearz” command.

Example:

Operator	Operand		Meaning
`gearn`	`x1`	`3`	Multiply the position value input at the encoder input with the gear ratio 3/4.
`gearz`	`x1`	`4`	
`mode`	`x1`	`1`	Set position following mode via encoder input.

goff

Syntax: `goff` `<x><kv>`

The “goff” command is used to adjust axes x1 to x4 with a position value in position following mode (position offset).
The position offset is interpreted as an absolute position and set to zero when changing to position following mode.

If the drive is stopped or another operating mode selected, the incoming pulses (positions) continue to be collected and intermediately stored. After returning to position following mode, these pulses are read in, which results in relative positioning of the drive.

The reference variable (position) is preset in position following mode through an encoder input or a variable (see “mode” command).

Example:

Operator	Operand		Meaning
`gearn`	`x1`	`3`	Set gear ratio 3/4.
`gearz`	`x1`	`4`	
`mode`	`x1`	`1`	Axis x1 set to position following mode via encoder.
`goff`	`x1`	`1000`	A position offset of 1000 pulses is added to the current position value of axis x1 and positioning is carried out.
`goff`	`x1`	`2000`	The shaft moves by another 1000 pulses.

:

handshake

Syntax: `handshake <i><q>`

The “handshake” command can be used for synchronization with a master controller. The application program stops and waits until the synchronizing input is energized (= 1). As soon as the input is energized, the controller continues program execution and the synchronizing output is reset (see chapter 1.11). If the synchronizing input = 0, the synchronizing output is set to 1.

Example:

Operator	Operand		Meaning
:			
`handshake`	`i10`	`q8`	The program is stopped until i10 = 1. When the program is stopped, the output q8 = 1, otherwise it is 0.
:			

linmove

Syntax: `linmove`

The “linmove” command is used to start a previously defined linear interpolation (see “setipos” command) to a relative target position in point-to-point mode using user-defined units. Program execution continues in parallel. The preset position cannot be changed during positioning.

Example:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`1000`	Preset position in user-defined units for axes x1 to x3, involved in interpolation.
`setipos`	`x2`	`2000`	
`setipos`	`x3`	`3000`	
`linmove`			Start linear interpolation to relative target position.
:			

linmovef

Syntax: `linmovef`

The “linmovef” command is used to start a previously defined linear interpolation (see “setipos” command) to a relative target position in point-to-point mode using user-defined units and to wait until the target position has been reached. Program execution is stopped during positioning.

Example:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`1000`	Preset position in user-defined units for axes x1 to x2, involved in interpolation.
`setipos`	`x2`	`2000`	
`linmovef`			Start linear interpolation to relative target position and wait until target position has been reached.
:			

linpos

Syntax: `linpos`

The “linpos” command is used to start a previously defined linear interpolation (see “setipos” command) to an absolute target position in point-to-point mode using user-defined units. Program execution continues in parallel. The preset position cannot be changed during positioning.

Example:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`5000`	Preset position in user-defined units for axes x1 to x3, involved in interpolation.
`setipos`	`x2`	`4000`	
`setipos`	`x3`	`3000`	
`linpos`			Start linear interpolation to absolute target position.
:			

linposf

Syntax: `linposf`

The “linposf” command is used to start a previously defined linear interpolation (see “setipos” command) to an absolute target position in point-to-point mode using user-defined units and to wait until the target position has been reached. Program execution is stopped during positioning.

Example:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`5000`	Preset position in user-defined units for axes x1 to x2, involved in interpolation.
`setipos`	`x2`	`4000`	
`linposf`			Start linear interpolation to absolute target position and wait until target position has been reached.
:			

mode

Syntax: `mode` `<x><k>`

The “mode” command is used to set the axis operating mode. Setting is only possible if the shaft is at a standstill.

0 Point-to-point mode
1 Position following mode through encoder input
The incoming pulses on the encoder input are converted to a motor movement.
2 Position following mode using a variable
The variable value is converted to a motor movement.

NOTE
Any gear ratio must be set before activating position following mode.

In position following mode with variables, the position following variables for axes x1 to x4 are predefined.

