Programming Software
for Series 300 Units and
WDP3-014/018

ProOED 3
Version 3

Doc. no. 214.143/DGB

Ident. no.: 00441110871

Software version: 3.2XX

Edition: c220 09.2003

	Proposals **Improvements**
Berger Lahr GmbH & Co. KG Breslauer Str. 7 Postfach 1180 D-77901 Lahr	**ProOED3** Edition: b162 July 97 Doc. no. 214.143/DGB 07.97
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.

Proposal and/or improvements:

Table of contents

Table of contents

Table of contents

Table of contents

Table of contents

1 General description

1.1 Functionality of ProOED3

Purpose

The ProOED3 programming software has been designed for creating control programs for BERGER LAHR Series 300 controllers (e.g. WDP3-318) and WDP3-014/018 controllers with the operating system software OED3 Version 3.0 installed.

NOTE
The Series 300 controllers and the WDP3-014/018 controllers differ with respect to the hardware and the functionality (see chapter 1.2). The differences with regard to front panel operation, parameters and commands are mentioned in the individual chapters.

Controllers with OED3

Series 300 controllers with OED3:
- WP-311 (single-axis control unit without power controller)
- WDP3-314/318 (single-axis control unit with power controller)
- WDPM3-314 (two-axis control unit with power controller)
- WPM-311 (four-axis control unit without power controller)

WDP3-01X controller with OED3:
- WDP3-014/018 (single-axis control unit without power controller)

Functions

The most important functions of the ProOED3 programming software are:
- Parameter setting and controller programming
- Loading programs into the controller
- Debugging
- Testing the input and output wiring of the controller
- Easy programming in manual and teach-in mode using the sample programs supplied

The control programs are created using comfortable editors on a PC. The pull-down menue and window architecture allows you to work conveniently. The ProOED3 installation diskette includes comprehensive sample projects which may be used for programming your applications. ProOED3 offers comfortable options for debugging, e.g. viewing program execution while the program runs on the controller.

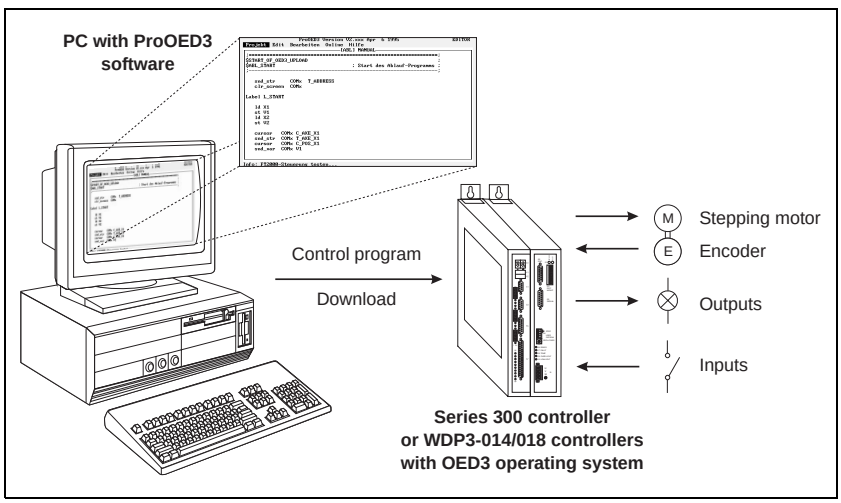


Fig. 1-1 System environment

1.2 Comparison of WDP3-01X and Series 300 controllers with OED3

Differences	Effects
Indexer The WDP3-01X controller comprises a software indexer. Series 300 controllers comprise a hardware indexer.	Processing an instruction with the WDP3-01X takes up to three times longer than with a Series 300 controller.
Main memory The WDP3-01X controller does not have a battery-buffered RAM. Series 300 controllers have a battery-buffered RAM installed.	On Series 300 controllers, the application program created with ProOED3 is retained in the main memory (RAM) after power-off. On WDP3-01X controllers, the application program is deleted after power-off. Saving in EEPROM: The application program can be saved in the controller's EEPROM with ProOED3. An application program saved to the EEPROM is automatically loaded into the controller's main memory after power-on and is retained after power-off.
Inputs/outputs The WDP3-01X controller has 11 freely assignable inputs and 4 freely assignable outputs. A Series 300 single-axis controller has 15 freely assignable inputs and 10 freely assignable outputs. I/O extension is not possible on the WDP3-01X. You can connect up to 5 MP 926 I/O extensions (with 16 I/O's) to a Series 300 controller.	The WDP3-01X controller has only a limited set of I/O options as compared to the Series 300 controller.
Trigger input The WDP3-01X does not feature a trigger input. Series 300 controllers are equipped with a trigger input.	The trigger input on Series 300 controllers can be used for responding very quickly to events; see "settrigger" command.
Analog interface The WDP3-01X controller cannot be equipped with an analog interface.	Series 300 controllers can be used for input or output of analog voltage signals via an analog interface.
Encoder interface Up to one encoder interface can be installed on the WDP3-01X, up to two encoder interfaces on Series 300 controllers.	One encoder interface can be used for implementing either an electronic gear or rotation monitoring. Two encoder interfaces can be used for implementing both an electronic gear and rotation monitoring.

Differences	Effects
Serial interface The WDP3-01X has only one serial interface. Series 300 controllers can be equipped with two serial interfaces.	The WDP3-01X controller allows only limited testing of programmed operator dialogs for a terminal (e.g. FT 2000); this is effected by terminal simulation in ProOED3 debugging mode.
Current setting On the WDP3-01X, the maximum phase current is set with a front panel parameter; on Series 300 controllers, it is set via a rotary switch on the front panel.	The current setting made with the OED3 command “setcurrent” refers to the value set via the front panel.
Number of axes The WDP3-01X is a single-axis controller. Series 300 controllers are available as single-axis, two-axis and four-axis control units for 3-phase or 5-phase motors.	You can control a complete handling system using a Series 300 multi-axis controller.
Lauer operating panel An operating panel made by Lauer can be connected to the Series 300 controllers.	Applications can be created for Series 300 controllers with a Lauer operating panel.

NOTE
The commands and control parameters for Series 300 controllers and WDP3-01X controllers differ in part (see chapters 6.1.2 and 6.1.3.2).

1.3 An overview of the ProOED3 programming software

1.3.1 Program editors

Program editors for freely creating and editing the control programs are provided. Comments can be entered in order to enhance the readability of the programs.

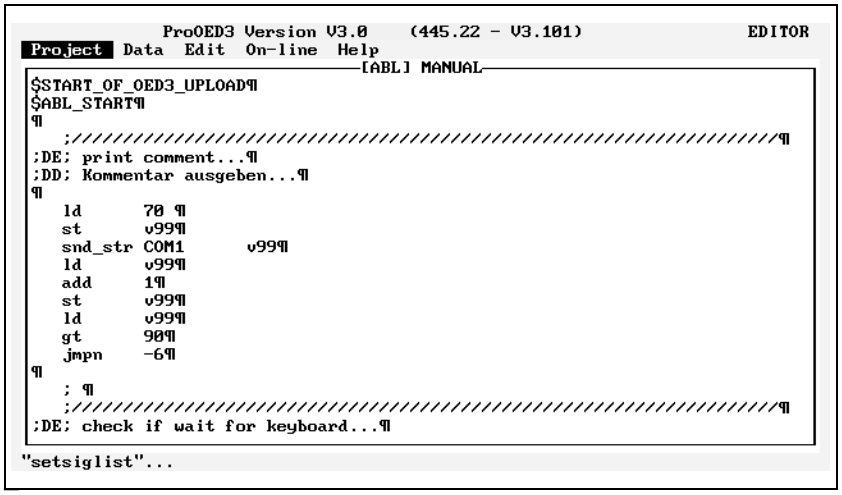


Fig. 1-2 Program editor

– Functions such as copying, finding and replacing facilitate programming.

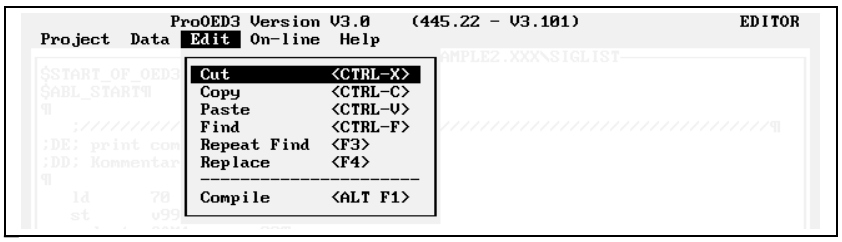


Fig. 1-3 Editing functions

1.3.2 Parameters and variables

Predefined controller parameters can be modified with an editor.

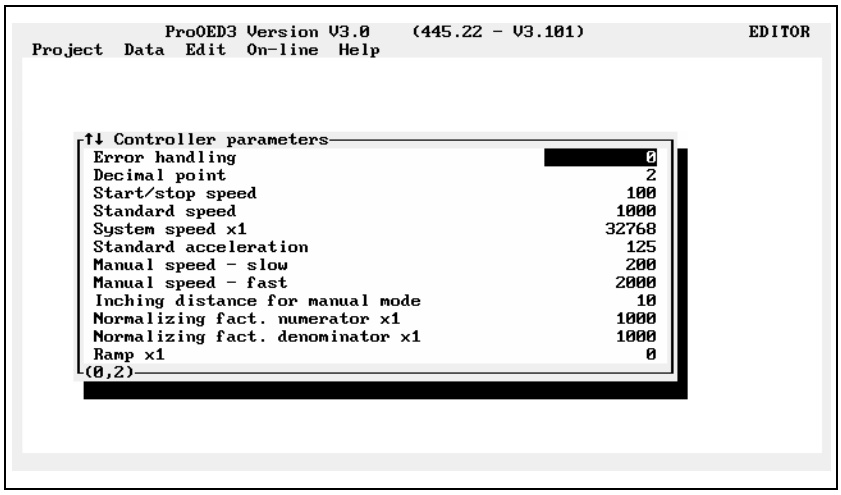


Fig. 1-4 Editing control parameters

Editors are also provided for position variables, text variables and symbolic names.

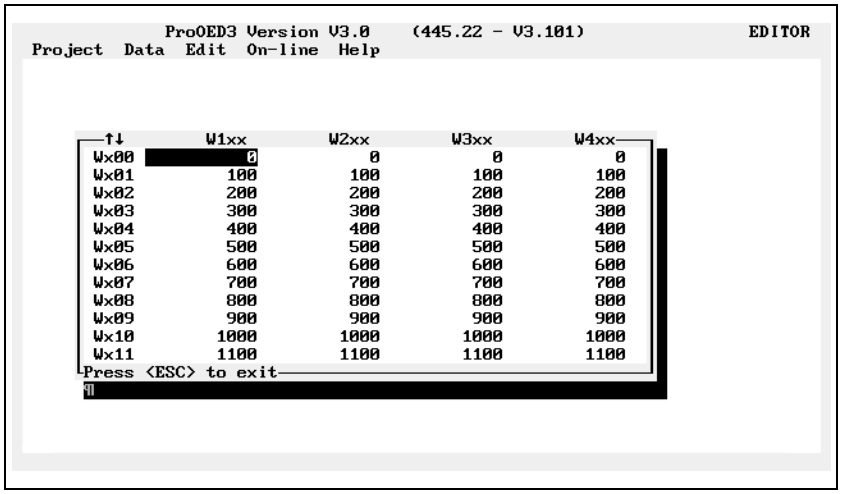


Fig. 1-5 Editing position variables

1.3.3 On-line functions

In the “On-line” menue, options for data transmission and debugging are available.

A few examples:

– Transfer a program into the controller
– Display the input states on screen
– Manipulating the output states

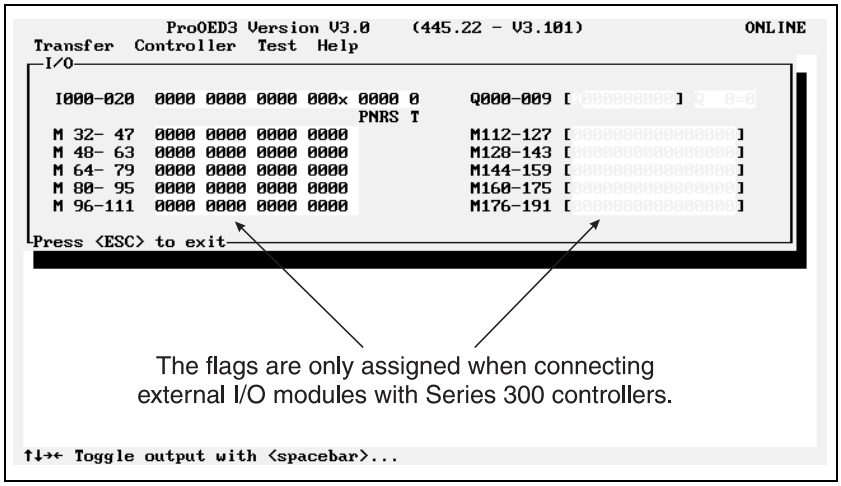


Fig. 1-6 Inputs, outputs and flags

– Manual movement and positioning of axes as well as reading in position values.

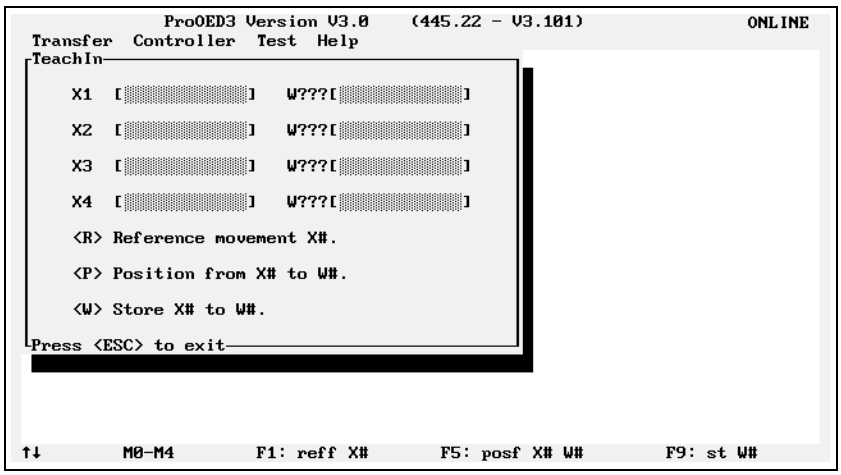


Fig. 1-7 Teach-in

– Viewing the control program and its current results during program execution

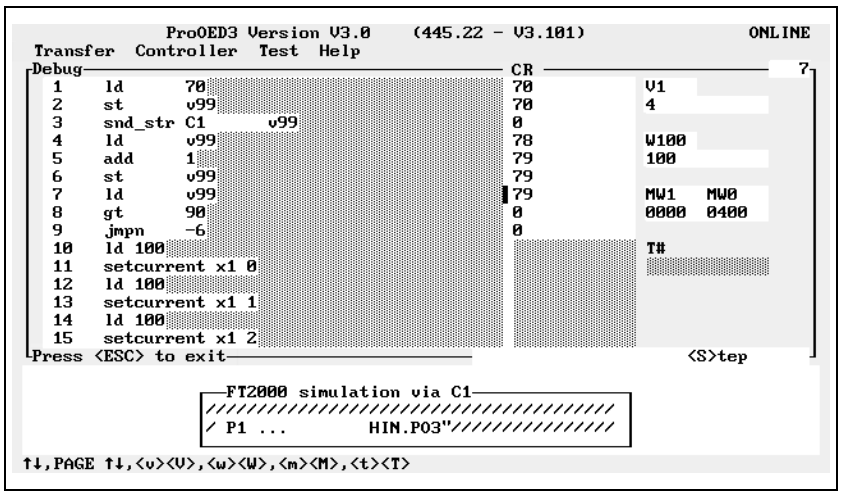


Fig. 1-8 Viewing a control program

– Simulation of the FT2000 operating terminal

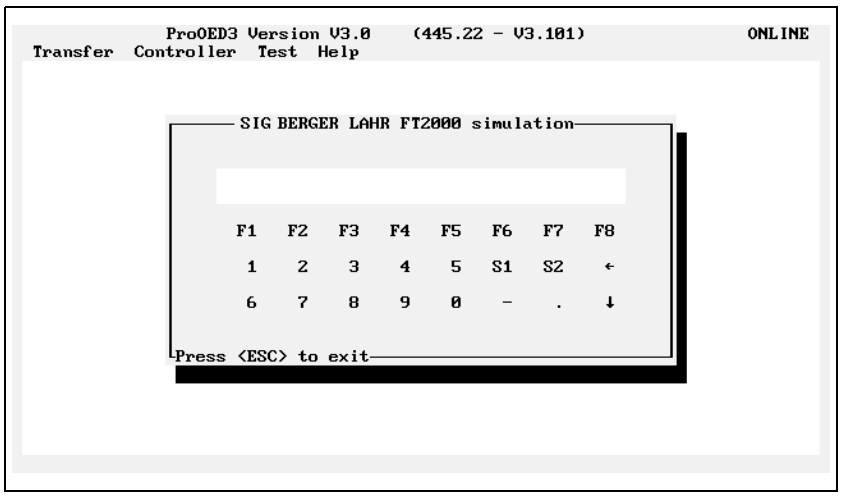


Fig. 1-9 FT2000 simulation

1.4 Documentation

This documentation contains information on installing and using the ProOED3 programming software and on programming a Series 300 or WDP3-014/018 controller using ProOED3.

The controller manual contains information on installing and operating the unit.

NOTE
The screens shown in this manual are samples which do not necessarily have to be identical with those appearing on your monitor.

1.5 Menue structure and operating steps

The following figure shows the menue structure of ProOED3.

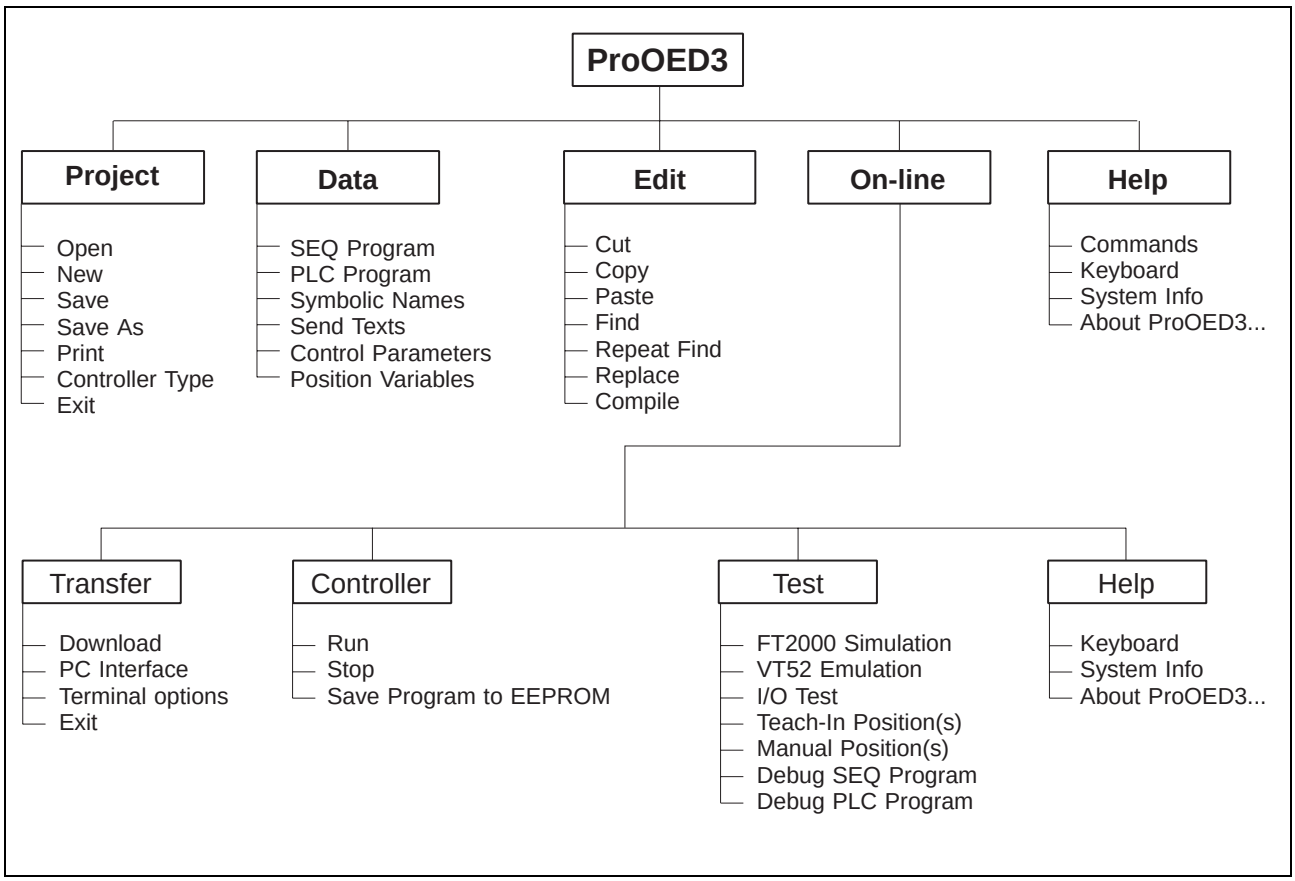
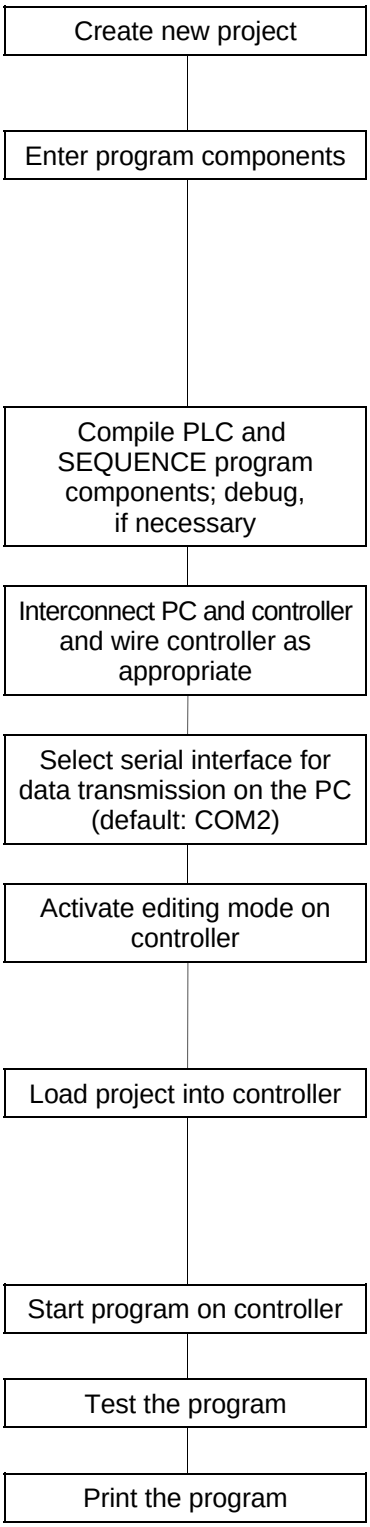


Fig. 1-10 Menue structure of ProOED3

The operating steps described in the following chapters illustrate the procedures involved in ProOED3 programming and operation.
For details on operation, refer to the appropriate chapters with the individual menue descriptions.