Variable	Meaning
v101	Position following variable for axis x1
v102	Position following variable for axis x2
v103	Position following variable for axis x3
v104	Position following variable for axis x4

After switching on, the controller is always set to point-to-point mode. The following examples show axis operating mode programming.

Example 1:

Operator	Operand		Meaning
:			
`ld`	`0`		Load value 0 into CR.
`st`	`x1`		Store CR contents as position value for axis x1.
`mode`	`x1`	`0`	Axis x1 is set to point-to-point mode.
:			

Example 2:

Operator	Operand		Meaning
:			
`gearn`	`x1`	`3`	Set gear
`gearz`	`x1`	`4`	ratio 3/4.
`mode`	`x1`	`1`	Axis x1 is set to position following mode with encoder.
:			

Example 3:

Operator	Operand		Meaning
:			
`gearn`	`x1`	`3`	Set gear
`gearz`	`x1`	`4`	ratio 3/4
`mode`	`x1`	`2`	Axis x1 is set to position following mode with variable.
:			

move

Syntax: `move <x><kv>`

The “move” command is used to preset a relative target position in user-defined units, to start a movement and to calculate a new absolute target position in point-to-point mode. Program execution continues in parallel. The preset position can be changed during positioning. For calculating the new target position, the position value is added to the previous target position.

The respective motor status conditions (positioning, standstill) are stored in flags during positioning of axes x1 to x4.

Flag	**Meaning**	
	1	**0**
m1001	Positioning axis x1	Standstill axis x1
m1002	Positioning axis x2	Standstill axis x2
m1003	Positioning axis x3	Standstill axis x3
m1004	Positioning axis x4	Standstill axis x4

Example:

Operator	Operand		Meaning
:			
`move`	`x1`	`500`	500 user-defined units are preset as relative target position for axis x1 and a new absolute target position is calculated.
:			

movef

Syntax: `movef` `<x><kv>`

In point-to-point mode, the “movef” command is used to preset a relative target position in user-defined units, to start a movement and to wait until the target position has been reached. A new absolute target position is calculated by presetting a relative target position. Program execution is stopped during positioning.
The signal status conditions of flags m1001 to m1004 are the same as for the “move” command.

Example:

Operator	Operand		Meaning
:			
`movef`	`x1`	`1000`	Preset 1000 user-defined units as relative target position for axis x1 and wait until calculated absolute target position has been reached.
:			

mul

Syntax: `mul` `<kv>`

The “mul” command is used to perform multiplication with an operand and the CR.

Example:

Operator	Operand	Meaning
`ld`	`100`	Load value 100 into CR.
`mul`	`20`	Multiply CR contents (value 100) by 20.
`st`	`v3`	Store result as variable v3.
:		

pos

Syntax: `pos` `<x><kv>`

The “pos” command is used to preset an absolute target position in user-defined units and to start positioning in point-to-point mode. Program execution continues in parallel. The preset position can be changed during positioning.

The signal status conditions of flags m1001 to m1004 are the same as for the “move” command.

Example:

Operator	Operand		Meaning
:			
`ld`	`0`		Load value 0 into CR.
`st`	`x1`		Store CR contents as position value for axis x1.
`pos`	`x1`	`1000`	1000 user-defined units are preset as absolute target position for axis x1.
:			

posf

Syntax: `posf` `<x><kv>`

In point-to-point mode, the “posf” command is used to preset an absolute target position in user-defined units and to wait until this target position has been reached. Program execution is stopped during positioning.

The signal status conditions of flags m1001 to m1004 are the same as for the “move” command.

Example:

Operator	Operand		Meaning
:			
`ld`	`0`		Load value 0 into CR.
`st`	`x1`		Store CR contents as position value for axis x1.
`posf`	`x1`	`2000`	Preset 2000 user-defined units as absolute target position for axis x1 and wait until new target position has been reached.
:			

rec_char

Syntax: `rec_char <v>`

The “rec_char” command is used to read a character from the receive buffer.

Example:

Operator	Operand	Meaning
:		
`rec_char`	`v110`	Character received, e.g. “A”. The character’s ASCII value is stored in variable v110, e.g. 65.
:		

rec_var

Syntax: `rec_var <v>`

The “rec_var” command is used to read a character from the receive buffer.

Example:

Operator	Operand	Meaning
:		
`rec_var`	`v110`	Variable v110 received, e.g. +219.
:		

ref

Syntax: `ref <x>`

The “ref” command is used to perform a reference movement to a limit or reference switch. The limit or reference switch is selected by changing the parameter “Type of reference movement” (see chapter 3.3).
A reference movement is only possible in point-to-point mode.
After initiating the reference movement with the “ref” command, the motor moves towards the limit or reference switch at the set speed. Thereafter, it moves in the opposite direction towards the reference point. The speed for this movement is set with the parameter “Clearing speed from limit switch”.

Example:

Operator	Operand	Meaning
:		
`ref`	`x1`	The appropriate type of reference movement is performed.
:		

reff

Syntax: `reff <x>`

The “reff” command is the same as the “ref” command, except that program execution continues only after successful completion of the reference movement.
If an error occurs during the reference movement, program execution is disabled by the reference movement.

Example:

Operator	Operand	Meaning
:		
`reff`	`x1`	Perform a reference movement and wait until a reference position has been reached.
:		

ret

Syntax: `ret`

The “ret” command is used to leave a subroutine.

setipos

Syntax: `setipos <x><kv>`

The “setipos” command is used to prepare linear interpolation with two or three axes in point-to-point mode. The target position must be preset in user-defined units for each axis involved in linear interpolation.
The number of successive “setipos” commands determines whether a 2-axis or 3-axis interpolation is performed.
Interpolation is started using the appropriate command (e.g. “linmove” or “linpos”).
The preset position may not be changed during positioning.

Example 1:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`1000`	Preset positions in user-defined units for axes x1 to x3 involved in interpolation.
`setipos`	`x2`	`2000`	
`setipos`	`x3`	`3000`	
`linmove`			Start linear interpolation to relative target position.
:			

Example 2:

Operator	Operand		Meaning
:			
`setipos`	`x1`	`1000`	Preset positions in user-defined units for axes x1 to x3 involved in linear interpolation.
`setipos`	`x2`	`2000`	
`setipos`	`x3`	`3000`	
`linpos`			Start linear interpolation to absolute target position.
:			

snd_str

Syntax: `snd_str <kv>`

The "snd_str" command is used to write a line of text (up to 40 characters) into the transmit buffer from where it is automatically output to the interface selected with the "Control unit" parameter.
The line of text must be entered using the text editor, e.g.:

```
14: Reference movement is performed
:
```

Example:

Operator	Operand	Meaning
:		
`snd_str`	`14`	The text line 14 "Reference movement is performed" is output through the serial interface.
:		
`ld`	`14`	
`st`	`v1`	
`snd_str`	`v1`	The text line 14 "Reference movement is performed" is output through the serial interface.

snd_var

Syntax: `snd_var` `<v>`

The “snd_var” command is used to write a variable into the transmit buffer.
For this purpose, a variable, e.g. the position following variable v101 of axis x1, is output through the serial interface.

Example:

Operator	Operand	Meaning
:		
`snd_var`	`v101`	Value of position following variable v101 for axis x1, e.g. +219111 is output.
:		

stop

Syntax: `stop` `<x>`

The “stop” command is used to stop a shaft movement or a linear interpolation process. After entering the “stop” command, the drive is disabled and can be re-enabled using the “pos” command.

Example:

Operator	Operand	Meaning
:		
`stop`	`x1`	Stop shaft x1.
:		

stopa

Syntax: `stopa`

The “stopa” command is used to stop all shaft movements or a linear interpolation process. After entering “stopa”, the drives are disabled and can be re-enabled using the “pos” command.
Starting the drive again allows to complete an interrupted linear interpolation and to move to the predefined position.

Example:

Operator	Operand	Meaning
:		
`stopa`		Stop all shaft movements or any linear interpolation.
:		

vel

Syntax: `vel` `<x><kv>`

The “vel” command is used to adjust the set speed of an axis. If no set speed is defined in point-to-point mode, the shaft moves at the “standard speed” entered in the parameter editor.
In point-to-point mode, the set speed can be changed prior to or during positioning. The set speed must always exceed 0 in point-to-point mode.