1.5.1 Creating and testing a new project



⇒ Menue “Project/New”. Specify a project name and the controller type. The controller type can be changed in the “Project/Controller Type” menue.

⇒ Menue “Data/...”
* “SEQ Program” (sequence) for movement sequences, communication via serial interface, I/O operations
* “PLC Program” for parallel processing of I/O operations and SEQUENCE program component
* “Symbolic Names” for individual designations in the program
* “Send Texts” for communication via serial interface
* “Control Parameters” for the basic settings of the controller
* “Position Variables” for teach-in mode

⇒ Menue “Edit/Compile”

⇒ Check the I/O wiring; see chapter 3

⇒ Menue “On-line/Transfer/PC interface”

⇒ Start OED3 from the controller front panel. On Series 300 controllers, keep the keys 41 and 42 pressed on “+” side until “Ed” appears in the display. On WDP3-014/018 controllers, press the “+” key and the “↵” key simultaneously to display “Ed”. If the communication between PC and controller is o.k., “EDIT>” is displayed on the PC screen.

⇒ Menue “On-line/Transfer/Download”

NOTE
If a program component has not been compiled, a “?” precedes the program component name when opening the menue “.../Download”.

⇒ Menue “On-line/Controller/Run”

⇒ For the menues “On-line/Test/...”, see chapters 1.5.6.

⇒ Menue “Project/Print”

NOTE
You can save the control program to the EEPROM by selecting “On-line/Controller/Save Program to EEPROM” from the menue. On WDP3-014/018 controllers, it is indispensable to save the program to the EEPROM since they do not have a battery.

1.5.2 Loading an existing project into the controller

Open existing project	⇒ Menue “Project/Open”
Specify controller type	⇒ Menue “Project/Controller Type”
Compile PLC and SEQUENCE program components; debug, if necessary	⇒ Menue “Edit/Compile”
Interconnect PC and controller and wire controller as appropriate	⇒ Check the I/O wiring; see chapter 3
Select serial interface for data transmission on the PC (default: COM2)	⇒ Menue “On-line/Transfer/PC interface”
Activate editing mode on controller	⇒ Start OED3 from the controller front panel. On Series 300 controllers, keep the keys 41 and 42 pressed on “+” side until “Ed” appears in the display. On WDP3-014/018 controllers, press the “+” key and the “↵” key simultaneously to display “Ed”. If the communication between PC and controller is o.k., “EDIT>” is displayed on the PC screen.
Load project into controller	⇒ Menue “On-line/Transfer/Download”
Start program on controller	⇒ Menue “On-line/Controller/Run”

NOTE
If a program component has not been compiled, a “?” precedes the program component name when opening the menue “.../Download”.

NOTE
You can save the control program to the EEPROM by selecting “On-line/Controller/Save Program to EEPROM” from the menue.
On WDP3-014/018 controllers, it is indispensable to save the program to the EEPROM since they do not have a battery.

1.5.3 Displaying positions during manual movement

A manual movement can be executed from the front panel (see controller manual) or under application program control. During a manual movement under application program control, the current position can be displayed on the screen during the movement. A few sample programs for manual movements are included when installing ProOED3 on the PC. The manual movement under application program control is effected with the controller flags m0 to m4. In the sample program “MANUAL.PO3”, the signal inputs i0 to i4 are assigned to the flags m0 to m4.

Open project for manual movement and load it into the controller

⇒ See chapter 1.5.2

Operate manual inputs and display manual positions

⇒ Menue “On-line/Test/Manual Position(s)”

NOTE
You may want to modify the parameters “Manual speed – slow” and “Manual speed – fast” in the “Control Parameters” editor and load them into the controller.

1.5.4 Teaching positions into the controller

In teach-in mode under application program control, a maximum of 400 positions per axis can be approached manually and stored in the controller.
A few sample programs for teach-in are included when installing ProOED3 on the PC. The manual movement is effected with the controller flags m0 to m4. In the sample program “TEACH-IN.PO3”, the signal inputs i0 to i4 are assigned to the flags m0 to m4.

Open project for teach-in and load into controller

⇒ See chapter 1.5.2

Perform manual movement and store the desired position

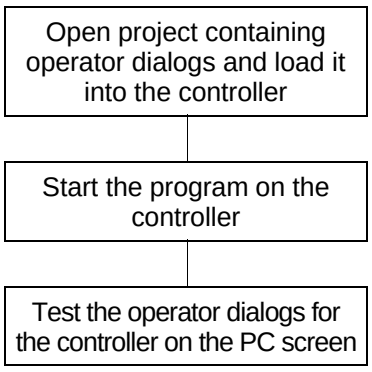
⇒ Menue “On-line/Test/Teach-In Position(s)”

NOTE
The stored positions can be edited later with the “Position Variables” editor.

1.5.5 Testing the operator dialogs for VT52 or FT2000 Terminal

You can use ProOED3 for emulating VT52 or FT2000 terminals. This allows you to test programmed operator dialogs without having connected the respective terminal to the controller.
Operator dialogs are programmed in the SEQUENCE program component. The output texts for operator dialogs are created with the “Send Texts” editor. A few sample programs for operator dialogs are included when installing ProOED3 on the PC.
If you want to test operator dialogs without an operating terminal installed, you can connect one or two PCs with ProOED3 to the controller (when using two PCs, you can view the program in addition; see chapter 1.5.6).

NOTE
If the controller has only one serial interface, the programmed operating interface of the controller must be the same as the PC interface (standard PC interface: “c1”). It may be necessary to assign “c1” to the name of the operating interface in the “Symbolic Names” editor.



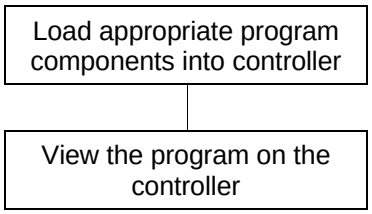
⇒ See chapter 1.5.2

⇒ Menue “On-line/Controller/Run”

⇒ Menue “On-line/Test/FT2000 Simulation” or “VT52 Emulation”

1.5.6 Viewing the SEQUENCE or PLC program component

The flags and variables used in the SEQUENCE or PLC program component can be displayed on the PC during program execution on the controller. The program can be executed step by step.



⇒ See chapter 1.5.2

⇒ Menue “On-line/Test/Debug SEQ Program” or menue “On-line/Test/Debug PLC Program”

2 Installation

2.1 Scope of supply

The delivery must be checked for completeness.

The scope of supply comprises:

Qty.	Designation
1	3½" diskette with ProOED3
1	ProOED3 documentation

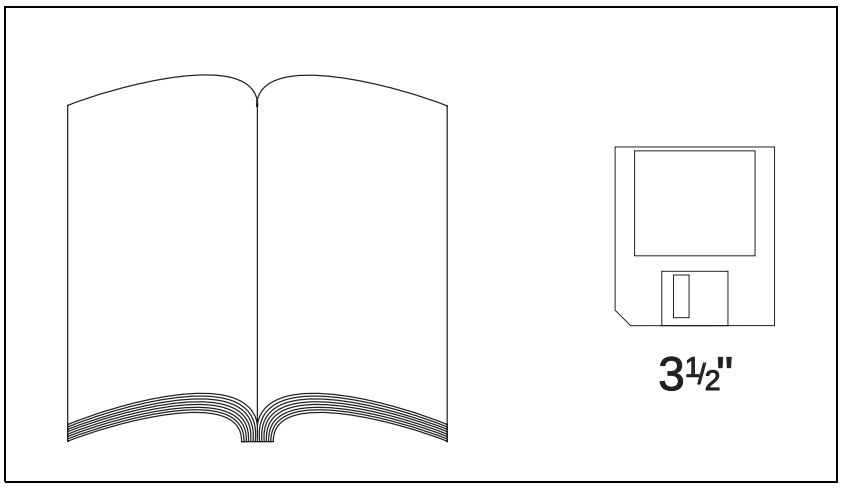


Fig. 2-1 Scope of supply

2.2 Accessories

The following accessories are available and should be ordered separately (for a description of the accessories, see controller manual, chapter 6.2):

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

2.3 System requirements

Hardware and software requirements for ProOED3:

- IBM or 100% compatible PC AT
- MS-DOS operating system version 6.0 or later
- At least 555 Kb memory available
- At least 1 diskette drive, 3½", 1.44 MB
- Serial interface RS 232 or RS 485
- Correctly wired interface cables for communication with the controller
- RS 485/RS 232 interface converter, if applicable (e.g. MP 923)
- Series 300 or WDP3-014/018 controller with OED3

2.4 Software installation

To install ProOED3, the following steps are required:

1. Create a backup copy of all diskettes included in the package (for instructions on copying diskettes, refer to your DOS manual).
2. Insert the 3½" ProOED3 program diskette into the appropriate drive.
3. Select the same drive, e.g.:

   A:

   by entering it at the DOS prompt and pressing <↵>.
4. Start the installation program by entering:

   INSTALL

   and pressing the <↵> key.

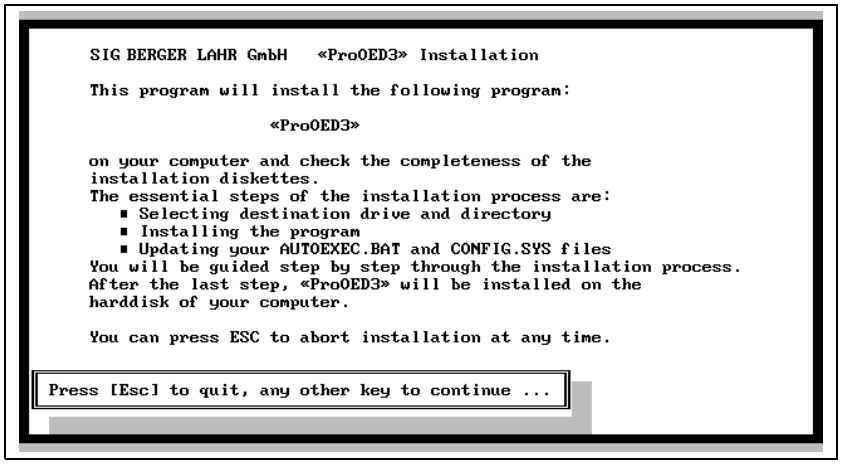


Fig. 2-2 ProOED3 installation

A few messages regarding the installation process will be displayed on the screen. Continue by pressing any key or exit the installation program by pressing <Esc>.

NOTE
If you should have insufficient DOS memory available, a message is displayed on the screen. In this case you should exit the installation program and enhance your memory configuration; see MS-DOS manual.

5. Acknowledge the system configuration screen by pressing any key.

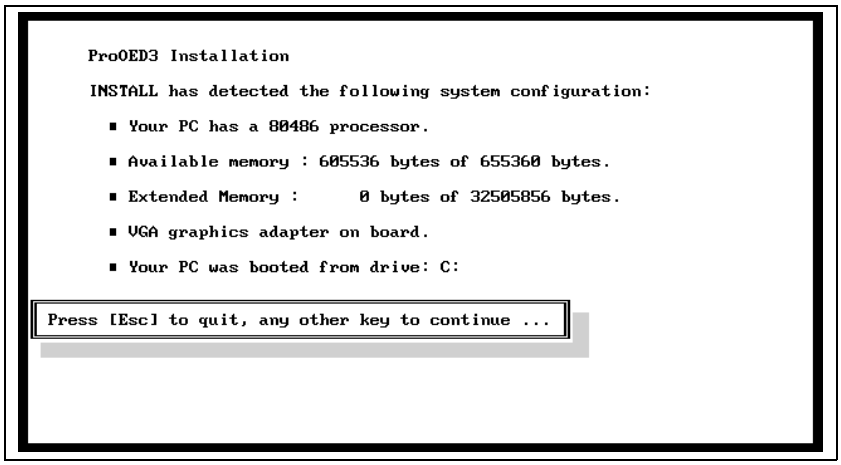


Fig. 2-3 System configuration display

6. Select the controller type to be used and acknowledge by pressing the <↵> key.

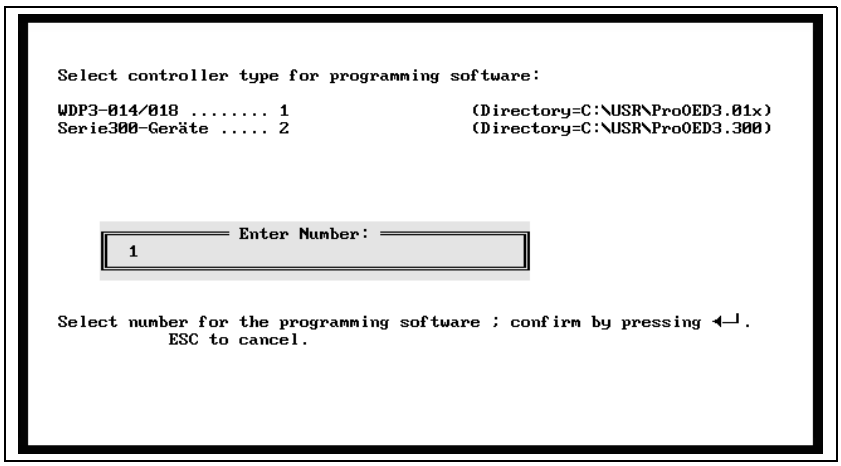


Fig. 2-4 Select controller type

7. You are prompted whether to create/modify your AUTOEXEC.BAT and CONFIG.SYS files automatically; confirm by pressing the <Y> (“YES”) key (recommended). If any changes are required, the installation program saves the original files by renaming them to AUTOEXEC.BAK and CONFIG.BAK.

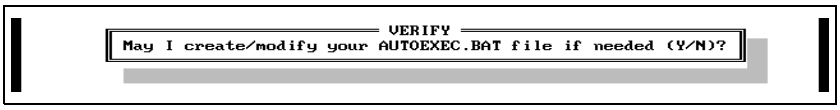


Fig. 2-5 Update AUTOEXEC.BAT?

The following directories may have been created on your hard disk, depending on which controller type you selected in step 6:

ProOED3 directory for WDP3-014/018 controllers:

C:\USR\PROOED3.01X
- \PROJECTS
- \EXAMPLES

ProOED3 directory for Series 300 controllers:

C:\USR\PROOED3.300
- \PROJECTS
- \EXAMPLES

The “PROJECTS” subdirectory is used for storing the project data, the “EXAMPLES” subdirectory contains sample projects for the corresponding controller.

NOTE
The files with the extensions “OED” and “INI” are configuration files and must never be modified or deleted.

NOTE
For a description of the sample projects, refer to chapter 8.2.3.

2.5 Starting ProOED3

The prerequisite for starting ProOED3 is the correct installation of the software as described in chapter 2.4.

Use the "START" command to start the ProOED3 programming software.

Command description

`START <PROJECT>.PO3 <COMX> </MONO>`

Parameter	Description
<COMX>	Specify the PC interface. This may be "COM1" or "COM2", as appropriate for the interface used.
<PROJECT>.PO3	Specify the project to be loaded after starting ProOED3. The extension ".PO3" *must* be specified.
</MONO>	Parameter for monochrome display screens.

NOTE
The START command is a batch file. It is stored in the ProOED3 project directory called "...\PROJECTS".
The parameters can be specified in any order.

Example:

1. Switch on your PC.
2. Change to the ProOED3 project directory, e.g.:

   CD C:\USR\PROOED3.01X\PROJECTS

   or

   CD C:\USR\PROOED3.300\PROJECTS

   and press the <↵> key.

NOTE
You may omit this step if the ProOED3 project directory is included in your DOS PATH specification.

3. Enter the following command:

   START MANUAL.PO3 COM2

   and press the <↵> key (PC interface COM2 and project "MANUAL.PO3").

4. ProOED3 comes up with the following screen. If you do not specify a project, either the last project edited is loaded or an empty project is displayed (when starting ProOED3 for the first time).

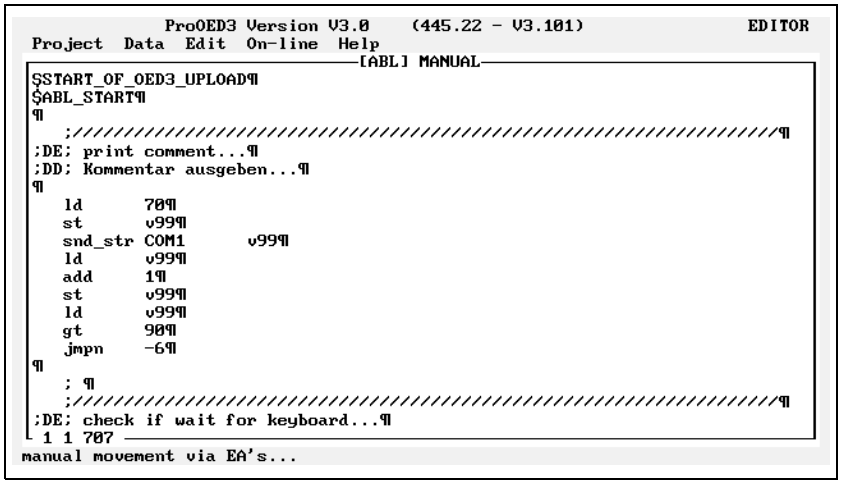


Fig. 2-6 ProOED3 after program start

5. Exit ProOED3 by pressing <Alt>-<F4>.

2.6 Converting a project from ProOED3 version 2 to version 3

If a project which had been created with the ProOED3 version 2 programming software is to be loaded on a controller with the OED3 version 3 operating system software, it must be converted to a ProOED3 version 3 project. To do this, proceed as follows:

1. Copy the ProOED3 version 2 project files to the ProOED3 version 3 project directory.

   Example:

   COPY C:\PROOED3.200\PROJECTS\PROJECT1.X C:\USR\PROOED3.300\PROJECTS

2. Start the ProOED3 version 3 programming software and select the ProOED3 version 2 project using the "Project/Open" menue option.

NOTE
A dialog window appears which prompts you whether to convert the existing ProOED3 version 2 project to a version 3 project.

3: If yes, enter a new project name.

NOTE
The project is converted and saved in the ProOED3 version 3 project directory.

4. Check the control parameter settings (see chapter 6.1.2) and load the project into the controller using the "On-line/Transfer/Download" menue option; see the chapter on operation.

NOTE
When having tested the ProOED3 version 3 project successfully on the controller, you may delete the old version 2 project from the ProOED3 version 2 project directory.

3 Setting up a controller with ProOED3

3.1 Wiring the controller

The following components must be wired in accordance with chapter 2 of the controller manual:

– Inputs and outputs
– Limit switches
– Stepping motor
– Encoder (if required)
– Serial interfaces

3.1.1 Serial interfaces

The communication between the PC and the controller is effected by serial data transmission. The default interface is COM2 on the PC and c1 on the controller (adapter slot 51 for Series 300, adapter slot OPT1 for WDP3-014/018). Figure 3-1 illustrates the connection diagram which applies between controller and PC.

ATTENTION
When wiring, take into account 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.

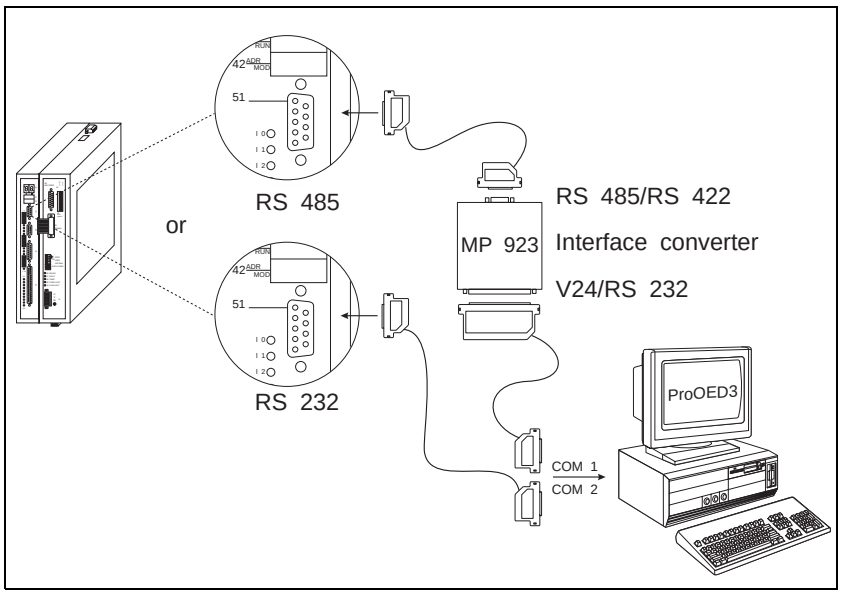


Fig. 3-1 Connection diagram, e.g. for WDP3-318 controller

The interface cable connectors must be wired as follows:

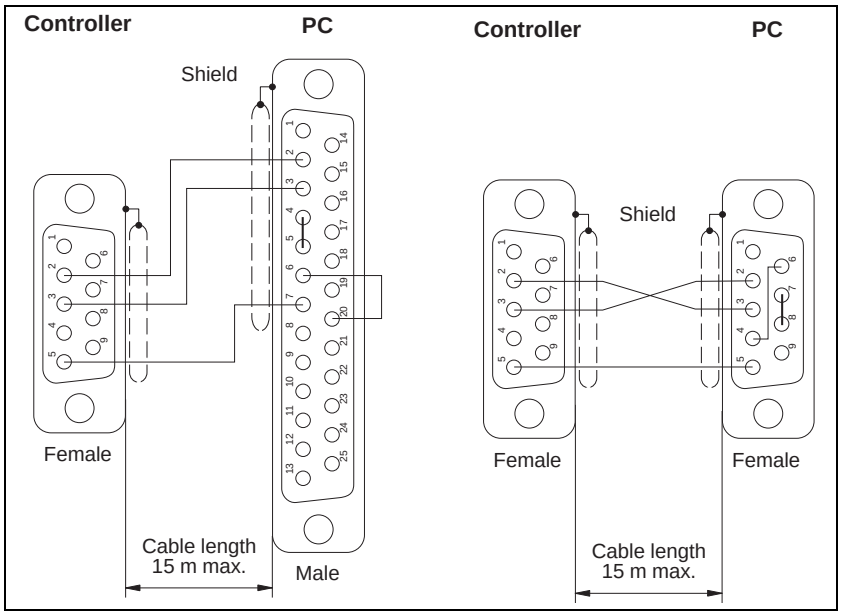


Fig. 3-2 Connector wiring

NOTE
Details on the interface assignment and information on wiring are described in the controller manual (chapters 2 and 6).