Example:

Operator	Operand	Meaning
:		
`vel`	`x1 10000`	A set speed of 10000 Hz is defined for axis x1.
:		

wait

Syntax: `wait` `<kv>`

The “wait” command is used to specify a wait time in the sequence program. The time resolution is 1 ms.
When wait = 0, the full processing capacity is assigned to the PLC program until the next 2 ms timing signal. This setting is only reasonable if e.g. a loop for interrogating an input is programmed in the sequence program. The time for interrogating the input may be, for example, 0.01 ms. The remaining 1.99 ms of processor capacity are then available for the PLC program.

Example:1

Operator	Operand	Meaning
:		
`wait`	`10`	The sequence program is stopped for 10 ms.
:		

Example:2

Operator	Operand	Meaning
`label`	`l1`	
`ld`	`i1`	The PLC program is assigned the full processing capacity while input i1 = 0.
`jmpc`	`l2`	
:		
`wait`	`0`	
`jmp`	`l1`	
`label`	`l2`	

3.5.2 List sequence program

List sequence program <L> <↵>

List required section <L1-14> <↵>

```
SEQ edit> l1-14

 1:    ld       0
 2:    st       x1
 3:    gearn    x1      1000
 4:    gearz    x1      1000
 5:    mode     x1      1
 6:    label    l1
 7:    ld       i10
 8:    jmpn     -1
 9:    mode     x1      0
10:    ld       0
11:    st       x1
12:    vel      x1      15000
13:    movef    x1      10000
14:    end

SEQ edit> _
```

Fig. 3-12 Example of a sequence program

3.5.3 Insert line

Insert line, e.g. from line 12 <I12> <↵>

End inserting and editing <ESC>

Example:

```
SEQ edit>I12 <↵>
SEQ edit-I0012>LABEL l2<↵>
SEQ edit-I0013>ADD 100 <↵>
SEQ edit-I0014> <ESC>
```

3.5.4 Delete line(s)

Delete line, e.g. line 5	<D5> <↵>
Delete lines, e.g. lines 1 to 5	<D1-5> <↵>

NOTE
The delete operation is immediately carried out and the sequence program is updated.

Example:

```
SEQ edit>D5 <↵>
```

NOTE
After entering <↵> OK is displayed.

3.5.5 Edit line

Call required line, e.g. line 11	<E11> <↵>
End editing	<ESC>

Example:

```
SEQ edit>E11 <↵>
SEQ edit-0011>VEL X1 15000 <↵>
SEQ edit-0012>VEL X1 20000 <↵>
SEQ edit-0013>MOVEF X1 10000 <↵>
SEQ edit-0014> <ESC>
```

3.6 Text editor

The text editor is used to edit the texts to be output with the corresponding commands in the application program on a display unit (e.g. FT 2000). The line number of the text to be output is specified as a parameter for the OED3 string output command.

Select text editor	<T> <↵>
Call function selection	<H> <↵>

The following lines and sections are shown by way of example.

3.6.1 List text

List text	<L> <↵>
List required section	<L1-14> <↵>

```
TXT edit> l1-14

 1:    Comment
 2:    Speed
 3:    STRING3
 4:    STRING4
 5:    Reference movement
 6:    STRING6
 7:    STRING7
 8:    STRING8
 9:    Point-to-point mode
10:    STRING10
11:    STRING11
12:    STRING12
13:    STRING13
14:    STRING14

TXT edit> _
```

Fig. 3-13 Text examples

3.6.2 Edit text

Call required line, e.g. line 3	<E3> <↵>
End editing	<ESC>

Example:

```
TXT edit>E3
TXT edit-0003>STRING3
TXT edit-0004>Position following mode
TXT edit-0005>Reference movement
TXT edit-0006> <ESC>
```

NOTE
A maximum of 48 lines each comprising 40 characters can be entered.