3.2 Testing the wiring

The wiring can be checked as follows:

– Check the communication between the controller and the PC
– Carry out an I/O test

For this purpose, start ProOED3 with the “START” command and the “TEST” parameter.

Command description

START /TEST <COMX> </MONO>

Parameter	Description
<COMX>	Specify the PC interface. This may be “COM1” or “COM2” as appropriate for the interface used.
</TEST>	This invokes the I/O test immediately. This option is used for testing the input/output wiring. You cannot edit any projects in this mode.
</MONO>	Parameter for monochrome display screens.

1. Switch on the PC and the controller.
2. Change to the ProOED3 project directory, e.g.:

   CD C:\USR\PROOED3.01X\PROJECTS

   or

   CD C:\USR\PROOED3.300\PROJECTS

   and press the <↵> key.
3. Enter the following command:

   START /TEST

   and press the <↵> key.

A menue with a few testing functions is displayed.

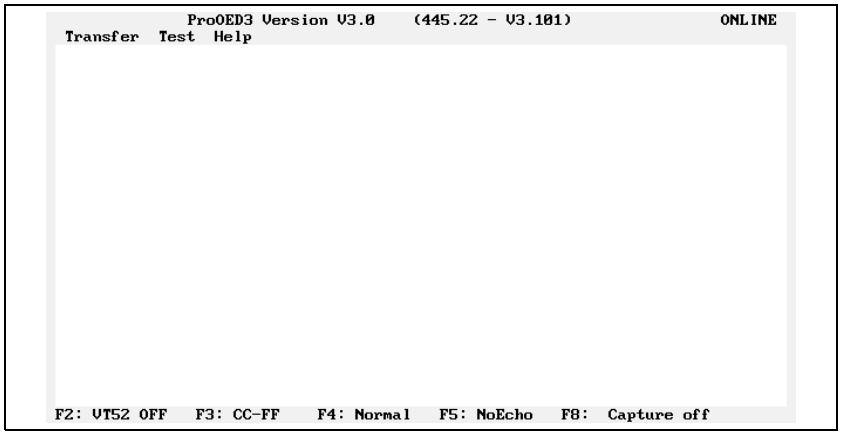


Fig. 3-3 Setup/test

To test the wiring, proceed as follows:

4. Press <Alt>-<F> to open the pull-down menue “Transfer”.

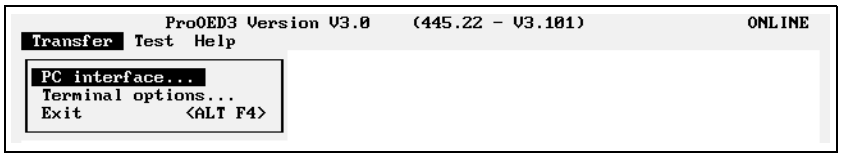


Fig. 3-4 Pull-down menue Transfer

5. The highlight is on “PC interface”.
   => Press the <↵> key.

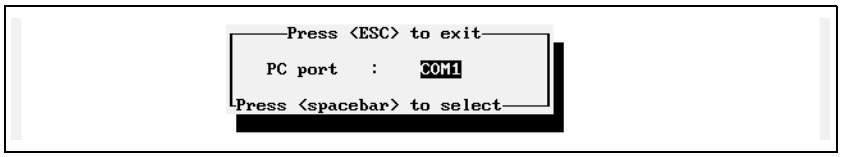
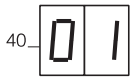
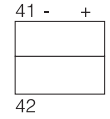


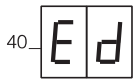
Fig. 3-5 Setting the PC Interface

6. Use the spacebar to select the desired PC interface.
7. Close the window by pressing <Esc>.
8. Switch on the controller.
9. Set the controller to STOP status (status display on the controller must show “01”), see controller manual.

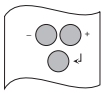


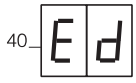
10. Select editing mode.





Series 300:
Press and hold key 41 together with key 42 on “+” side until “Ed” appears in the controller status display.





WDP3-014/018:
Press and hold the “+” key together with the “↵” key until “Ed” appears in the controller status display.

NOTE
If the “EDIT>” message appears on the screen, the wiring between PC and controller is o.k. Otherwise it is necessary to check the wiring, the interface assignment and possibly the terminal option settings.

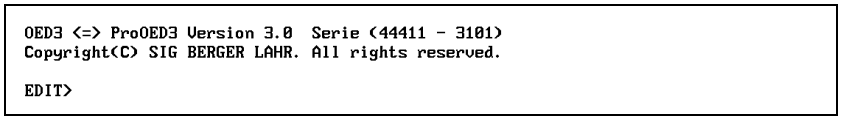


Fig. 3-6 Message “EDIT”

11. Press <Alt>-<T> to open the pull-down menue “Test”.

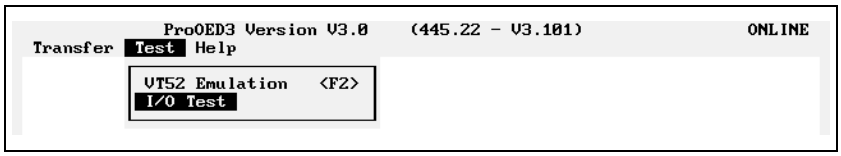


Fig. 3-7 Selecting I/O test

12. Press the <E> key to activate the I/O test.

    On the right of “I000-...”, ProOED3 displays the states of the inputs from left to right. This allows you to check the connected inputs easily.

    On the right of “Q000-...”, the states of the outputs are displayed:

13. Use the <←> and <→> keys to move the cursor (see figure).

14. Press the spacebar to change the output status.

15. Observe the status indicators for the outputs on the front panel of the controller.

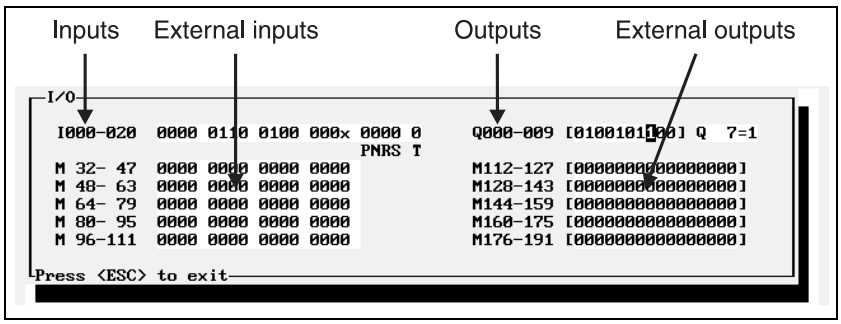


Fig. 3-8 I/O test

NOTE
On WDP3-014/018 controllers, only the inputs I0 to I8, I10, I11, I12 (LIMP), I13 (LIMN), I14 (REF.) and I15 (STOP) as well as the outputs Q0-Q3 are indicated. External I/O modules cannot be controlled with these controllers.

16. Exit the I/O test by pressing <Esc>.

17. Exit the program by pressing <Alt>-<F4>.

I/O modules (Series 300 only)

If external I/O modules of type MP 926 are connected to a Series 300 controller and adjusted to the appropriate parameter settings (see chapter 6.1.2), M32 to M111 show the states of the inputs and M112 to M191 the states of the outputs (see chapter 6.3.3). The output states of the I/O cards can also be changed by pressing the spacebar.

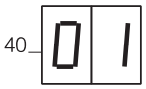
3.3 Testing the motor function Series 300

In manual mode, the selector switch (item 41) on the controller front panel can be used for moving the stepping motor in a clockwise (CW) or counterclockwise (CCW) direction. This can be used for checking the wiring of the stepping motor.

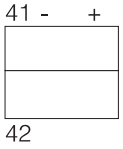
ATTENTION
Before moving the stepping motor manually, it is necessary to set the nominal motor current via the front panel; see controller manual.

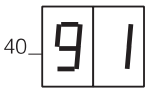
NOTE
In manual mode, all limit switches are monitored. This means that the LIMP, LIMN and STOP inputs must be energized (make sure they are wired).

1. Set the nominal motor current.

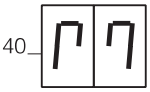


2. Set STOP status.
   Press the selector switch (item 41) on "–" side. The number "01" appears in the display.





3. Set manual mode.
   Keep the selector switch (item 42) pressed on "+" side. After 2 seconds, the 7-segment displays (item 40) start flashing. Set the number for manual mode for the appropriate axis by pressing "+" or "–" on the selector switch (item 41).



Axis	Number
1	91
2	92
3	93
4	94

   Release the selector switch (item 42) to accept the setting. A flashing "M" appears in the 7-segment displays (item 40) to indicate manual mode.

4. You can move the shaft of the stepping motor in single steps or in continuous operation.

Action	Effect
Press the selector switch (item 41) momentarily on "+" side	The stepping motor shaft moves in positive direction step by step.
Press the selector switch (item 41) momentarily on "–" side	The stepping motor shaft moves in negative direction step by step.
Keep the selector switch (item 41) pressed on "+" side	The stepping motor shaft moves in positive direction in continuous operation.
Keep the selector switch (item 41) pressed on "–" side	The stepping motor shaft moves in negative direction in continuous operation.

5. Exit manual mode by pressing selector switch (item 42) on "+" side.

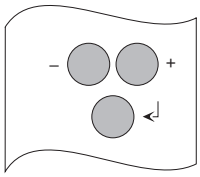
3.4 Testing the motor function WDP3-014/018

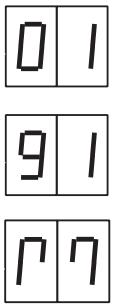
In manual mode, you can move the stepping motor in a clockwise or counterclockwise direction using the front panel keys. This can be used for checking the wiring of the stepping motor.

ATTENTION
Before moving the stepping motor manually, it is necessary to set the nominal motor current via the front panel; see controller manual.

NOTE
In manual mode, all limit switches are monitored. This means that the LIMP, LIMN and STOP inputs must be energized (make sure they are wired).

1. Set the nominal motor current.





2. Set STOP status.
   Press the "–" key until "01" is displayed.

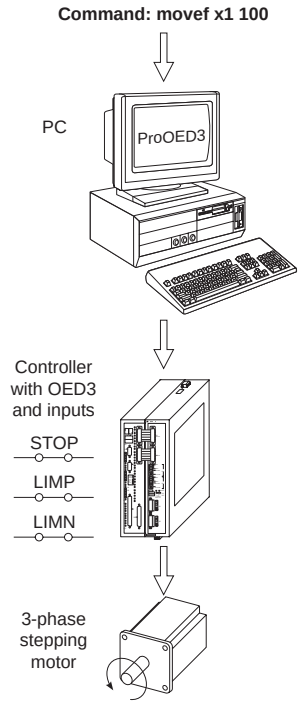
3. Select manual mode.
   Press the "↵" key until "01" starts flashing (3 sec.).
   Keep the "↵" key pressed and press the "+" key several times until "91" is displayed.
   Release the "↵" key. A flashing "M" is displayed.

4. You can move the shaft of the stepping motor in single steps or in continuous operation.

Action	**Effect**
Press the selector switch momentarily on "+" side	The stepping motor shaft moves in positive direction step by step.
Press the selector switch momentarily on "–" side	The stepping motor shaft moves in negative direction step by step.
Keep the selector switch pressed on "+" side	The stepping motor shaft moves in positive direction in continuous operation.
Keep the selector switch pressed on "–" side	The stepping motor shaft moves in negative direction in continuous operation.

5. Exit manual mode by pressing the "↵" key.

4 Creating a simple project



The following chapter describes the procedure for creating a simple program using ProOED3.

Prerequisites

Correct setup of ProOED3 and controller; see chapter 3.
The active low inputs STOP, LIMP and LIMN must be connected to 24 V.

Task

The motor is to move by 100 steps repeatedly.

Program

The OED3 control program can consist of a PLC and a SEQUENCE program component. Movement sequences are programmed in the SEQUENCE program.
For this task, only the SEQUENCE program must be created.
The SEQUENCE program for this task is as follows:

```
movef     x1   100       ;Comment: This command
                         ;moves the motor of axis
                         ;x1 forward by 100 steps.
```

A program is always processed in cyclic execution, i.e. the command is repeated after execution and the motor turns another 100 steps, etc.

Programming with ProOED3

The PC and the controller must be correctly interconnected for programming the controller; see chapter 3.

Step 1

Starting ProOED3

1. Switch on your PC.
2. Change to the ProOED3 project directory, e.g.:

   `CD C:\USR\PROOED3.01X\PROJECTS`

   or

   `CD C:\USR\PROOED3.300\PROJECTS`

   and press the <↵> key.
3. Enter the following command:

   `START`

   and press the <↵> key.

Step 2

Creating a new project

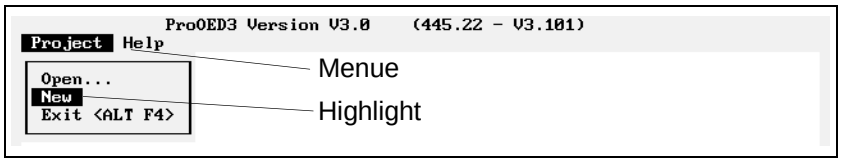
A project is an application program which consists of the following program components:

– PLC program
– SEQUENCE program
– Control parameters
– Position variables
– Symbolic names
– Send texts

A new project is created as follows:

1. Press the key combination <Alt>-<P> to open the “Project” pull-down menue.
2. Use the <↓> key to move the highlight to “New”.
3. The highlight is on “Project/New”.
   => Press the <↵> key.

Select the “Project/New” menue option

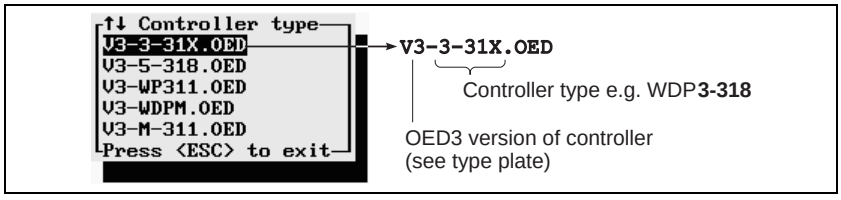


4. Enter a project name (8 characters max.) and press the <↵> key.

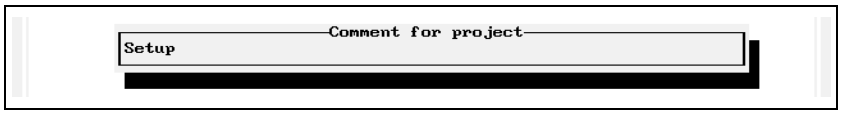
Enter a project name

5. Use the <↑> and <↓> keys to select the type of the connected controller and press the <↵> key.

Select a controller type



6. Enter a comment on the project, e.g. “Setup” and press the <↵> key. ProOED3 then creates an empty project.



Enter a comment on the project

Step 3

Creating a control program

You can test the operability of the system with a one-line program. Enter this program as follows:

Entering the SEQUENCE program

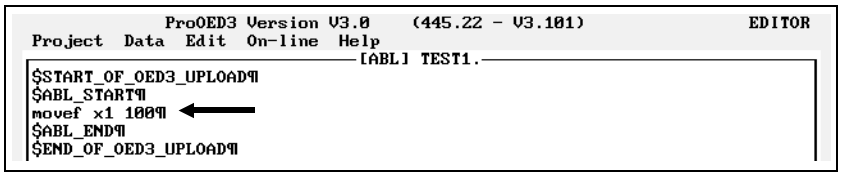
1. Press <↓> twice to move the cursor to the beginning of the line

   $ABL_END

2. Enter the following command:

   movef x1 100

3. Press <↵> key to insert a new line.



Enter a command

Saving the SEQUENCE program

1. Press <Alt>-<P> to open the pull-down menue “Project”.
2. Use the <↓> key to move the highlight to “Save” and press the <↵> key.

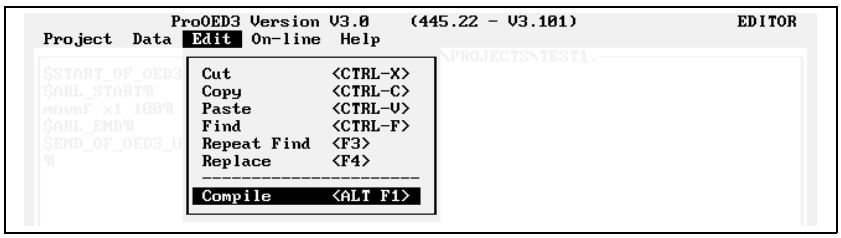
Save the SEQ program

Step 4

Compiling a SEQ program

The SEQ program must be compiled before loading it into the controller:

1. Press <Alt>-<B> to open the pull-down menue “Edit”.
2. Use the <↓> key to move the highlight to “Compile” and press the <↵> key. A message is displayed which informs you that the project has been compiled successfully (acknowledge by pressing the <↵> key).



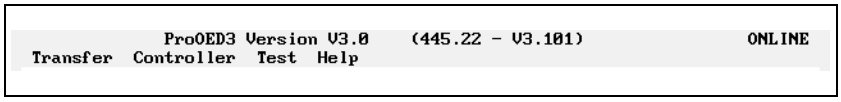
Compile the SEQ program

NOTE
To be able to execute the program on the controller, it must first be transferred into the controller memory using the on-line functions (Down-load).
This requires a correct link to the controller; see chapter 3.

Step 5

Selecting on-line functions

Press <Alt>-<O>. The on-line functions are displayed for selection. The on-line functions establish the link to the controller.

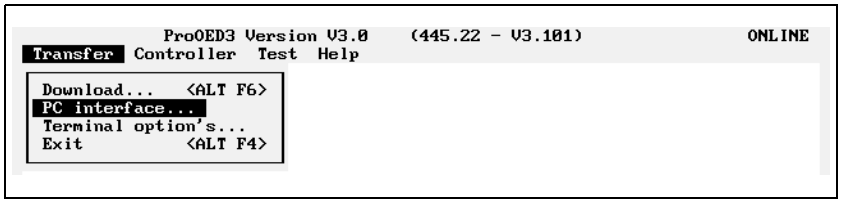


On-line

Step 6

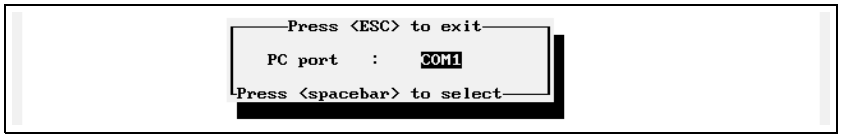
Selecting the PC interface

1. Press <Alt>-<F> to open the pull-down menue “Transfer”.
2. Use the <↓> key to move the highlight to “PC interface” and press the <↵> key.



Select PC interface

3. Use the spacebar to select the appropriate PC interface.



Set the PC interface

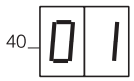
The interface parameters are automatically set by ProOED3. The following interface parameters are set by default on the controller:

Parameter	**Setting**
Baud rate	9600
Parity	EVEN
Stop bits	1
Data bits	7
Handshake	XON/XOFF

Step 7

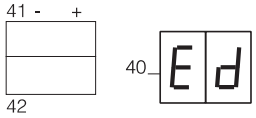
Activating editing mode on the controller

The controller must be in editing mode for the PC to be able to communicate with the controller.
Only in this mode, programs can be loaded into the controller.

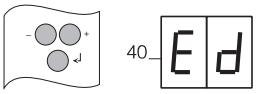


1. Set the controller to STOP status (status display on the controller must show “01”); see controller manual.

2. Select editing mode.



Series 300:
Press and hold key 41 on “+” side.
Press key 42 on “+” side and hold it together with key 41 until “Ed” appears in the controller status display.



WDP3-014/018:
Press and hold the “+” key together with the “↵” key until “Ed” appears in the controller status display.

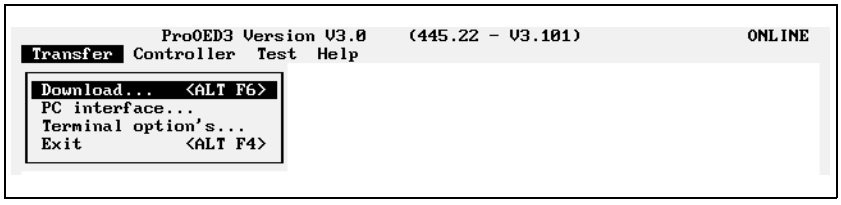
NOTE
“EDIT” is only displayed if the communication link between the PC and the controller is functional. If not, check the wiring and the interface configuration.

Step 8

Loading a project into the controller (Download)

1. Press <Alt>-<F> to open the pull-down menue “Transfer”.

2. Move the highlight to “Download” and press the <↵> key.

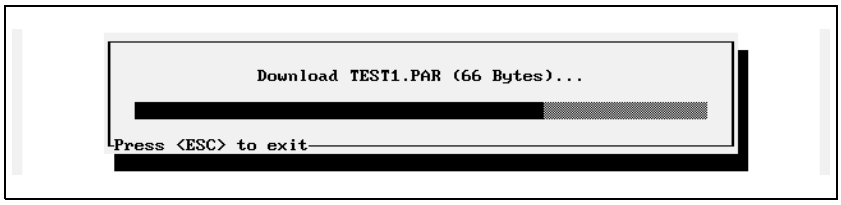


Select downloading

3. The highlight is on “Complete”.
   => Press the <↵> key.
   “Complete” means that the entire project is loaded into the controller.

Select an option

Variables, control parameters, texts, and the control program are transferred to the controller.

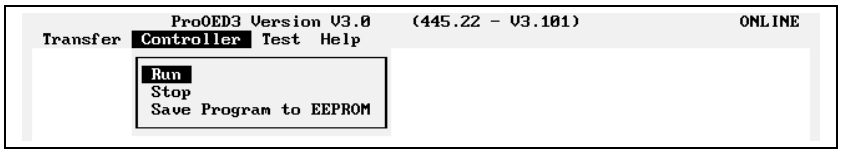


Download

Step 9

Starting the control program

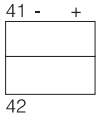
1. Press <Alt>-<S>.
2. The highlight is on “Run”. => Press the <↵> key.



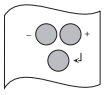
Start the program

NOTE
Since the control program is processed in cyclic execution, the motor turns by 100 steps, then stops, turns by another 100 steps, etc.

If there is no on-line link to the PC, the control program can also be stopped and started from the front panel.



Series 300:
Selector switch item 41: “–” = STOP, “+” = RUN



WDP3-014/018:
“–” key = STOP
“+” key = RUN

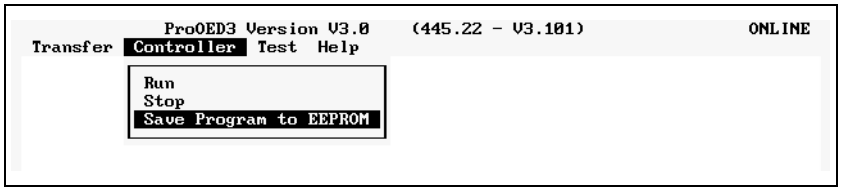
NOTE
The control program can be viewed from the PC during program execution; see chapter 5.4.3.5.

Step 10

Saving the control program in EEPROM

In order to retain the control program on a WDP3-014/018 controller after power-off, it must be saved to the controller's EEPROM.
You should also carry out this step on a Series 300 controller although this controller type has a battery-buffered RAM.

1. Press <Alt>-<S>.
2. Move the highlight to "Save Program to EEPROM" and press the <↵> key.



Saving program in EEPROM

NOTE
The control program can be viewed from the PC during program execution; see chapter 5.4.3.5.

5 Operation

User interface

This chapter describes the operation of ProOED3.
Figure 5-1 illustrates the display screen areas and the most important keys.

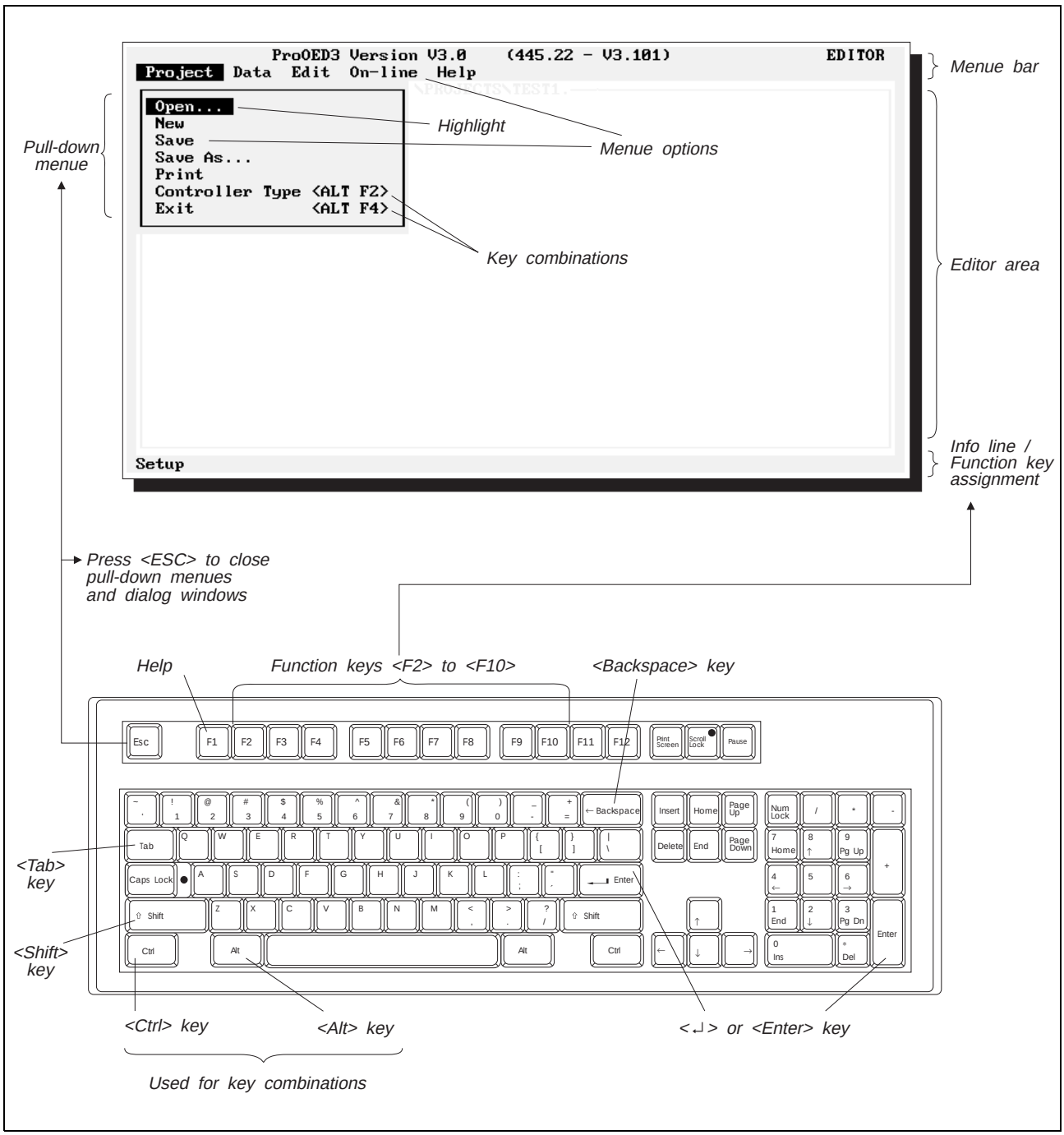


Fig. 5-1 ProOED3 user interface

The most important keys

The following table contains the functions of the most important keys.

Key	Function
<↑>, <↓>	Move the cursor to the corresponding direction.
	In windows with selection lists: Move the list to the start or end.
	Move the highlight
<←>, <→>	Move the cursor to the corresponding direction.
<↵>	Invoke the menue option marked by the highlight.
	Insert a new line.
	Insert a hard return.
<Ins>	Toggle insert and replace mode.
<Del>	Delete the character to the right of the cursor.
<Home>	Jump to the beginning of the line.
	In windows with selection lists: Jump to the beginning of the list.
<End>	Jump to the end of the line.
	In windows with selection lists: Jump to the end of the list.
<Esc>	Close the pull-down menue.
	Close the input or dialog window.
<Backspace>	Delete the character to the left of the cursor.
<Page Up>	Scroll upwards by pages.
<Page Down>	Scroll downwards by pages.
<Tab>	Enter a tab.
<F1> ... <F10>	Function keys: To directly execute frequently used functions. The key assignment is displayed in the status line.

General information on operation

Selecting menue options

Menue options are selected as follows:

1. Open the pull-down menue by pressing the key combination:

   <Alt>-<*Hotkey*>

   The *hotkey* is the key which corresponds to the highlighted letter of a menue option.

2. With the pull-down menue open, press the *hotkey* of the desired menue option

   or

   use the <↑> and <↓> keys to move the menue bar to the desired menue option and press the <↵> key.

NOTE
A menue option can also be invoked with a key combination. If a menue option is assigned a key combination, it is displayed next to the option.

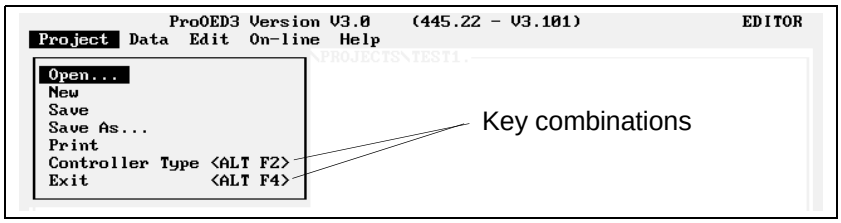


Fig. 5-2 Key combinations

Selection windows

ProOED3 features various selection windows from which you can select project names, controller types and other options.

1. Use the <↑> and <↓> keys to move the highlight to the desired item.

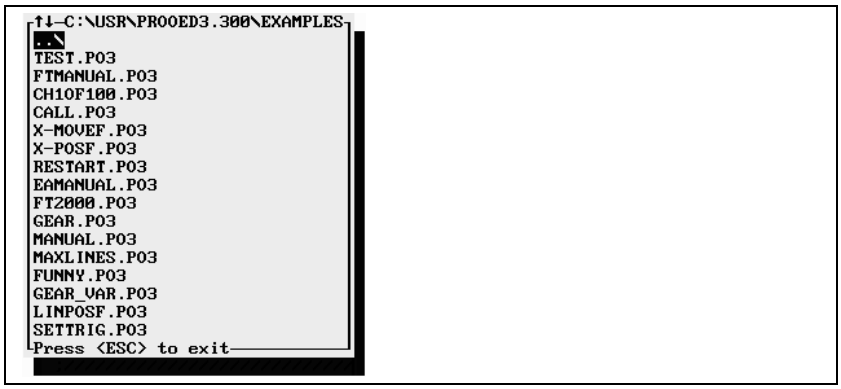


Fig. 5-3 Selection window

2. Press the <↵> key.

5.1 Project-related functions (Project)

The “Project” pull-down menue contains functions related to the entire project or a part of it.

Fig. 5-4 Project

Menue option	Function	Chapter
Open	Open an existing project.	5.1.1
New	Create a new project.	5.1.2
Save	Save the project to the hard disk.,	
Save As	Save the project under a different name.	5.1.3
Print	Print all project data to a file.	5.1.4
Controller Type	Select the type of the connected controller.	5.1.5
Exit	Exit ProOED3 and return to DOS.,	

5.1.1 Opening a project

Open a project stored on the hard disk.

1. Select the menue option “Project/Open”.
2. Use the <↑> and <↓> keys to select the directory and press the <↵> key. To return to the previous directory level, select “..\”.

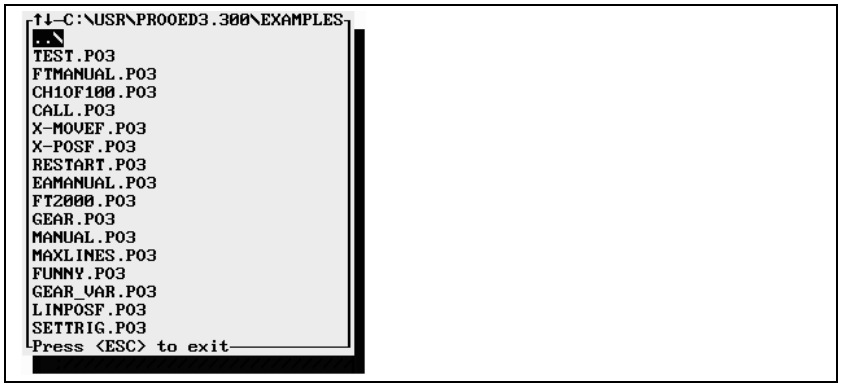


Fig. 5-5 Opening a project

3. Use the <↑> and <↓> keys to select the project and press the <↵> key. The project is opened and ready for editing.

NOTE
For converting a ProOED3 version 2 project to a version 3 project, see chapter 2.6.

5.1.2 Creating a new project

To create a new project, proceed as follows:

1. Select the menue option “Project/New”.
2. Enter a project name and press the <↵> key. The project name must not be longer than 8 characters and not include any special characters.

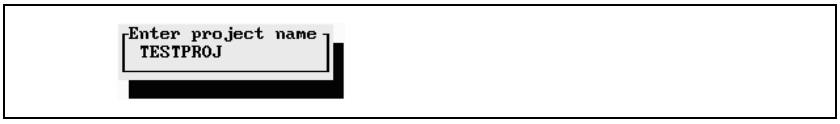


Fig. 5-6 Creating a new project

3. Select the controller type.

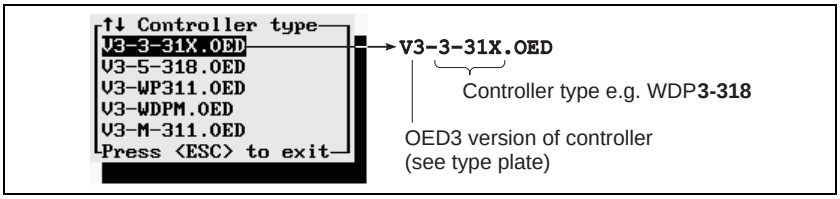


Fig. 5-7 Selecting the controller type

4. Enter a comment on the project. This comment will be displayed later in the status line.

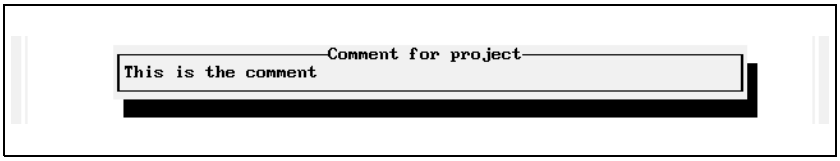


Fig. 5-8 Entering a comment for the project

ProOED3 then creates a new project. With the exception of four lines each in the SEQUENCE and PLC editor, this project does not contain any data yet. These four lines mark the beginning and end of the SEQUENCE and PLC program components and must neither be deleted nor modified.

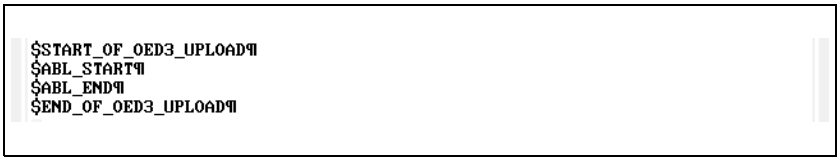


Fig. 5-9 Beginning and end of a SEQ program

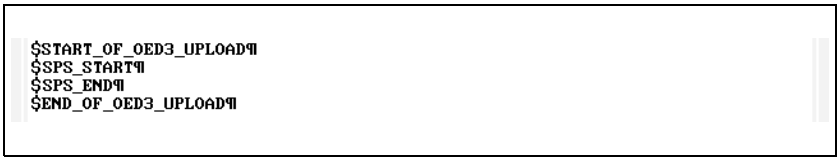


Fig. 5-10 Beginning and end of a PLC program

5.1.3 Save As

This option is used for copying an existing project and saving it under a different name.

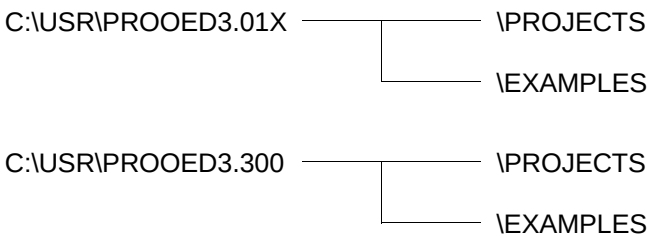
1. Select the menue option “Project/Save As”.
2. Enter a new project name and press the <↵> key. The project name must not be longer than 8 characters and not include any special characters.

NOTE
The project will be stored in the current project directory “...\PROJECTS”.

You can also copy the project to a different directory.

Example:

To copy the MANUAL project from the EXAMPLES directory to the PROJECTS directory and use it there with the name TEST1.



1. Press <Alt>+<F4> to exit ProOED3 and return to the DOS command prompt.
2. Enter

   CD C:\USR\PROOED3.01X\PROJECTS

   or

   CD C:\USR\PROOED3.300\PROJECTS

   and press the <↵> key.
   If the destination directory does not exist yet, create the directory with the command

   MD C:\USR\PROOED3.01X\PROJECTS

   or

   MD C:\USR\PROOED3.300\PROJECTS
3. Enter

   COPY C:\USR\PROOED3.01X\EXAMPLES\MANUAL.*␣TEST1.*

   or

   COPY C:\USR\PROOED3.300\EXAMPLES\MANUAL.*␣TEST1.*

   and press the <↵> key.

   You can then open and edit the “TEST1” project just as any other project.

5.1.4 Print

To store all project data in a file which can be output to the printer with the DOS command PRINT:

1. Select the menue option “Project/Print”.

   A window with the file name of the print file opens.

Fig. 5-11 Printing a project

2. Exit ProOED3.
3. Change to the project directory, e.g. by entering

   CD C:\USR\PROOED3.01X\PROJECTS

   or

   CD C:\USR\PROOED3.300\PROJECTS.
4. Use the PRINT command to output the print file to the printer, e.g.

   PRINT TESTPROJ.LS0.

5.1.5 Selecting the controller type

Different command sets are valid for the various controller types. For example, it would be impossible to address a fourth axis on a controller with only one axis.
For this reason, OED3 must know at program compilation time for which controller type the program has been designed.

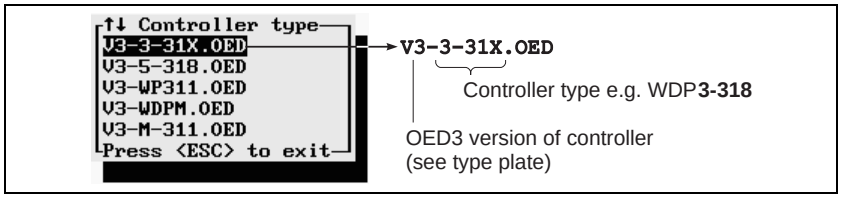


Fig. 5-12 Selecting the controller type

The controller type highlighted in black is the current one.

Select by pressing the <↓> and <↑> keys, followed by <↵>.

5.2 Data (editors)

This menue offers several editors for selection.

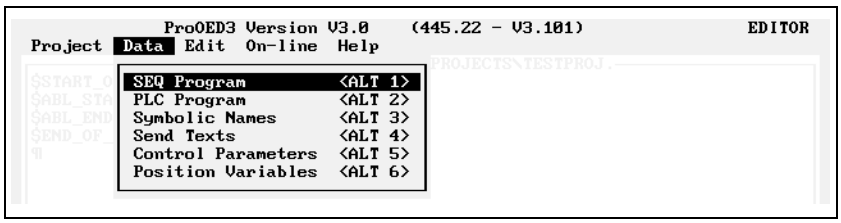


Fig. 5-13 Editors

Menue option	Function	Chapter
SEQ Program	To call the SEQUENCE program editor.	5.2.1
PLC Program	To call the PLC program editor.	5.2.1
Symbolic Names	To call the editor for symbolic names.	5.2.2
Send Texts	To call the editor for send texts.	5.2.3
Control Parameters	To call the editor for control parameters.	5.2.4
Position Variables	To call the editor for position variables.	5.2.5

5.2.1 SEQUENCE and PLC editors

These editors are used for entering the program code.
Operation of the two editors is identical.

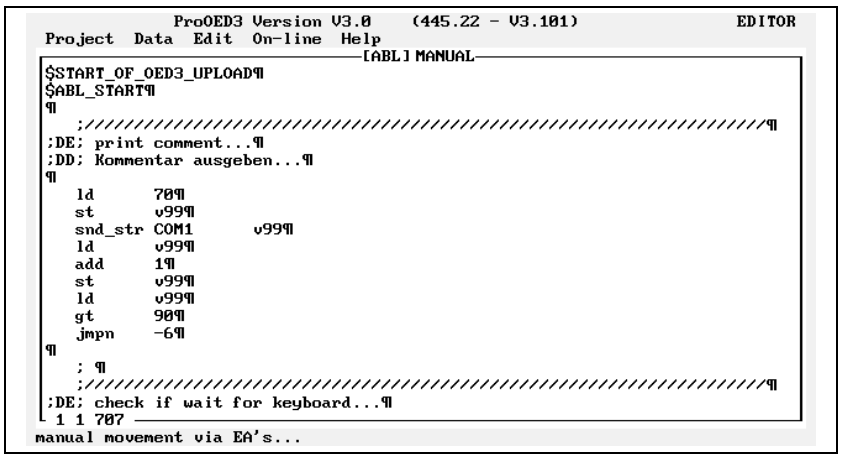


Fig. 5-14 SEQUENCE editor

NOTE
The first two lines and the last two lines marking the beginning and end of the program must not be modified or deleted.

NOTE
All editing features (pull-down menue "Edit") are available in these editors. See chapter 5.3.

The following table shows the functions of the keys and key combinations:

Key	Function
<↑>, <↓>	Move the cursor to the corresponding direction.
	In the help window: Move the selection bar.
<←>, <→>	Move the cursor to the corresponding direction.
<↵>	Insert a new line.
	Insert a hard return.
<Ins>	Toggle insert and replace mode.
<Del>	Delete the character to the right of the cursor.
<Home>	Jump to the beginning of the line.
	In the help window: Jump to the beginning of the list.
<End>	Jump to the end of the line.
	In the help window: Jump to the end of the list.
<Esc>	Close the help window.
<Backspace>	Delete the character to the left of the cursor.

Key	Function
<Page Up>	Scroll upwards by pages.
<Page Down>	Scroll downwards by pages.
<Tab>	Enter a tab.
<Shift>-<↑> <Shift>-<↓> <Shift>-<←> <Shift>-<→>	Mark text for copying or cutting.
<Strg>-<C>	Copy marked text to the clipboard.
<Strg>-<X>	Cut out marked text and copy it to the clipboard.
<Strg>-<V>	Insert text from the clipboard at the cursor position.
<Strg>-<Y>	Delete the line marked by the cursor.
<Strg>-<Home>	Jump to first line.
<Strg>-<End>	Jump to last line.
<Strg>-<F>	Find a specific string.
<Alt>-<G>	Go to a specific line.
<Alt>-<F1>	Compile the contents of the current editor.
<Alt>-<F2>	Change the controller type.
<F1>	Open the help window with a list of commands.
<F3>	Repeat finding.
<F4>	Find and replace.

Command list via <F1> key

Pressing <F1> activates a help window with operands, valid ranges of values and commands. The contents of the list depends on the current controller type and editor.

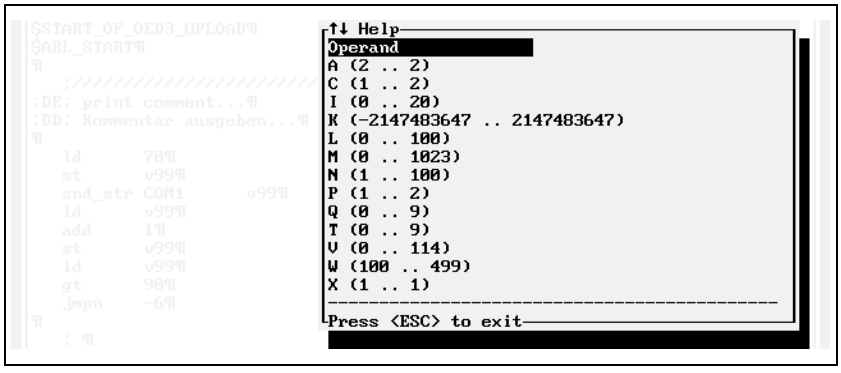


Fig. 5-15 Command list

5.2.2 Editing symbolic names

This editor can be used for assigning symbolic names to constants, variables, etc.
To invoke the editor, either select “Data/Symbolic Names” or press <Alt>-<3>.