3.7 Upload function

Using the upload function, all application data (e.g. sequence program, PLC program, text, parameters) are read from the controller, displayed on the screen through the serial interface and stored in a file.
When using the BTERM terminal program, the upload function is performed as follows:

1. The controller is in edit mode.
2. Press the <F8> key.
   → The options “open” and “close” are displayed in a window.
3. Confirm the option “open” with <↵>.
   → The Filename window appears.
4. Enter the required file name and confirm with <↵>.
   → The BTERM terminal program adds the .CAP suffix, e.g. TEST.CAP.
5. Start upload: <U> <↵>
   → All application data consisting of sequence program, PLC program, text and parameters are read from the controller and displayed on the screen.
6. After reading out all application data, press the <F8> key.
7. Confirm the option “close” with <↵>.
   → The application data are stored in the TEST.CAP file.

NOTE
The TEST.CAP file can be processed with any text editor. In addition, it is possible to add comment lines. A comment must be preceded by a semicolon and start at the begining of a line, e.g. ;this is a comment. Comments are not transmitted to the controller with the download function.

3.8 Download function

The download function is used to load all data saved with the upload function into the controller and to perform a validity check.
If an error is detected, the loading operation into the controller is interrupted and the remaining data are discarded.
When using the BTERM terminal program, the download function is performed as follows:

1. The controller is in edit mode.
2. The application data are saved in a file, e.g. TEST.CAP. Rename the extension .CAP to .MAC (see MS-DOS manual for renaming instructions).
3. Start download: <D> <↵>
4. Press the <F6> key.
   → The options “open”, “close” and “execute” are displayed in a window.
5. Confirm the option “execute” with <↵>.
   → A window comprising the .MAC files appears on the screen.
6. Select the required file, e.g. TEST.MAC.
   → The application data are loaded into the controller and submitted to a validity check.

3.9 End edit mode

Leave the OED3 programming interface with <X> <↵>.

ATTENTION
The entered application program is immediately executed after leaving edit mode.

4 Error messages

The following errors may occur during program creation and execution and are output through the serial interface.

Error type/error number	**Cause/rectification**
Syntax error during program creation [E4291] [E4295] [E4296] [E4297] [E4299] [E4300] [E4302]	Syntax errors are immmediately displayed on the screen and must be corrected. This ensures that there are no syntax errors in the program after leaving the editor. Command not implemented Illegal command in PCL program Invalid operand Unknown command OED3 system error Numerical input error Data conflict when calling OED3
Parameterizing error [E4290] [E4292] [E4293] [E4294] [E4298]	The input value range is immediately checked. If an error occurs, the range limits are displayed and new input values expected. Incomplete OED3 parameterizing Incorrect parameters Insuffficient number of parameters Excessive number of parameters Parameterizing error
Error during data transfer (down-load) to controller	The download function checks the validity of any line transferred to the controller which means that the paramater range and the program line syntax are checked. The data transfer to the controller is interrupted if an error occurs. All data sent after the error are discarded. Acknowledge the error with <ESC>.
Error during program execution (see following pages for error output)	Errors occuring during program execution are displayed by a flashing number on the status display 40 and stored in the controller error memory (see controller manual). In addition, the error is displayed on the screen by the controller (see figure 4-1). System errors (fatal errors) occuring in the controller result in controller reset and are only displayed on the status display 40 (see controller manual).