NOTE
Symbolic names are case-sensitive.

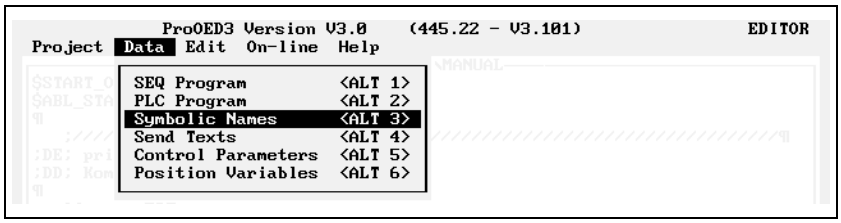


Fig. 5-16 Selecting symbolic names

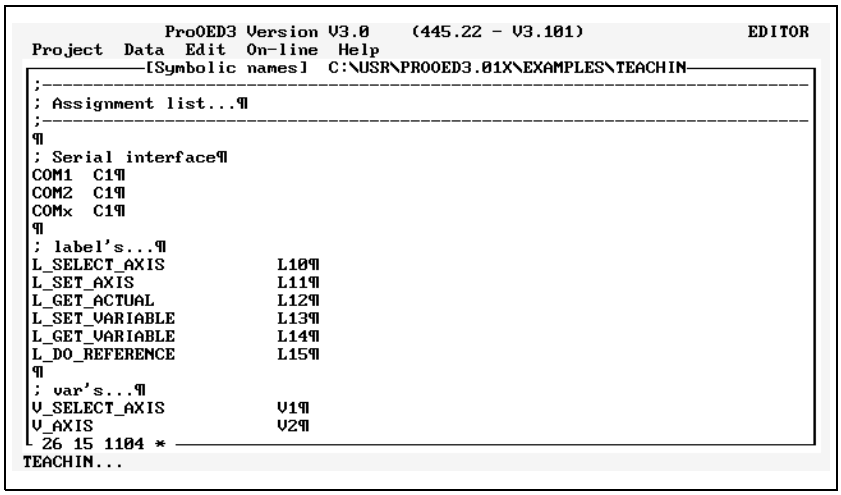


Fig. 5-17 Symbolic names

During compilation, symbolic names are replaced by the original names (see figure).
Using symbolic names is not mandatory, however, it will enhance the readability of a program.

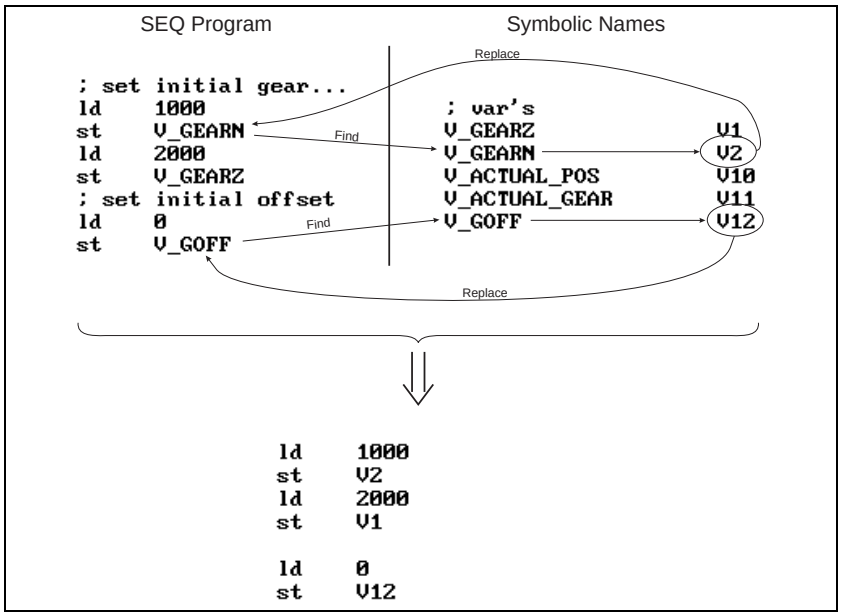


Fig. 5-18 Symbolic name translation

Rules for symbolic names

The following rules apply to symbolic names:

- Symbolic names are valid for the SEQUENCE and the PLC programs.
- The maximum length is 30 characters.
- A maximum of 250 assignments per project are allowed in total.
- A symbolic name must be contiguous and not contain any blanks or tabs.

NOTE
*Comments can be entered in addition. As in the SEQUENCE and PLC editor, they must be marked with a "**;**" .*

Using the editors:

Key	Function
<↑>, <↓>	Move the cursor to the corresponding direction.
	In the help window: Move the selection bar.
<←>, <→>	Move the cursor to the corresponding direction.
<↵>	Insert a new line.
	Insert a hard return.
<Ins>	Toggle insert and replace mode.
<Del>	Delete the character to the right of the cursor.
<Home>	Jump to the beginning of the line.
	In the help window: Jump to the beginning of the list.
<End>	Jump to the end of the line.
	In the help window: Jump to the end of the list.
<Esc>	Close the help window.
<Backspace>	Delete the character to the left of the cursor.
<Page Up>	Scroll upwards by pages.
<Page Down>	Scroll downwards by pages.
<Tab>	Enter a tab.
<Shift>-<↑> <Shift>-<↓> <Shift>-<←> <Shift>-<→>	Mark text for copying or cutting.
<Strg>-<C>	Copy marked text to the clipboard.
<Strg>-<X>	Cut out marked text and copy it to the clipboard.
<Strg>-<V>	Insert text from the clipboard at the cursor position.
<Strg>-<Y>	Delete the line marked by the cursor.
<Strg>-<Home>	Jump to first line.
<Strg>-<End>	Jump to last line.
<Strg>-<F>	Find a specific string.
<Alt>-<G>	Go to a specific line.
<Alt>-<F1>	Compile the contents of the current editor.
<Alt>-<F2>	Change the controller type.
<F1>	Open help window.
<F3>	Repeat previous find operation.
<F4>	Find and replace.

5.2.3 Editing send texts

Send texts are used to be output on a terminal, e.g. FT 2000 or a PC with BTERM installed.
To invoke the editor, either select “Data/Send Texts” or press <Alt>-<4>.

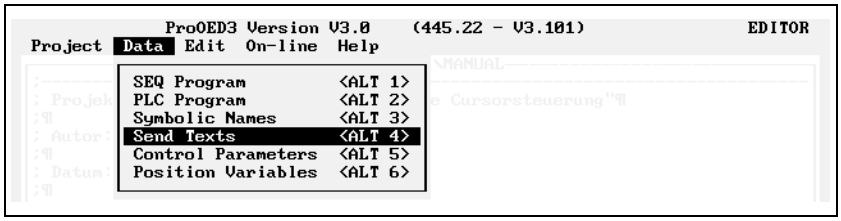


Fig. 5-19 Selecting send texts

When the editor is opened, the screen below is displayed. The send text shown in inverse representation can be edited.
Move the highlight by pressing <↑> or <↓>.
Move the text cursor by pressing <←> or <→>.
At the bottom of the “Send Texts” window, the number of the string marked by the highlight is displayed. This number is used in programming for identifying the send text.

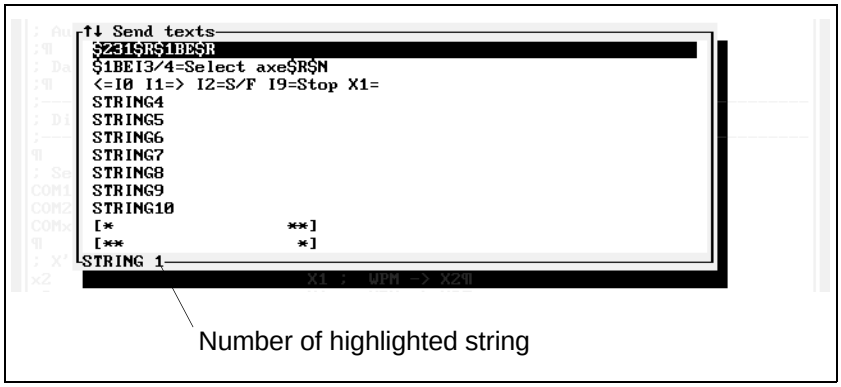


Fig. 5-20 Send texts

NOTE
The “$XX” characters are control codes. For a more detailed description, refer to chapter 6.3.5.

Exit the “Send Texts” editor by pressing <Esc>. A window with the message “Saving the file” is displayed:

– To exit without saving => Press <Esc>.
– To exit with saving => Press the <↵> key.

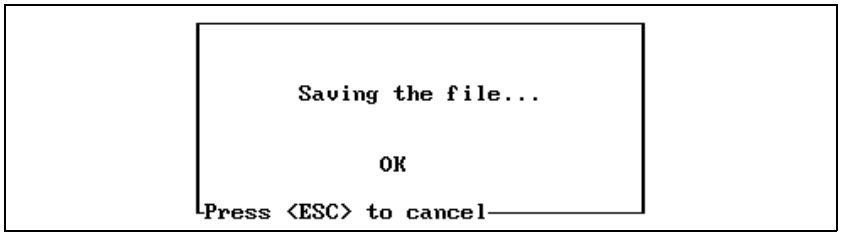


Fig. 5-21 Exiting send text editing

5.2.4 Editing control parameters

To invoke the editor, either select “Data/Control Parameters” or press <Alt>-<5>.

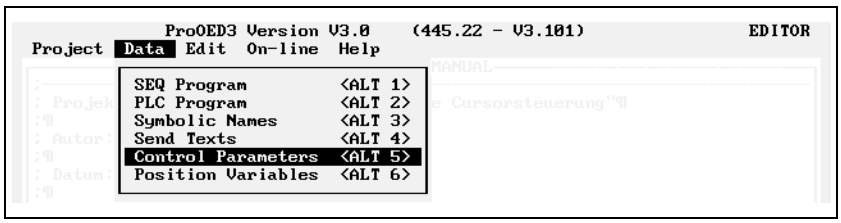


Fig. 5-22 Selecting control parameters

When the editor is opened, the screen below is displayed. The parameter shown in inverse representation can be edited.
Move the highlight by pressing <↑> or <↓>.
At the bottom of the “Controller parameters” window, the minimum and maximum values for the control parameter marked by the highlight are displayed.

NOTE
For a description of the control parameters, refer to chapter 6.1.2.

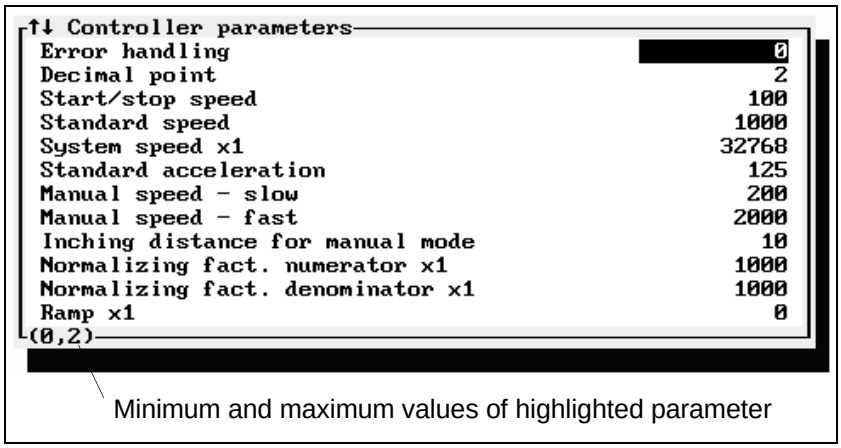


Fig. 5-23 Control parameters

Exit the editor by pressing <Esc>. A window with the message “Saving the file” is displayed:

– To exit without saving => Press <Esc>.
– To exit with saving => Press the <↵> key.

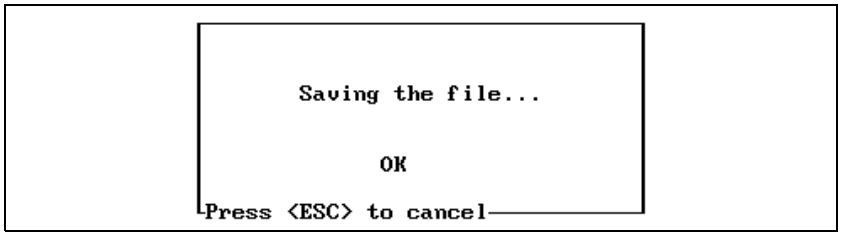


Fig. 5-24 Exiting control parameter editing

5.2.5 Editing position variables

To invoke the editor, either select “Data/Position Variables” or press <Alt>-<6>.

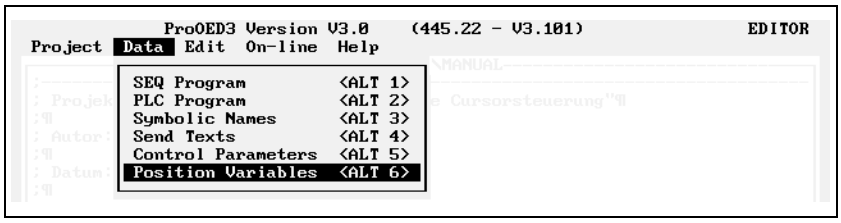


Fig. 5-25 Selecting position variables

When the editor is opened, the screen below is displayed. The position variable shown in inverse representation can be edited.
Move the highlight by pressing <↑> or <↓>.

NOTE
Position variables can also be read in directly from the controller into the variables with the “teach-in” feature.
See chapter 5.4.3.3.

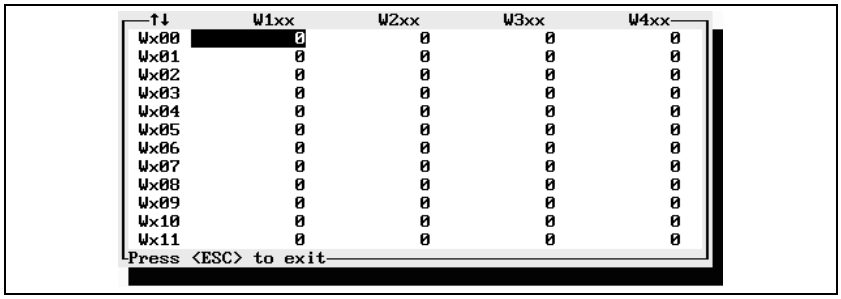


Fig. 5-26 Position variables

A total of 400 position variables from w100 to w499 are available for the axes x1 to x4.

5.3 The “Edit” pull-down menue

This menue contains useful editing features and a menue option for compiling the program in the current editor.

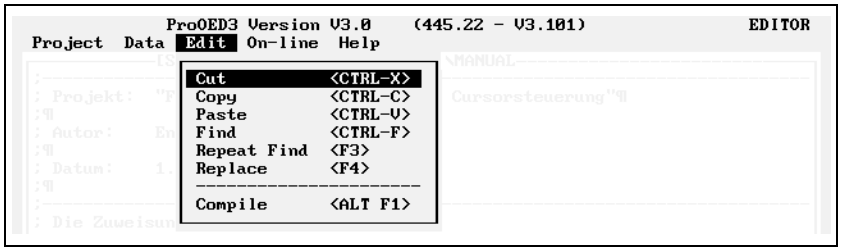


Fig. 5-27 Editing

Menue option	Function	Chapter
Cut	Cut out marked text and copy it to the clipboard.	5.3.1
Copy	Copy marked text to the clipboard.	5.3.2
Paste	Insert the contents of the clipboard at the cursor position.	5.3.3
Find	Find a specific string.	5.3.4
Repeat Find	Repeat a finding operation without having to enter a new string.,	
Replace	Find and replace a string.	5.3.5
Compile	Compile the contents of the current editor. The SEQUENCE and PLC programs must be compiled before they can be loaded into the controller.	5.3.6

5.3.1 Cut

Mark text, cut it out and copy it to the clipboard as follows:

1. Press and hold the <Shift> key.
2. Use the <↑>, <↓>, <←> or <→> keys to mark the desired text area and release the <Shift> key.

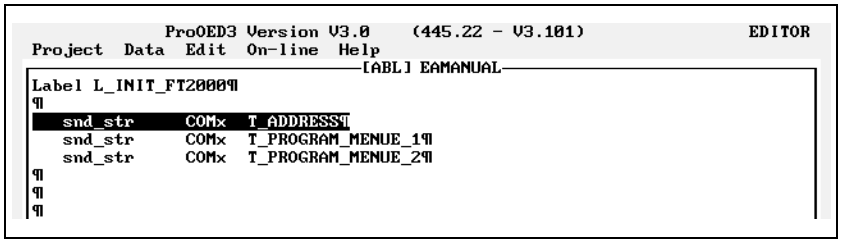


Fig. 5-28 Marking text

3. Select the menue option “Edit/Cut” or press <Ctrl>-<X>.

The marked text is removed and stored in the clipboard.

5.3.2 Copy

Mark text and copy it to the clipboard as follows:

1. Press and hold the <Shift> key.
2. Use the <↑>, <↓>, <←> or <→> keys to mark the desired text area and release the <Shift> key.

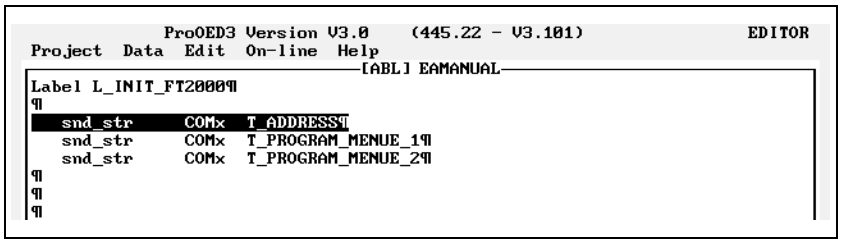


Fig. 5-29 Marking text

3. Select the menue option "Edit/Copy" or press <Ctrl>-<C>.

The marked text is stored in the clipboard.

5.3.3 Paste

To paste the contents of the clipboard:

1. Move the cursor to the position where you want to insert the text.
2. Select "Edit/Paste" or press <Ctrl>-<V>.

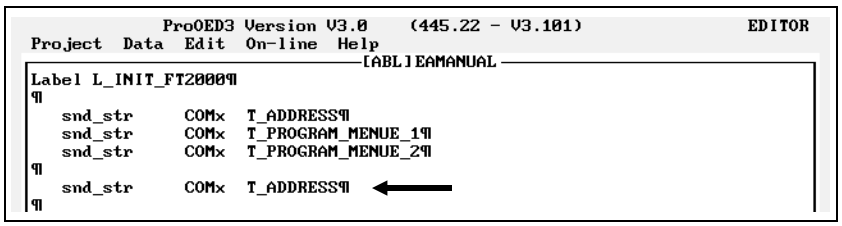


Fig. 5-30 Pasting text

The contents of the clipboard is then inserted at the cursor position.

5.3.4 Find

To find a specific string within the current editor:

1. Select the menue option "Edit/Find" or press <Ctrl>-<F>.

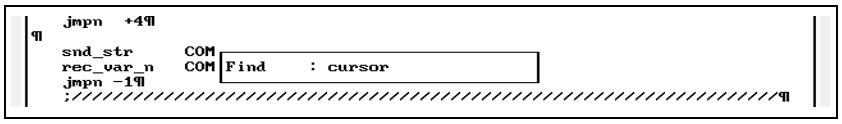


Fig. 5-31 Finding text

2. Enter a string and press the <↵> key.

If the specified string is found, ProOED3 moves the cursor to the corresponding location.
Otherwise a window with a message is displayed.

5.3.5 Replace

Find and replace a specific string:

1. Select the menue option “Edit/Replace”.
2. Enter the string to be found and press the <↵> key.
3. Enter the string to replace the string found and press the <↵> key.

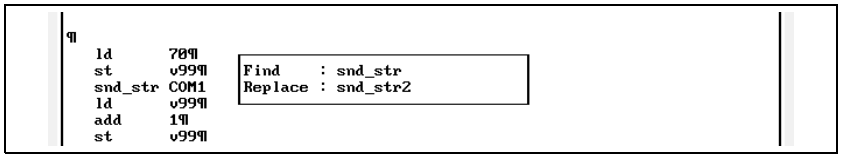


Fig. 5-32 Replacing text

4. If the string is found, a window with three options is displayed:

   Press <Y> to replace once
   Press <N> to skip
   Press <A> to replace all matching strings

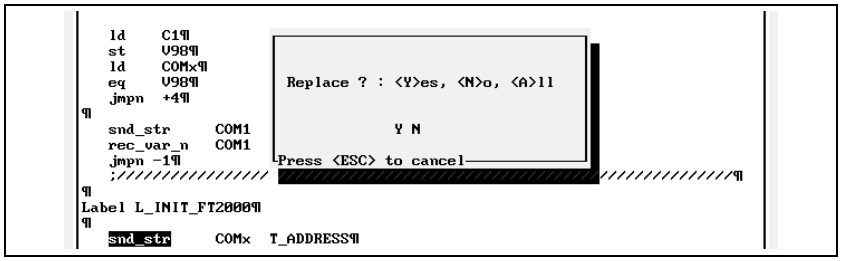


Fig. 5-33 Confirm replacing

5.3.6 Compile

A program *must* be compiled before loading it into the controller.

ATTENTION
It is indispensable that the correct controller type be set for compiling (Project/Controller Type).

In the compilation process, ProOED3 translates the program to a language which can be interpreted by the controller. If there are no errors in the program, a message is displayed which informs you about this.

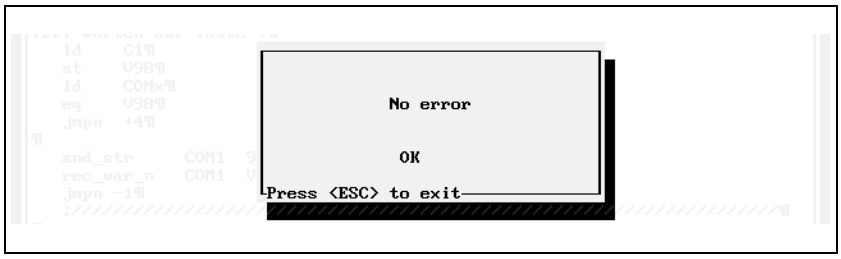


Fig. 5-34 Program compilation

During compilation, a syntax check is performed. If errors are found, a message window informs you about these. See also chapter 7.1.1.

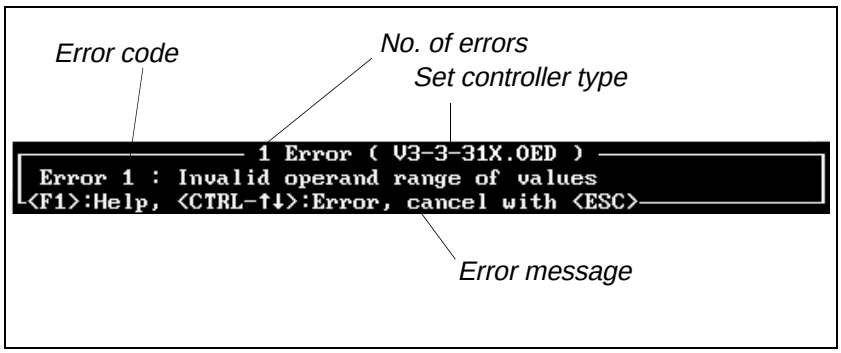


Fig. 5-35 Compilation error

Key	Function
<F1>	Opens a help window with operands, valid ranges of values and commands.
<Ctrl>-<↓> or <Ctrl>-<↑>	The cursor is positioned at the error location in the editor.
<Esc>	To close the error window and return to the editor.

5.4 On-line functions (On-line)

The on-line functions are used for transferring the created program together with variables, strings, etc. to the controller. A number of options for program testing are also provided.

Access the on-line functions by pressing <Alt>-<O>.

Fig. 5-36 On-line functions

5.4.1 The “Transfer” pull-down menue

This option is used for data transfer. The required settings for the data transfer must also be made here.

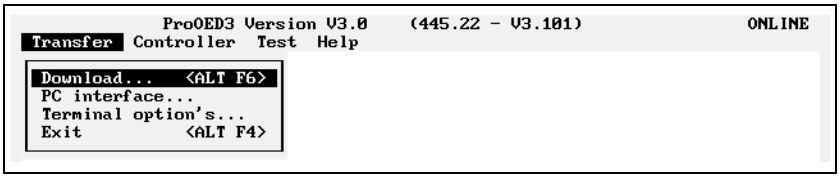


Fig. 5-37 Pull-down menue “Transfer”

Menue option	**Function**
Download	To transfer the program and all pertaining data to the controller (see chapter 5.4.1.1).
PC interface	To select the serial interface.
Terminal options	Further settings for the data transfer.
Exit	To exit the on-line functions and return to the editors.

5.4.1.1 Loading a project into the controller (Download)

ATTENTION
Before switching off power supply the programme transfer must be completed.
Important program data will otherwise be destroyed which can only be restored by the Berger Lahr Service.

In order to execute a program on the controller, it must be loaded into the controller first. Proceed as follows:

1. Select the menue option “Transfer/Download”.

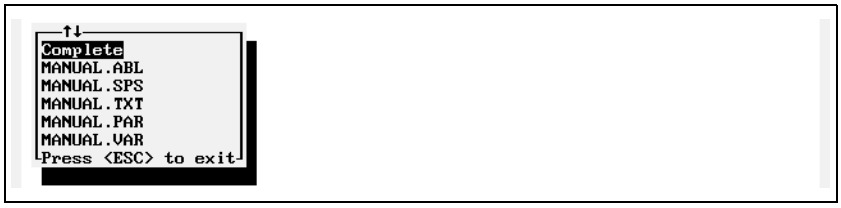


Fig. 5-38 Option selection

2. Use the <↑> and <↓> keys to select the appropriate option and press the <↵> key. The following table shows you which portions are transferred to the controller.

Option	**Meaning**
Complete	Transfer the entire project.
XXXXXXXX.ABL	SEQUENCE program
XXXXXXXX.SPS	PLC program
XXXXXXXX.TXT	Send texts
XXXXXXXX.PAR	Control parameters
XXXXXXXX.VAR	Position variables

NOTE
The string “XXXXXXXX” represents the project name. If a series of question marks (“?”) are displayed instead, the corresponding program component must first be compiled.

3. The program is then transferred to the controller. If any errors should occur, check the wiring.

5.4.2 The "Controller" pull-down menue

This menue offers options for starting or stopping the controller in automatic mode and for saving the control program in the EEPROM.

NOTE
To use this option, a control program should be loaded in the controller.

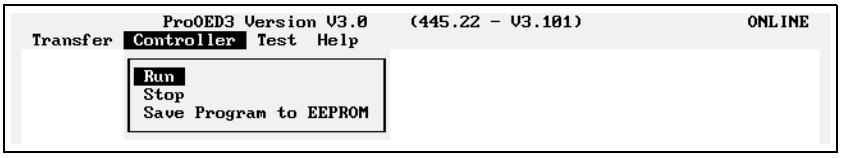


Fig. 5-39 On-line/Controller

Menue option	Function
Run	To start controller automatic mode.
Stop	To stop controller automatic mode.
Save Program to EEPROM	To save the control program in the installed controller EEPROM.

5.4.3 Various test functions

This pull-down menue offers various options for program testing.

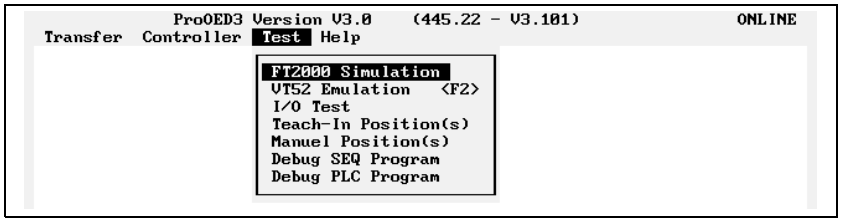


Fig. 5-40 On-line/Test

Menue option	Function	Chapter
FT2000 Simulation	To emulate an FT2000 on the screen.	5.4.3.1
VT52 Emulation	Terminal emulation with VT52.,	
I/O Test	To test the input/output wiring.	5.4.3.2
Teach-In Position(s)	To manually move the stepping motor and read current positions into position variables.	5.4.3.3
Manual Position(s)	To perform a manual movement.	5.4.3.4
Debug SEQ Program	To view current results, variables, flags, etc. during automatic mode execution. Step by step execution of the program.	5.4.3.5
Debug PLC Program		

5.4.3.1 FT2000 Simulation

This option is used for simple testing of programs addressing the FT2000. Texts are output in the simulated display of the FT2000 in the same way as if the FT2000 were connected.

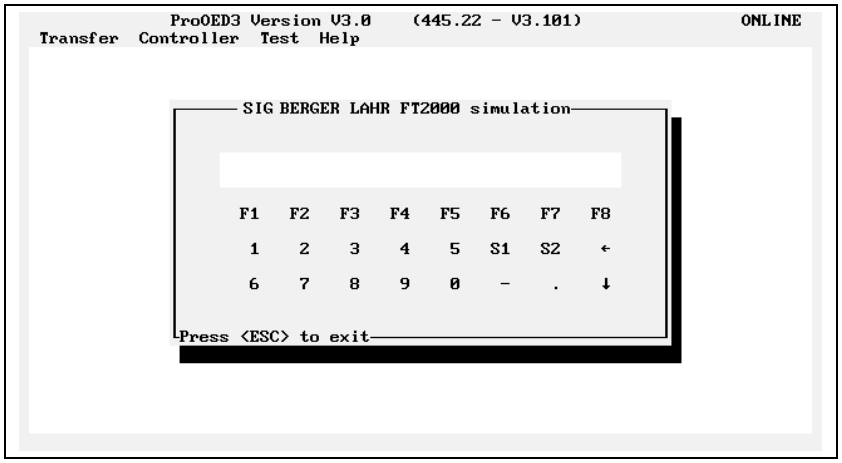


Fig. 5-41 FT2000 simulation

Operation

The following table shows the key assignment on the PC for FT2000 simulation.

Key on FT2000	Key on PC
<F1> to <F8>, <0> to <9>, <->, <.>	Corresponding keys on the keyboard.
<S1> and <S2>	Not required on the PC.
<←>	<Backspace>
<↓>	<↵>

NOTE
Pressing the <X> key and the <↵> key starts the controller in automatic mode.

5.4.3.2 I/O Test

An I/O test can be performed for checking the inputs and outputs:

1. Press <Alt>-<T>.
2. Use the <↓> key to move the highlight to “I/O Test” and press the <↵> key.

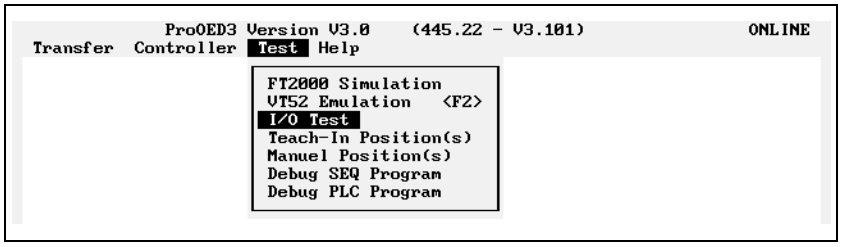


Fig. 5-42 Invoking I/O test

Next to “I000-...”, ProOED3 shows the states of the inputs from left to right. This allows you to check the connected inputs easily.

Next to “Q000-...”, the states of the outputs are displayed:

1. Use the <←> and <→> keys to move the cursor (see figure).
2. Press the spacebar to change the output status.
3. Observe the status indicators for the outputs on the front panel of the controller.

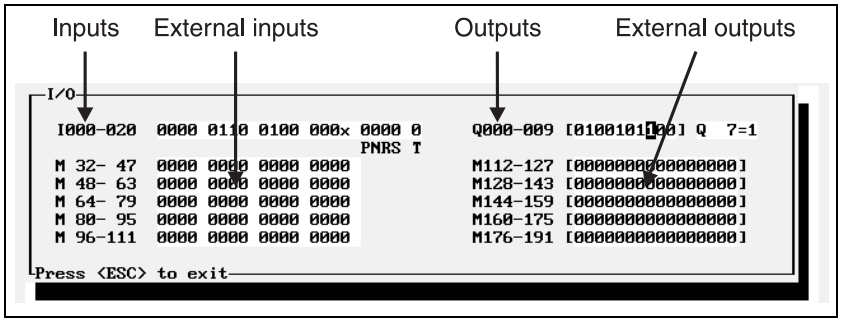


Fig. 5-43 I/O test

NOTE
On WDP3-014/018 controllers, only the inputs I0 to I8, I10, I11, I12 (LIMP), I13 (LIMN), I14 (REF.) and I15 (STOP) as well as the outputs Q0-Q3 are indicated. External I/O modules cannot be controlled with these controllers.

4. Exit the I/O test by pressing <Esc>.

I/O modules (Series 300 only)

If external I/O modules of type MP 926 are connected to a Series 300 controller and adjusted to the appropriate parameter settings (see chapter 6.1.2), M32 to M111 show the states of the inputs and M112 to M191 the states of the outputs (see chapter 6.3.3). The output states of the I/O cards can also be changed by pressing the spacebar.

5.4.3.3 Reading in positions (teach-in positions)

Teach-in allows you to read current stepping motor positions into up to 400 position variables (w100 to w499). The positions read in can then be used in the program.
A prerequisite for teach-in mode is that a suitable program is loaded in the controller in order to move the stepping motors and select the axes. You can use the supplied sample project “TEACH-IN.PO3” or “MANUAL.PO3” for this purpose.
The positions are approached manually via inputs and stored in position variables with ProOED3.
After teach-in, the position variables can be transferred from the controller to the PC (upload).

ATTENTION
On the WDP3-014/018, position variables input by teach-in are lost at power-off unless they are saved to the project with “Upload” after teach-in.

Proceed as follows:

1. Open the project “MANUAL.PO3”.
2. Press <Alt>-<2> to change to the PLC editor.
3. Compile the PLC program by pressing <Alt>-<F1>.
4. Set the desired control parameters.
5. Press <Alt>-<O> to open the on-line functions.
6. Press <Alt>-<T> to open the pull-down menue “Test”.
7. Press <T> to select “Teach-In Position(s)” (the controller must be in editing mode).

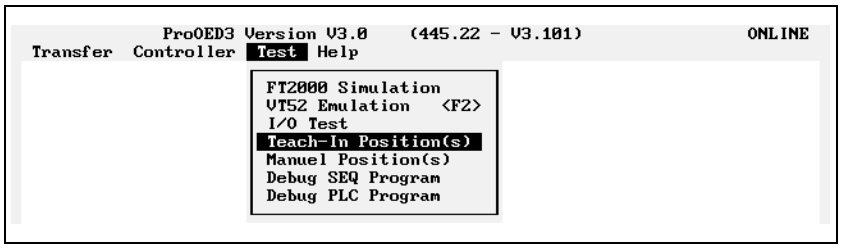


Fig. 5-44 Teach-in

8. If a suitable program for moving the motor is already loaded in the controller, press the <↵> key and continue with step 10.
   Otherwise press the spacebar to answer the prompt with “No” and press the <↵> key. This opens the selection window for “Download”.

ATTENTION
Before switching off power supply the programme transfer must be completed.
Important program data will otherwise be destroyed which can only be restored by the Berger Lahr Service.

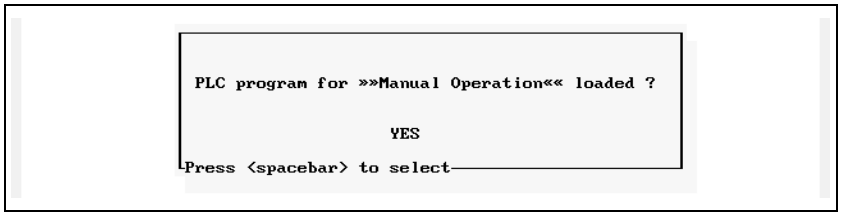


Fig. 5-45 Program transferred (Yes/No)?

9. The selection window for “Download” is displayed on the screen. The selection bar is positioned on “Complete”.
   => Press the <↵> key to transfer the entire project to the controller.

10. The “Teach-In” window is displayed on the screen. The table below contains information on using it.

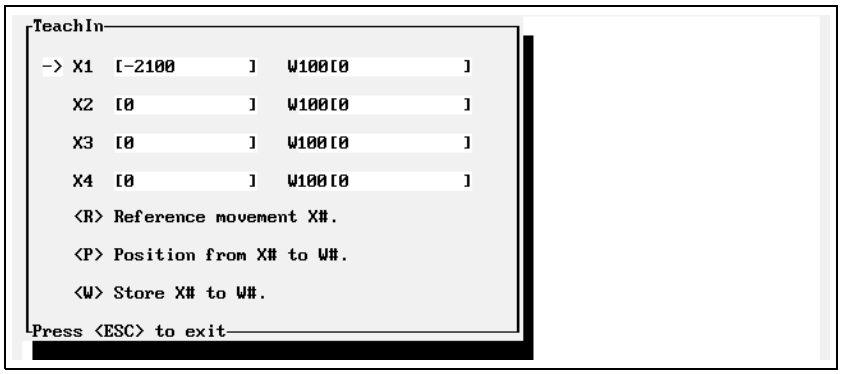


Fig. 5-46 Teach-in

Key	Function
<R>	The controller performs a reference movement.
<P>	The stepping motor moves to the position of the set position variable.
<W>	The current stepping motor position is stored in the set position variable.
<↓> and <↑>	To select the position variable for the current axis.
<Esc>	Close the window.

The way the stepping motor moves and the current axis is selected depend on the program in the controller. The supplied sample program “MANUAL.PO3” can be used for operating the controller as follows:

Move the stepping motor and select the speed via I0 to I2:

Input	Function
I0	Move the motor to the left
I1	Move the motor to the right
I2	Select slow or fast speed

Select the current axis via I3 and I4:

Axis	Input signal status	
	I3	I4
x1	0	0
x2	0	1
x3	1	0
x4	1	1

Close the “Teach-In” window by pressing <Esc>. The position variables read in must then be transferred to the PC.
Cancel by pressing <Esc>, transfer the position variables to the PC by pressing <↵>.

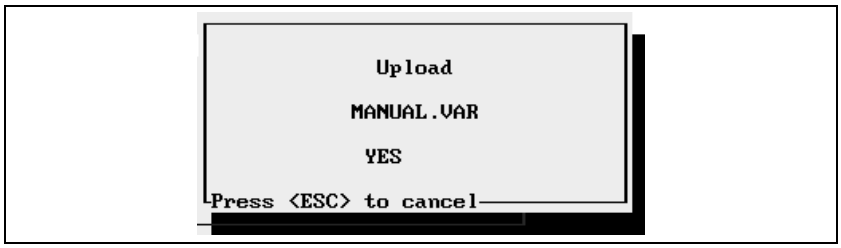


Fig. 5-47 Position variables transferred to PC

When transferring position variables, the following window should be displayed:

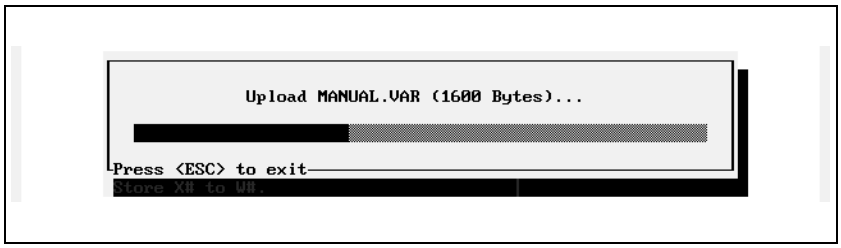


Fig. 5-48 Teach-in

5.4.3.4 Displaying positions during a manual movement

The “Manual Position(s)” option can be used for performing a manual movement. The current stepping motor positions are displayed on the screen.
A prerequisite for this mode is that a suitable program is loaded in the controller in order to move the stepping motors and select the axes.
You can use the supplied sample projects “TEACH-IN.PO3” or “MANUAL.PO3” for this purpose.
For operating instructions for moving the motor and selecting the axis, refer to chapter 5.4.3.3, “Teach-in”.

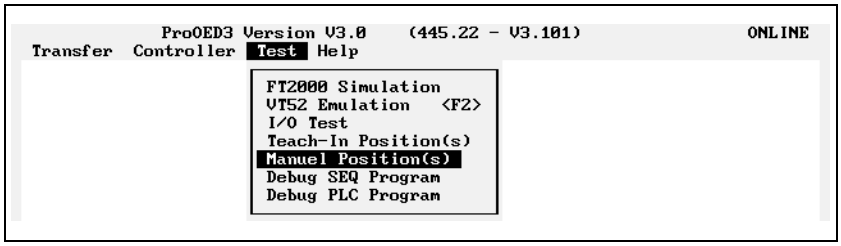


Fig. 5-49 Performing manual movements

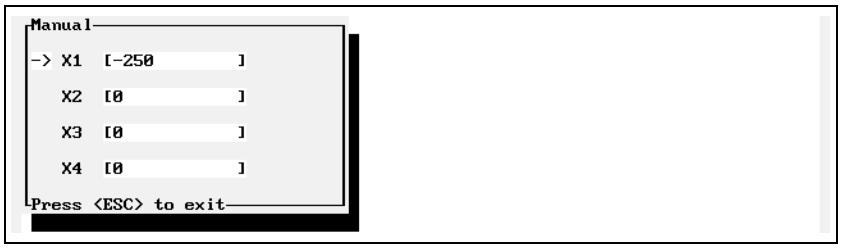


Fig. 5-50 Manual movement

5.4.3.5 Debugging the SEQUENCE and PLC program

This option can be used for displaying the SEQUENCE or PLC program and their current results on the screen during automatic execution. The programs can be executed step by step.

1. Select either “Debug PLC Program” or “Debug SEQ Program”.

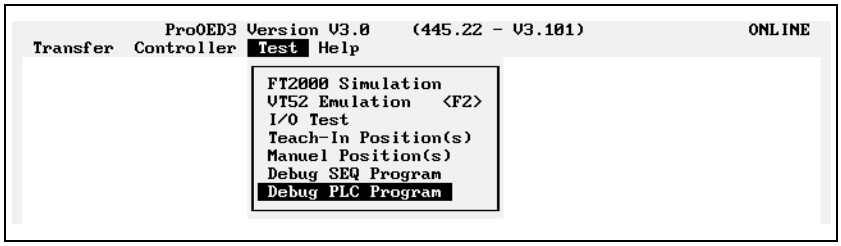


Fig. 5-51 Debugging

2. Answer the prompt “Project loaded into controller?” as appropriate and, if necessary, transfer the program to the controller.

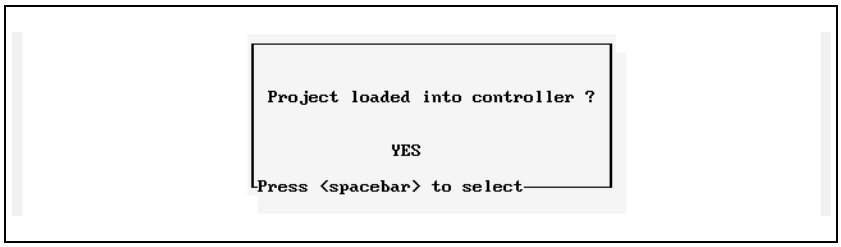


Fig. 5-52 Project loaded into controller?

STEP mode

The following debugging window is displayed and the controller changes to STEP mode.

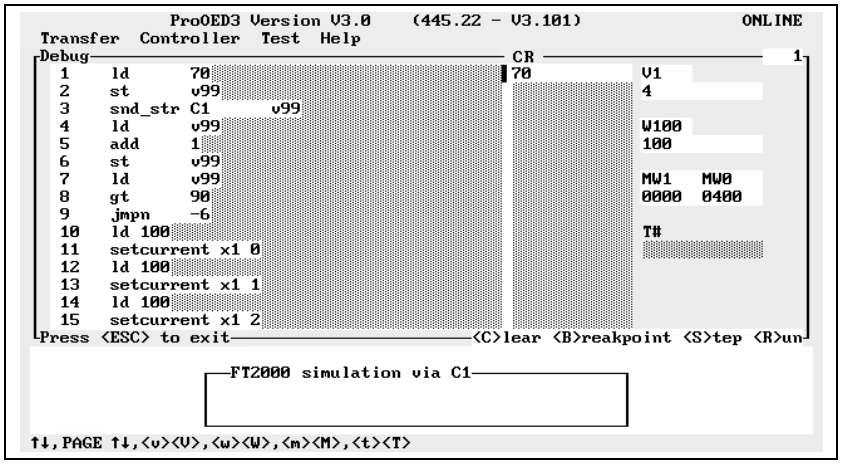


Fig. 5-53 Debugging window in STEP mode

Debugging window

The following information is displayed in the debugging window:
– Program commands of the program to be debugged
– CR (current result) for each program command
– Variables, flags, timers
– Debugging commands
– Window for FT 2000 simulation

Debugging commands

Keys	Function
<C>lear	Clear a breakpoint set with <B>.
<B>reakpoint	Set a breakpoint.
<S>tep	Change over from continuous operation to single STEP mode; execute single step.
<R>un	Change over from single STEP mode to continuous operation.