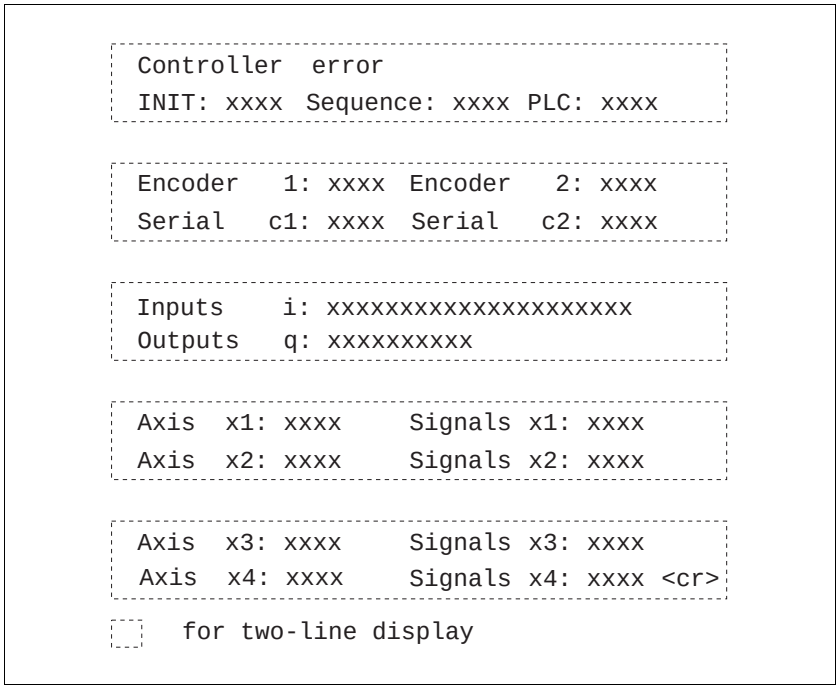


Bild 4-1 Error display on the screen

The following table explains the error memory contents and the meaning of the error display which is evident in fig. 4-1.

If several errors of the same error group occur at the same time, the sum of the error numbers is output.

Refer to the controller manual for information on eliminating faults.

Error display	**Meaning**
Controller error	INIT: Line number in INIT program
	Sequence: Line number in sequence program
	PLC: Line number in PLC program
Encoder 1 or 2	
16#0100	Contouring error
16#0200	Encoder error
Serial c1 or c2	
16#0002	Attempt to address non-initialized interface
16#0010	Hardware interface already assigned
16#0020	Error upon buffer allocation or deallocation
16#0040	Insufficient receive string buffer
16#0080	Transfer string too long
16#0100	Invalid character sequence in transfer string
16#0800	Receive buffer overflow
16#1000	Overrun: Excessive baud rate
16#2000	Parity: Transmission error
16#4000	Framing: Interface parameter error
16#7FFF	Break: Cable interrupted or disconnected

Error display	Meaning
Axis x1 to x4	
16#0001	One or more signals active, see “Signals x1 to x4”
16#0004	Invalid command in position following mode
16#0008	Wait positioning enabled (e.g. “posf”, “movef”)
16#0010	Invalid command for interrupted/blocked axis
16#0020	Master curve error
16#0040	Insufficient information on reference variable
16#0080	Invalid command during shaft movement
16#0100	Invalid command during reference movement
16#0400	Incorrect parameter value
16#0800	A value cannnot be calculated
16#1000	Invalid parameter value passed
16#2000	Command cannot be executed under these conditions
16#4000	Value not defined
16#7FFF	Resource not ready
Signals x1 to x4	
16#0001	Positive hardware limit switch
16#0002	Negative hardware limit switch
16#0004	Reference switch
16#0008	Hardware stop
16#0010	Hardware trigger
16#0020	Positive software limit switch
16#0040	Negative software limit switch
16#0080	Software stop
16#0100	Contouring error
16#0200	Encoder error
16#0400	Power controller not ready
16#0800	Power controller overtemperature
16#1000	Motor overtemperature

NOTE
The errors stored in the error memory are cleared after controller reset or application program restart.

NOTE
Depending on the parameter “control unit”, the error display may comprise two or more lines, see chapter 1.10.

5 Appendix

5.1 Glossary

Accumulator

A controller register which is used for storage of intermediate results. The contents of the accumulator is also called current result (CR).

Arithmetic operations

The operators “add”, “sub”, “mul”, “div” can be used for arithmetic operations with an operand and the CR.

Current result CR

The current result CR is an intermediate storage feature (accumulator) of the controller which is used for arithmetic and logical operations and data transfer. In contrast to conventional programmable logic controllers, a BERGER LAHR series 300 controller has only one accumulator for the CR. The memory capacity of the accumulator is dynamically adapted to the data type of the operand.

Drive units

Drive units are controller-specific processing units for positions, speed and acceleration values.

Electronic gear

In position following mode, the reference variable (e.g. encoder) can be used to implement a speed-transforming gear or reduction gear using a gear ratio. This is also called an electronic gear, where:

Drive units = Reference variable x Gear ratio

Encoder

Sensor for position detection (actual position detection) of a motor.