Displaying variables, flags and timers

Keys	Function
<v>	Increment number of variable displayed.
<SHIFT><v>	Decrement number of variable displayed.
<m>	Increment number of flag displayed.
<SHIFT><m>	Decrement number of flag displayed.
<t>	Increment number of timer displayed.
<SHIFT><t>	Decrement number of timer displayed.
<w>	Increment number of position variable.
<SHIFT><w>	Decrement number of position variable.

Scrolling in the program

Keys	Function
<Page Up>	Scroll the program upwards page by page.
<Page Down>	Scroll the program downwards page by page.

To execute the commands, use the keys in brackets.

RUN mode

When changing over to RUN mode (<R>un command), the debugging window changes slightly.

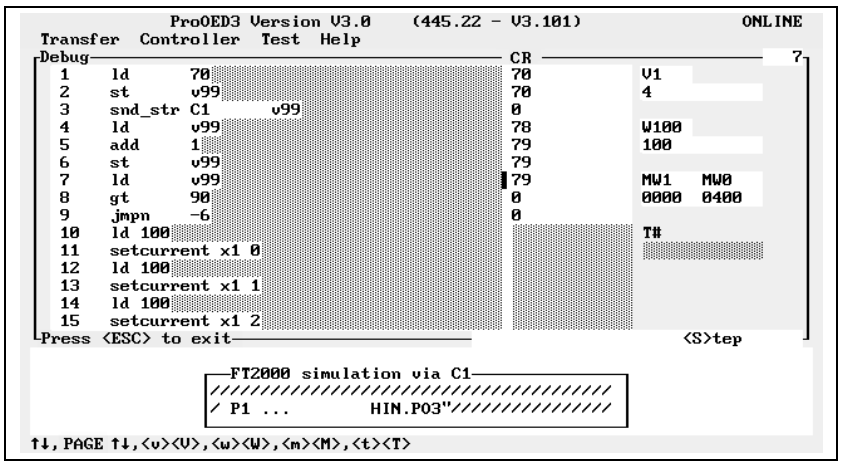


Fig. 5-54 Debugging window in RUN mode

In RUN mode, you can only execute the <S>tep command to change back to STEP mode.

NOTE
As a consequence of the CR (current result) being displayed, program execution in debugging mode takes approx. 100 times longer than without debugging mode.

Program execution in debugging mode

In debugging mode, the program can be executed in either of two modes:

– Single step (STEP mode)
– Continuous operation (RUN mode)

In STEP mode, you can execute the program line by line and debug it using the <S>tep key.
When pressing the <R>un key, the program is executed in RUN mode, i.e. in continuous operation.
You can use the <B>reakpoint key to specify a program line where the program is to be interrupted in RUN mode. When the program detects a breakpoint, it changes back to STEP mode.
You can use the <C>lear key to clear a breakpoint.

NOTE
Only one breakpoint can be set at a time.

FT 2000 simulation in debugging mode

The debugging window comprises a small window at the bottom for simulating an FT 2000 operating terminal. You can use this window for testing the programmed communication for an operating terminal with the PC; it is not necessary to have a real operating terminal connected. During operating terminal simulation, the input via the PC keyboard is used for the simulation. In this situation, you cannot make any input or action in the debugging window.

5.5 Help

This option offers you information on operating the system and on the system itself.

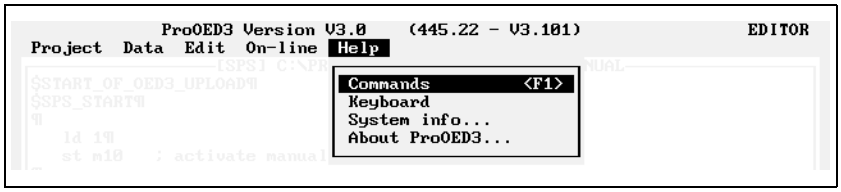


Fig. 5-55 Help

Menue option	**Function**
Commands	A list of the keyboard commands.
System info	System information such as: DOS version, available memory, etc.
About ProOED3	Brief information about ProOED3, e.g. version number.

6 Programming

6.1 Basic information

6.1.1 Developing a ProOED3 program

The ProOED3 programming software can be used for developing simple application programs for Series 300 controllers and for WDP3-014/018 controllers.

A ProOED3 application program (in short “program”) consists of several program components:

– Control parameters
– PLC program
– SEQUENCE program
– Position variables
– Symbolic names
– Send texts

Programming of the individual program components is effected with the appropriate editors provided (see chapter 5.2).

To execute and test the program components, they must be loaded into the controller (download).

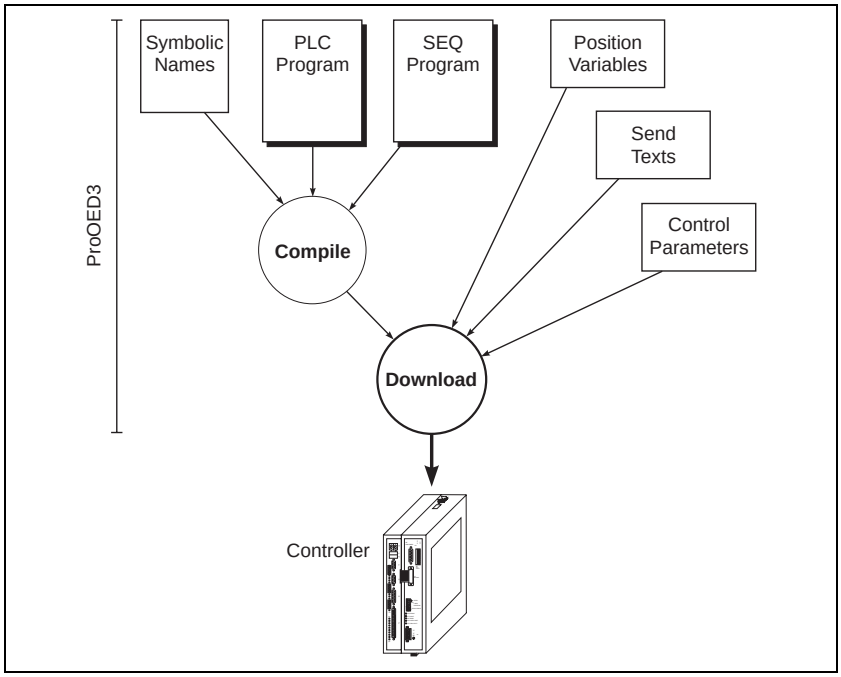


Fig. 6-1 Compiling and loading program components into controller

NOTE
The PLC program and the SEQUENCE program must be compiled before loading them into the controller.

For program testing, several features are available for selection:

- **PLC debugger**,
  for debugging the PLC program
- **SEQ debugger**,
  for debugging the SEQUENCE program
- **I/O test**,
  for testing the I/O wiring
- **FT 2000 simulation**,
  for testing operator dialogs for the FT2000 operating terminal
- **VT52 emulation**,
  for testing operator dialogs for a VT52 terminal

NOTE
It is not required to fully program all program components in order to obtain an executable project. For some applications it may be sufficient, for example, to set the control parameters and write the SEQUENCE program. However, it is indispensible to compile and load into the controller both the SEQUENCE and the PLC program.

6.1.1.1 Program development

To develop a program using ProOED3, the controller must be switched to editing mode. In this mode, programs developed with ProOED3 can be loaded into the controller (download).
A program loaded into the controller can be started from ProOED3 (Run); the controller switches to automatic mode for program execution.
When the program is stopped from ProOED3, the controller returns to editing mode.

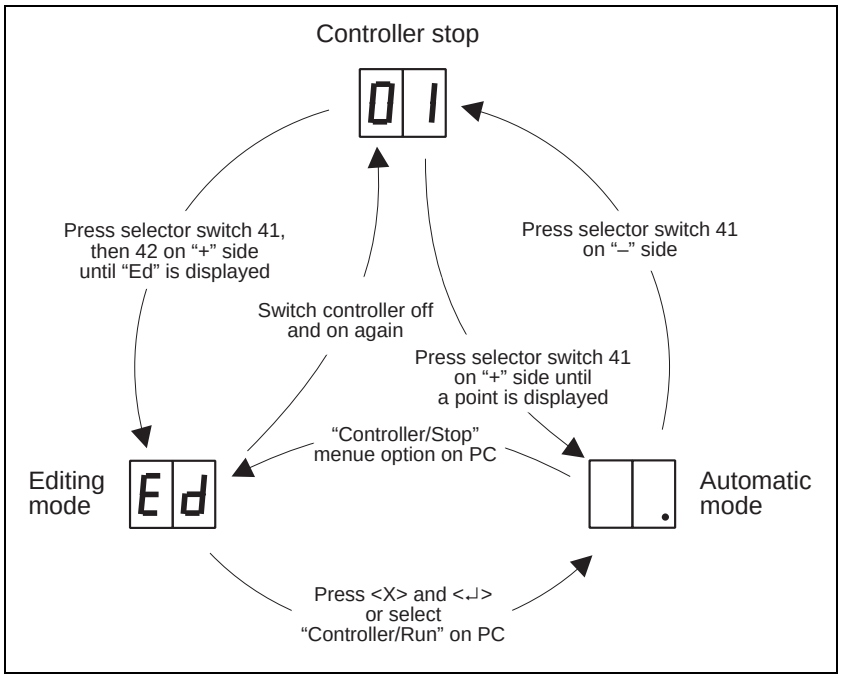


Fig. 6-2 Status display on Series 300 controllers

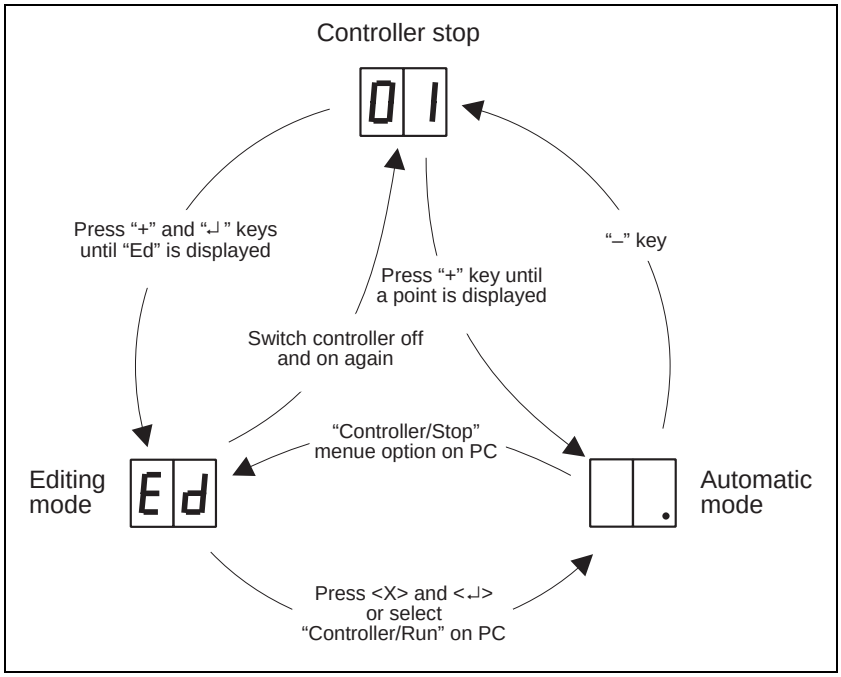


Fig. 6-3 Status display on WDP3-014/018 controllers

Procedure for program development

Prerequisites:
The controller must be wired completely and the link between the PC and the controller must be correct.

1. Start ProOED3.
2. Open a new or an existing project.
3. For a new project, set the controller type.
4. Create or modify all required program components using the appropriate editors.
5. Compile the SEQUENCE and PLC program.
6. Activate editing mode on the controller; see figure 6-2.
7. Load the program components into the controller individually or together.
8. Start the program.
9. Debug the program.
10. Save the program to EEPROM.

NOTE
Chapter 4 describes step-by-step development of a simple project in detail.

6.1.2 Control parameter setting

The control parameters are used in the **controller initialization** process. Initialization is performed at each program start. Control parameters affect motor control, axis positioning, the interfaces and the basic settings of the controller.
Control parameters are administrated with the parameter editor. They must be set specifically for each application.

NOTE
When opening a new project, the control parameters are preset to their default values.

Parameter list

Parameter	Range of values (default)	Description
Active limit switches x1 to x4	0 – 3 (3)	When a limit switch is reached, the drive is brought to a standstill using the set ramp. The controller is then in error status. 0 = No limit switch active 1 = Negative limit switch active 2 = Positive limit switch active 3 = Both limit switches active
Decimal point	1 – 3 (2)	Number of decimal places for the rec_dez and snd_dez commands.
Encoder evaluation DG1 or DG2 (DG = Encoder) (Encoder 1 only for WP-311 and WDP5-318)	0 – 5 (3)	Encoder resolution (increments/revolution) and encoder evaluation (single, double, quadruple) 0 = ’500 encoder, single evaluation 1 = ’500 encoder, double evaluation 2 = ’500 encoder, quadruple evaluation 3 = ’1000 encoder, single evaluation 4 = ’1000 encoder, double evaluation 5 = ’1000 encoder, quadruple evaluation
Encoder setting*	-1 – 2 (0)	Encoder connection usage -1 = Connection 1 (p1) not used Connection 2 (p2) for position following mode 0 = Connection 1 (p1) for position following mode Connection 2 (p2) not used 1 = Connection 1 (p1) for rotation monitoring Connection 2 (p2) for position following mode 2 = Connection 1 (p1) for position following mode Connection 2 (p2) for rotation monitoring Encoder connection 1 (p1) is not available on WDP3-314/3-318 controllers. The following settings are possible: -1 = Connection 2 (p2) for position following mode 0 = Connection 2 (p2) not used 2 = Connection 2 (p2) for rotation monitoring
Rotation monitor x1 to x4	0 – 1 (0)	0 = Disable rotation monitoring 1 = Enable rotation monitoring
External I/O modules*	0 – 5 (0)	Number of external MP926 I/O modules on RS 485 HS interface (16 inputs and 16 outputs per card)

Parameter	Range of values (default)	Description
Error handling	0 – 2 (0)	Controller response in case of an error (see also chapter 7.2) 0 = Total controller stop. 99 appears in the 7-segment display. Error menue via serial interface c1. 1 = SEQUENCE program and axis are stopped. Error code in 7-segment display. No error menue. 2 = Error handling by application program. Subprogram “L0”. No error menue.
Clearing distance limit switch	0 – 1000 steps (0 steps)	Clearing distance from limit switch or reference switch after a reference movement (position = 0).
Clearing speed limit switch	1 – 10000 Hz (200 Hz) 4 – 10000 Hz** (200 Hz)	Speed when moving out of limit switch or reference switch range.
Lauer operating panel*	0 – 8 (0)	1 = 8 data bytes of type micro 2 = 8 data bytes of type mini 3 = 8 data bytes of type midi 4 = 8 data bytes of type maxi 5 = 16 data bytes of type micro 6 = 16 data bytes of type mini 7 = 16 data bytes of type midi 8 = 16 data bytes of type maxi If either the value 5, 6, 7 or 8 has been set for “Lauer operating panel”, the “External I/O modules” parameter may only be set to 0, 1 or 2.
Manual speed – slow	1 – 10000 Hz (200 Hz) 4 – 10000 Hz** (200 Hz)	Speed for slow manual movement via flag m0 or m1.
Manual speed – fast	1 – 10000 Hz (2000 Hz) 4 – 10000 Hz** (2000 Hz)	Speed for fast manual movement via flag m2.
Max. allowed dist. limit switch	10 – 55924053 (10000)	The drive must have left again a limit switch (reference switch) within this distance.
Normalizing factor denominator x1 to x4	1 – 2147483647 (1000)	The normalizing factor is used in point-to-point mode for converting the user-defined units (e.g. mm) to drive units (steps or increments).
Normalizing factor numerator x1 to x4	-55924053 – +55924053 (1000)	
Ramp x1 to x4	0 – 3 (0)	The shape of the ramp at which acceleration and deceleration are effected. 0 = Linear ramp 1 = Exponential ramp*** 2 = Sine square ramp*** 3 = Optimized stepping motor ramp***

Parameter	Range of values (default)	Description
Gear interface signals	0 – 1 (1)	Set type of input signal at encoder input 0 = Pulse/direction signal 1 = A/B signal
Standard acceleration	1 – 2000 Hz/ms (125 Hz/ms)	If there is no acceleration preset in the application program (with the “acc” command), all axes x1 to x4 are accelerated or decelerated using this acceleration.
Standard speed	1 – 200000 Hz (1000 Hz) 4 – 40000 Hz** (1000 Hz)	If there is no speed preset in the application program (with the “vel” command), all axes x1 to x4 move at this maximum frequency.
Start/stop speed	1024 – 10000 Hz (100 Hz) 4 – 10000 Hz** (100)	The speed at which the axis is started or stopped.
System speed x1 to x4	1 – 200000 Hz (32767 Hz) 4096 – 40000 Hz** (32767 Hz)	The maximum system speed *NOTE* *To perform a movement at a speed of precisely 1 Hz with a Series 300 controller, this value must be set to 32767 Hz.*
Inching distance for manual mode	0 – 100 steps (10 steps)	The distance for momentary activation (approx. 500 ms) of the manual movement flags m0 or m1. Inching distance = 0 specifies that the motor changes to continuous operation immediately.
Type of reference movement x1 to x4	0 – 3 (0)	Specifies the limit switch to be approached in a reference movement 0 = Negative limit switch 1 = Positive limit switch 2 = CCW reference switch (as seen from front towards motor shaft) 3 = CW limit switch (as seen from front towards motor shaft)

* Series 300 only ** WDP3-014/018 only *** Only available in PTP mode

Examples

Setting	Parameter
Reference movement of axis x1 to CCW reference switch	“Type of reference movement x1” = 2
Set sine square ramp for axis x4	“Ramp x4” = 2
Monitor positive limit switch for axis x1	“Active limit switch x1” = 2

6.1.3 SEQUENCE and PLC program

The tasks of the controller are programmed in the SEQUENCE program and the PLC program.
After controller initialization (control parameters), the SEQUENCE program and the PLC program are executed alternatingly every 2 ms in such a way that the PLC program is processed for 2 ms, then the SEQUENCE program for 2 ms and so on. Data exchange between the two programs can be effected via flags (m) and variables (v, w, f). Inputs and outputs can be addressed from both programs.

NOTE
During execution of the two program components, the controller is in automatic mode.

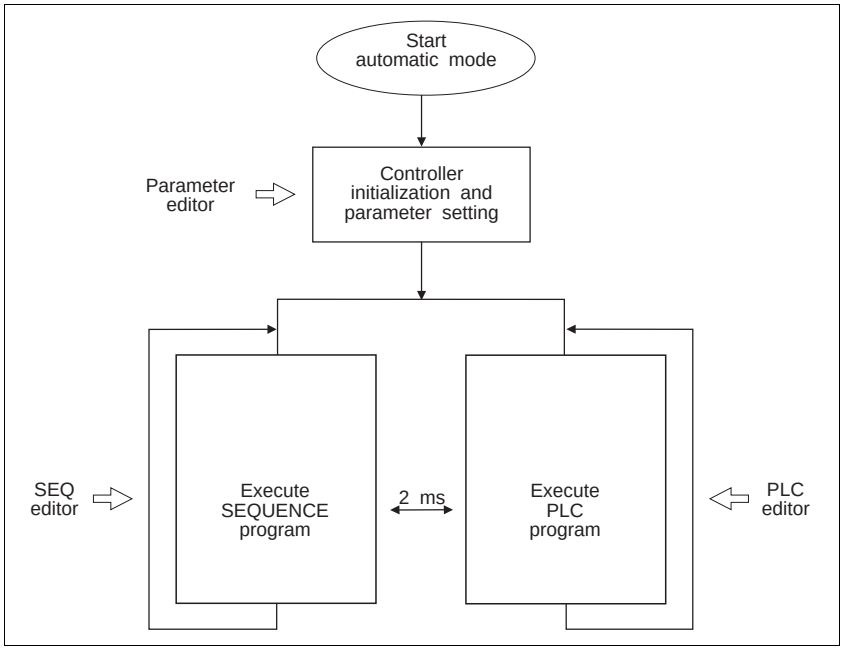


Fig. 6-4 Program execution in automatic mode

PLC program

The instructions of the PLC program are processed cyclically. However, no cycle time monitoring is performed.

The following functions can be implemented in a PLC program:
- Writing and reading data
- Logical operations
- Arithmetic operations
- Relational operations
- Direct I/O signal reading (***no process image***)
- Timer functions

Controller-specific commands are not permitted in a PLC program; the valid commands for a PLC program are listed in the table in chapter 6.1.3.2, “Operators”.
A PLC program may comprise up to 1000 lines of program code.

A PLC program is created using the PLC editor and must be compiled before loading it into the controller.

SEQUENCE program

The SEQUENCE program is also processed cyclically.
A SEQUENCE program is created using the SEQ editor and must be compiled before loading it into the controller.

In addition to PLC commands, also controller-specific commands (positioning, linear interpolation, encoder, serial interface) can be programmed in a SEQUENCE program. SEQUENCE commands are listed in the table in chapter 6.1.3.2, “Operators”.

The following additional functions can be implemented in a SEQUENCE program:

- Reference movement
- Positioning
- Linear interpolation
- Electronic gear
- Rotation monitoring
- Data exchange via the serial interface
- Synchronization with a master controller
- Direct access to I/O signals
- Subprograms

NOTE
Data exchange between PLC and SEQUENCE program can be effected using flags (m) and variables (v, w, f).

Up to 2000 lines of program code are permitted in a SEQUENCE program.

6.1.3.1 Commands

PLC programs and SEQUENCE programs consist of a sequence of commands.

Structure of a command

A command is made up of an operator and one or two operands.

Examples:

ld	100
Operator	Operand

vel	x1	100
Operator	Operand 1	Operand 2

See also chapters 6.1.3.2 and 6.1.3.3.

Current result (CR)

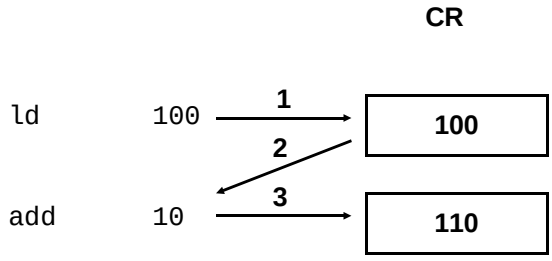
The current result (CR) is a temporary storage element (accumulator) on the controller for

– Data transfer
– Storing results
– Conditions for conditional command execution

The contents of the CR is used in commands as an operand for logical operations, relational operations, arithmetic calculations and as a condition for conditional jump instructions.

The result of a command is stored in the CR. The CR can only take a Boolean value (1-bit values) or an integer number (32-bit values).

CR	Range of values
Boolean value	0, 1
Integer number	-2147483647 to +2147483647



Examples:

```
ld      5           ;CR  <- 5
add     3           ;CR  <- CR + 3  (CR <- 8)
div     4           ;CR  <- CR \ 4  (CR <- 2)
mul     10          ;CR  <- CR * 10  (CR <- 20)
```

Execution of some commands (s, r, calc, caln, jmpc, jmpn) depends on the Boolean value (TRUE, FALSE) of the CR.

Value in CR	Boolean value of CR
0	FALSE
<> 0	TRUE

Example 1:

```
ld      i10         ;CR  <- i10
r       q5          ;Output 5 is only
                    ;reset if the
                    ;CR = 1 (TRUE).
```

Example 2:

```
ld      i10         ;CR <- i10 (input 10)
jmpn    L4          ;Jump to label L4 if
                    ;i10 = 0.
```

6.1.3.2 Operators

An operator represents the actual function of a command.
The operator defines the number and the type of the operands of a command. The following command tables list all operators with the associated operands.
The first table contains the commands available for both the PLC and the SEQUENCE program components. The second table contains the commands available for the SEQUENCE program component only.
The commands are grouped according to functional groups.
Refer to the appendix for a more detailed description of the commands.

Commands for PLC and SEQUENCE program components

Functional group	**Commands Operator <avail. operands>**		**Function**
Loading and storing	ld	<fknvwiqmxtp>	Loading or reading into the CR
	ldn	<iqm>	Loading or reading the negated value into the CR
	st	<fnvwqmxtp>	Storing
	stn	<qm>	Storing the negated value
Setting and resetting	r	<qm>	Setting output or flag to 0
	s	<qm>	Setting output or flag to 1
Timer	stimer	<t>	Start timer (only available for PLC program component)
Logical operation commands	and	<iqm>	AND operation
	andn	<iqm>	Negated AND operation (NAND)
	or	<iqm>	OR operation
	orn	<iqm>	Negated OR operation (NOR)
Arithmetic commands	add	<fknvw>	Adding
	div	<fknvw>	Dividing
	mul	<fknvw>	Multiplying
	sub	<fknvw>	Subtracting
Relational commands	eq	<fknvwiqm>	Equal to
	gt	<fknvw>	Greater than
	lt	<fknvw>	Less than
Program jump operations and flags	jmp	<lk>	Unconditional jump
	jmpc	<lk>	Conditional jump, CR = 1 (TRUE)
	jmpn	<lk>	Conditional jump, CR = 0 (FALSE)
	label	<L>	Label for program jump operation
	end		Program end and jump to program beginning
Communication with Lauer operating panel*	ld_LKey*	<knvw>	Read key status of operating panel
	ld_LBit*	<knvw> <knvw>	Read bit from data interface of operating panel
	ld_LInt*	<knvw>	Read word from data interface of operating panel
	ld_LDint*	<knvw>	Read double word from data interface of operating panel
	st_LBit*	<knvw> <knvw>	Write bit to data interface of operating panel
	st_LInt*	<knvw>	Write word to data interface of operating panel
	st_LDint*	<knvw>	Write double word to data interface of operating panel

* Series 300 only

Commands for SEQUENCE program component

Functional group	Commands Operator <avail. operands>			Function
Subprogram calls	cal	<L>		Unconditional subprogram call
	calc	<L>		Conditional subprogram call, CR = 1 (TRUE)
	caln	<L>		Conditional subprogram call, CR = 0 (FALSE)
	ret			Return from subprogram
Axis operating mode	mode	<x>	<k>	Set the axis operating mode
Positioning in point-to-point mode	vel	<x>	<fknvw>	Set the set speed
	acc	<x>	<fknvw>	Set the acceleration ramp
	move	<x>	<fknvw>	Relative positioning without waiting
	movef	<x>	<fknvw>	Relativ positioning, and wait until position reached
	pos	<x>	<fknvw>	Absolute positioning without waiting
	posf	<x>	<fknvw>	Absolute positioning, and wait until position reached
	ref	<x>		Reference movement without waiting
	reff	<x>		Reference movement, and wait until position reached
	stop	<x>		Stop axis x
	stopa*	<x>		Stop all axes
	setsiglist*	<x>	<knvw>	Activate position list
	settrigger*	<x>	<k>	Position trigger
Electronic gear in position following mode	gearn	<x>	<fknvw>	Set gear ratio denominator
	gearz	<x>	<fknvw>	Set gear ratio numerator
	goff	<x>	<fknvw>	Set position offset
Linear interpolation for multi-axis controllers	linpos*			Start linear interpolation (absolute)
	linposf*			Start linear interpolation (absolute), and wait until position reached
	linmove*			Start linear interpolation (relative)
	linmovef*			Start linear interpolation (relative), and wait until position reached
	setipos*	<x>	<knvw>	Prepare linear interpolation
Communication via serial interface	cursor	<c>	<fknvw>	Position the cursor
	rec_char	<c>	<fnvw>	Receive character
	rec_char_n	<c>	<fnvw>	Receive character and check
	rec_dez	<c>	<fnvw>	Receive and store decimal numbers
	rec_var	<c>	<fnvw>	Output and edit number
	rec_var_n	<c>	<fnvw>	Receive and store number
	screen	<c>	<fknvw>	Screen control
	snd_char	<c>	<fknvw>	Output character
	snd_dez	<c>	<fnvw>	Output decimal numbers
	snd_str	<c>	<fknvw>	Output string
	snd_var	<c>	<fnvw>	Send number
Analog output	getanalog*	<a>	<knvw>	Read analog voltage
	setanalog*	<a>	<knvw>	Output analog voltage

Functional group	Commands Operator <avail. operands>			Function
Miscellaneous commands	amp	<x>	<k>	Switching the power controller on and off
	brake*	<x>		Activate output for brake
	clrerror	<x>		Reset axis signal error
	getport	<im>	<fknvw>	Read inputs or flags and convert to integer number
	handshake	<i>	<q>	Synchronization with master controller
	restart			Restart SEQUENCE program
	setcurrent	<x>	<fknvw>	Set the motor current
	wait	<fknvw>		Suspend program execution for a specific time (ms)
	wsave			Save position variables
	nop			Dummy command (500 µs to 1000 µs execution time)

* Series 300 only

6.1.3.3 Operands <ac ... wx>

Commands can take one or two operands (see table in chapter 6.1.3.2). The operands contain the values required for command execution.

The following operand types may occur:

Operand	Function		Range of values
a	**Analog interface**		Input: ± 10000 mV Output: + 10000 mV
	a2*	Analog module	
c	**Serial interface**		
	c1	Interface 1	
	c2*	Interface 2	
f**	**FRAM variables (only for WDP3-014/018 with OED3)**		- 2147483647 to +2147483647
	f1, f2, f3		
i	**Inputs**		0 or 1
	i0 to i8, i10, i11	WDP3-014/018	
	i0 to i20	For single-axis controllers (Series 300)	
	i0 to i30/40	For multi-axis controllers (Series 300) i30: for WDPM3-314 i40: for WPM-311	
k	**Constants (value)**		- 2147483647 to +2147483647
l	Labels L0 to L100 (L0 reserved for error handling by application program)		

Operand	Function					Range of values
m	**Flags**					0 or 1
	m0	Manual movement to the right (m0 = 1)				
	m1	Manual movement to the left (m1 = 1)				
	m2	Rapid speed				
	m3 m4	Multi-axis selection bit 0 Multi-axis selection bit 1	Axis x1 x2 x3 x4	m4 0 0 1 1	m3 0 1 0 1	
	m5 to m9	Reserved				
	m10	Activate manual movement: 0 = Inactive, 1 = Active				
	m11	Manual movement active? 0 = No, 1 = Yes				
	m12 to m20	Reserved				
	m21 to m999 m32 to m111 m112 to m191	Freely available; For controllers with external MP 926 I/O modules: For the external inputs i 32 to i 111 For the external outputs q 112 to q 191				
	m1001	Axis x1: 0 = Axis stopped, 1 = Axis positions				
	m1002	Axis x2: 0 = Axis stopped, 1 = Axis positions				
	m1003	Axis x3: 0 = Axis stopped, 1 = Axis positions				
	m1004	Axis x4: 0 = Axis stopped, 1 = Axis positions				
	m1005 to m1010	Reserved				
	m1011*	Trigger x1: 1 = Trigger signal from axis 1				
	m1012*	Trigger x2: 1 = Trigger signal from axis 2				
	m1013*	Trigger x3: 1 = Trigger signal from axis 3				
	m1014*	Trigger x4: 1 = Trigger signal from axis 4				
	m1015**	0 = Invalid FRAM variables 1 = Valid FRAM variables				
	m1016 to m1023	Reserved				
n	**Indirect indexing of variables**					1 to 499
	n1 to n99	Indirect access via v1 to v99 and w100 to w499				
p	**Encoder**					
	p1	Encoder 1 (only for WP-311 and WDP5-318)				
	p2	Encoder 2				
q	**Outputs**					0 or 1
	q0 to q3	WDP3-014/018				
	q0 to q9	Series 300				
t	**Timer** t0 to t9 (resolution 100 ms)					0 to 864000

Operand	Function		Range of values
v	**Variables**		-2147483647 to +2147483647
	v0	Axis number in case of error in SEQUENCE program	
	v1 to v99	Freely available	
	v100	Error code for the 7-segment display	
	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	
	v105 to v110	Reserved	
	v111*	Trigger position for axis x1	
	v112*	Trigger position for axis x2	
	v113*	Trigger position for axis x3	
	v114*	Trigger position for axis x4	
w	**Position variables**		-2147483647 to +2147483647
	w100 to 499	Axes x1, x2, x3, x4	
x	**Axis**		
	x1	For single-axis controllers	
	x1 to x4	For multi-axis controllers	

* Series 300 only ** WDP3-014/018 only

NOTE
Fixed-point numbers are invalid for operands.

Command description

The operands available for an operator are specified by the abbreviation of the operand type in the command description.

Example 1:

```
and      <iqm>
```

The “and” operator takes one operand. Valid operand types are only **i** (input), **q** (output) or **m** (flag).

Example 2:

```
move     <x>      <fknvw>
```

The “move” operator takes two operands. The first operand **x** defines the axis. The second operand may only be of type **f** (FRAM variable), **k** (constant), **v** (variable), **w** (position variable) or **n** (index variable).

Constants

Constants (k)

Constants are used for specifying values in a command directly.
Constants are only integer numbers with a range of values from -2147483647 to +2147483647.

Example 1:

```
ld        5000        ;This multiplication
                      ;produces an incorrect
                      ;result since the range of
mul       10000000    ;values of the CR is
                      ;exceeded.
```

Example 2:

```
jmp       -10         ;Jump 10 program lines
                      ;back.
```

Example 3:

```
snd_str   c1  10      ;Transmit send text 10
                      ;via the serial
                      ;interface.
```

Variables

Variables (v)

Variables are storage elements for temporary storage of values.
99 variables (v1 to v99) are freely available; they can be used for calculations or as operands for positioning commands.
Variables are 32-bit variables with a range of values from -2147483648 to +2147483647. Only integer numbers can be stored in variables.

Some variables are required for special purposes.

Variable v0

The variable v0 receives the number of the corresponding axis in case of an error in the SEQUENCE program.
In case of an error, you can still use the command

```
movef    v0       <position>
```

to move the axis.

Variable v100

The variable v100 receives an error code in case of an error (see chapter 7.2: “Runtime errors”).

Variables v101 to v104

The variables v101 to v104 can be used as reference variables in position following mode (see “mode” command).

Variables v111 to v114 (Series 300 only)

The variables v111 to v114 are used for storing the trigger positions during a trigger movement (see “settrigger” command).

Variables	**Function**
v0	Axis number in case of an error in the SEQUENCE program
v1 to v99	Freely available
v100	Error code for 7-segment display
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
v105 to v110	Reserved
v111*	Trigger position of axis x1
v112*	Trigger position of axis x2
v113*	Trigger position of axis x3
v114*	Trigger position of axis x4

* Series 300 only

NOTE
If the controller has a battery installed, the variables are protected against voltage failure, i.e. they retain their values also when the controller is switched off. WDP3-014/018 controllers do not have a battery, i.e. the variables v0 to v99 are initialized with 0 at controller start.

NOTE
Data exchange between PLC program and SEQUENCE program can be effected via variables and flags since both program components can access variables.

Programming example:

```
;CR = v1 + v2 (CR = 10 + 20)
ld   10   ;Load the value 10 into the CR CR <-10
st   v1   ;Store contents of CR in
          ;variable v1; v1 <- 10
ld   20   ;Load the value 20 into the CR
st   v2   ;Store contents of CR in ;variable v2
          ;v2 <- 20
ld   v1   ;Load the variable v1 (10) into CR
          ;CR <- 10
add  v2   ;Add the variable v2 (20) to
          ;the value in the CR; CR <- 30
```

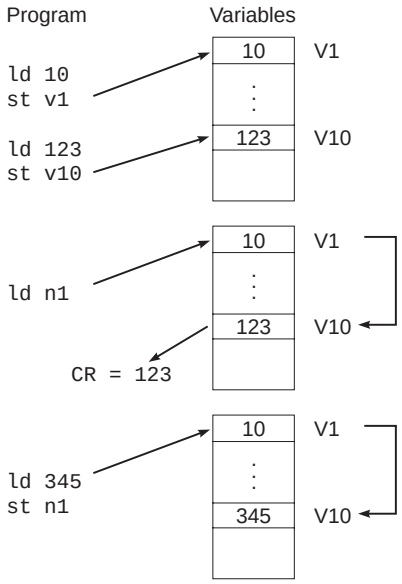
Indirect access

Indirect access (n) to variable

The n operand can be used for accessing variables indirectly. The value following n defines the variable to be used for accessing. This allows you to write short programs, for example, for copying variables using a program loop.

Example:

n1 means indirect access via variable v1, i.e. the value of the variable v1 is the address of the variable to be read or written.



NOTE
Indirect access is possible for the variables v1 to v99 and w100 to w499.

Programming example 1:

The four input signals i0 to i3 are used as a binary code index for the position variables of the axis x1.

```
label      L1
getport    i0    4     ;Read in index value,
                       ;formed by the
                       ;inputs i0 to i3
                       ;(binary code)
add        100         ;Index + 100 (index points
                       ;to the position variables
                       ;of axis x1)
st         v10         ;Store as an index value
                       ;in v10 (n10)
posf       x1    n10   ;The variable v10 is used
                       ;for indirect position
                       ;variable w(100+index)
jmp        L1          ;access and positioning
```

Programming example 2:

Variables v10 to v20 to be copied to the variables v50 to v60. For this purpose indirect access via variables v1 and v2 is used.

```
ld          10          ;Load the value 10 into the
                        ;CR; CR <- 10
st          v1          ;Store contents of the
                        ;CR in the variable v1
                        ;v1 <- 10
ld          50          ;Load the value 50 into the
                        ;CR ;CR <- 50
st          v2          ;Store contents of the
                        ;CR in the variable v2
                        ;v2 <- 50

LABEL       L1          ;Label L1
ld          n1          ;Value from variable v10
                        ;to be loaded indirectly
                        ;via v1; CR <- v10
st          n2          ;Value from variable v10
                        ;to be stored indirectly
                        ;via v2; v50 <- CR
ld          v1          ;Increment variable v1
add         1
st          v1
ld          v2          ;Increment variable v2
add         1
st          v2
eq          60          ;Check for loop end
jmpn        L1          ;Jump to label L1 if
                        ;v2 < 60
```

FRAM variables (WDP3-014/018 only)

FRAM variables (f)

If a power supply undervoltage is detected on the WDP3-014/018 controllers (e.g. power-off), or if you change to STOP status or editing mode, the variables f1, f2 and f3 are automatically saved to the FRAM. At power-on, the contents of the variables are automatically set to the values they had before the voltage failure. The variables f1, f2 and f3 can be used as ordinary variables in the program.

ATTENTION
After power-off, the contents of the FRAM variables is retained only for a certain period:
– at an ambient temperature of 30°C: approx. 250 days
– at an ambient temperature of 50°C: approx. 60 days
The status of the flag m1015 = 1 indicates whether the contents of the FRAM variable is still valid.

Position variables

Position variables (w)

Position variables are special variables for storing positional values.
400 position variables are available (w100 to w499).
Position variables are used for teach-in and for position-dependent setting/resetting of an input (see “setsiglist” command).

NOTE
Position variables with no positions assigned may be used for other purposes.

NOTE (WDP3-014/018 only)
When switching on the controller (power-on), the values saved with “wsave” are reloaded into the position variables.

Position variables can also be edited with the “Position Variables” editor.
Position variables can be used as operands in positioning commands.

Programming examples:

```
movef    x1    w100   ;Axis x1 relative
                      ;positioning. The position
                      ;variable w100 is used as
                      ;the relative position.
movef    x2    w299   ;Axis x2 relative
                      ;positioning. The position
                      ;variable w299 is used as
                      ;the relative position.
posf     x3    w300   ;Axis x3 absolute
                      ;positioning. The position
                      ;variable w300 is used as
                      ;the absolute position.
posf     x4    w410   ;Axis x4 absolute
                      ;positioning. The position
                      ;variable w410 is used as
                      ;the absolute position.
```

Flags **Flags (m)**

Flags are 1-bit storage elements internal to the controller.
1024 flags m0 to m1023 are available on the controller. Some of the flags are reserved for specific functions.

Operand	Function			
m0	Manual movement to the right (m0 = 1)			
m1	Manual movement to the left (m1 = 1)			
m2	Rapid speed (m2 = 1)			
m3	Multi-axis selection bit 0	Axis	m4	m3
m4	Multi-axis selection bit 1	x1	0	0
		x2	0	1
		x3	1	0
		x4	1	1
m5 to m9	Reserved			
m10	Activate manual movement: 0 = Inactive, 1 = Active			
m11	Manual movement active? 0 = No, 1 = Yes			
m12 to m20	Reserved			
m21 to m999 m32 to m111 m112 to m191	Freely available; For controllers with external MP 926 I/O modules: For the external inputs i32 to i111 For the external outputs q112 to q191			
m1001	Axis x1: 0 = Axis stopped, 1 = Axis positions			
m1002	Axis x2: 0 = Axis stopped, 1 = Axis positions			
m1003	Axis x3: 0 = Axis stopped, 1 = Axis positions			
m1004	Axis x4: 0 = Axis stopped, 1 = Axis positions			
m1005 to m1010	Reserved			
m1011*	Trigger x1: 1 = Trigger signal of axis 1			
m1012*	Trigger x2: 1 = Trigger signal of axis 2			
m1013*	Trigger x3: 1 = Trigger signal of axis 3			
m1014*	Trigger x4: 1 = Trigger signal of axis 4			
m1015**	0 = Invalid FRAM variables 1 = Valid FRAM variables			
m1016 to m1023	Reserved			

* Series 300 only

Programming example 1:

```
;AND operation of flags m201 and m200
ld       1        ;Load the value 1 into the CR;
                  ;CR <- 1.
st       m200     ;Store contents of CR in
                  ;flag 200; m200 <- 1.
ld       0        ;Load the value 0 into the CR;
                  ;CR <- 0.
st       m201     ;Store contents of CR in
                  ;flag 201; m201 <- 0.
and      m200     ;CR and m200 (m201 and m200)
                  ;CR <- 0
```

Programming example 2:

```
;Monitoring axis x1 with flag m1001
LABEL    L1       ;Label L1
ld       m1001    ;Load flag 1001 (axis status
                  ;of axis x1) into the CR;
jmpn     L2       ;Jump to label L2 if
                  ;axis x1 stopped (m1001 = 0).
jmp      L1       ;Jump to L1 while axis x1
                  ;positions.
LABEL    L2       ;Label L2
```

Inputs/outputs

Inputs (i) and outputs (q)

For reading inputs or setting outputs, the identifier **i** is used for input signals and **q** for output signals.
The number of inputs which can be used depends on the controller.

NOTE
On controllers with external inputs/outputs via the input/output card MP 926, the external I/O signals are assigned specific flags.

Inputs	
i0 to i8, i10, i11	WDP3-014/018
i0 to i20	For single-axis controllers (Series 300)
i0 to i30/40	For multi-axis controllers (Series 300) i30: for WDPM3-314 i40: for WPM-311
Outputs	
q0 to q3	WDP3-014/018
q0 to q9	Single and multi-axis controllers (Series 300)

NOTE
On controllers with external inputs/outputs via the MP 926 input/output card, the external I/O signals are assigned specific flags. WDP3-014/018 controllers cannot use external I/O's.

Axes **Axes (x)**

Positioning commands, or commands for movement control, require as an operand the identifier of an axis for which the command is to be executed.
The following axis identifiers are available, depending on the controller:

Axis	
x1	For single-axis controllers
x1 to x4	For multi-axis controllers

Labels **Labels (L)**

Labels **L** are used as the destinations in jump instructions and subprogram calls.
In the PLC and SEQUENCE programs, the labels L0 to L100 may be used.

NOTE
A jump to label L0 is automatically executed in the SEQUENCE program if error handling is programmed by the user; see chapter 7.2.3, "Error handling by application program".

Label L0 to L100 (L0 reserved in case of error handling by user)

6.1.3.4 Comments

Comments may be inserted at any place in the program.
Comments must start with a semicolon (**;**) and may extend to the end of the line.

Example:

```
;This is a comment.
ld          100         ;This is also a valid
                        ;place for a comment
;This is another comment.
st          v1          ;Comment
```

6.1.4 Symbolic names

Symbolic names can be used as a **text substitute** for operators, operands and labels.
This allows you to design your programs for improved readability and ease of modification. For example, you may substitute the labels L0 to L100 by names which can be more easily remembered. Operators can be substituted by new and more common names.

Rules for symbolic names

– A symbolic name must not contain any blanks.
– A symbolic name has a maximum length of 30 characters.
– You can define up to a maximum of 250 names.

Symbolic names are created with the “Symbolic Names” editor.

NOTE
The names are valid for both the PLC and SEQUENCE program components.

Examples of symbolic names:

```
C_Interface1        c1
X_Axis1             x1
T_Text              70
L_Begin             L1
```

Programming examples