Encoder evaluation

This parameter is used for setting the encoder resolution (500 or 1000 marks) and the internal evaluation factor (single, dual, quadruple), where:

$$\text{Motor revolutions} = \text{Encoder pulses} \times \frac{\text{Evaluation factor}}{\text{Encoder resolution}}$$

Encoder programming

Each encoder input can be used for reference variable input (in position following mode) or for rotation monitoring.

Flag

Flags are storage elements which can be used to indicate certain states. The controller has a defined memory area for flags.

Inputs/outputs

The controller is provided with a fixed number of inputs/outputs which are used for sequential operations. Input/output processing can be carried out in parallel to movement control.

Interpolation

A coordinated simultaneous movement of several shafts or axes (at least two).

Logical operations

The operators “and” and “or” are used to perform logical operations with one or more boolean operands and the CR.

Normalizing factor

The normalizing factor is used to convert user-defined units (e.g. cm) to drive units (steps or increments).

Operands

Many instructions require an operand with which to perform the operation.

Point-to-point mode

In point-to-point mode, a positioning command is used for positioning from a given point A to a specified point B. Positioning can be absolute (based on the reference position of the shaft) or relative (based on the current position of the shaft).

Position following mode

In position following mode, target positions are specified through an encoder input or a variable.
A gear ratio can be applied to the position. Thus, an electronic gear can be implemented.

Process image PI

The process image is an intermediate storage feature for the inputs/outputs.

Reference movement

The “ref” and “reff” commands can be used to execute reference movements. Reference movements are only possible in point-to-point mode. With a reference movement, a reference point is addressed which constitutes the zero point for the system of dimensions.

Relational operations

The “eq”, “gt” and “lt” commands can be used for comparative evaluations of an operand and the CR. After the operation, the CR has always the BOOL data type and contains one of the following values:
0 or FALSE, if the comparison is false or
1 or TRUE, if the comparison is true.

Set and reset

The set command “s” and the reset command “r” can be used for setting (1) or resetting (0) a boolean variable (e.g. output or flag). This can be performed with reference to the current result.

Setpoint

In point-to-point mode, setpoints are specified and a movement is initiated using the “pos(f)” or “move(f)” command.

Transfer operation

The transfer commands “ld” and “st” are used for loading values into the CR and for storing them from the CR into variables. When loading (“ld”), the CR takes the data type of the loaded value. When storing (“st”), the data types of the CR and operand values must be compatible.

User-defined units

User-defined units are processing units which can be freely defined by the user, where:
Drive units = User-defined units x Normalizing factor
(drive units in steps or increments, user-defined units e.g. in cm).
Positions, speed and acceleration values are always specified as user-defined units.

Variables

Variables are storage elements which are used in a program for data transfer and data storage.

5.2 Abbreviations

AC	Alternating current
add	Addition command
and	AND operation
ASCII	American Standard Code for Information Interchange
B	Byte
b	Boolean variable
c1	Serial interface 1
c2	Serial interface 2
cal	Call subroutine
CR	Current result
DC	Direct current
div	Division command
Doc. no.	Documentation number
E	Encoder
eq	Relational operation “equal”
gt	Relational operation “greater than”
Hz	Hertz
i	Input
jmp	Jump to label

k	Value
l	Label
ld	Load
LIMN	Negative limit switch
LIMP	Positive limit switch
lt	Relational operator “less than”
m	Flag
M	Motor
ms	Millisecond
mul	Multiplication command
N	Number of encoder marks
or	OR operation
PC	Personal computer
PI	Process image
PID	Proportional, integral, differential controller
PLC	Programmable logic controller
q	Output
r	Reset
ref	Reference movement
ret	Return from subroutine

s	Set
st	Store
Stop	Stop signal
sub	Subtraction command
t	Timer
TRIG	Trigger signal
v	Variable
x	Axis number

6 Index

	Proposals **Improvements**
BERGER LAHR GmbH Breslauer Str. 7 Postfach 1180 D-77901 Lahr	**OED3** Edition: April 94 Doc. no. 212.954/DGB 04.94
Sender: Name: Company/department: Address: Telephone no.:	Please inform us, using this form, if you have discovered any errors when reading this document. We should also appreciate any new ideas and proposals.

Proposals and/or improvements: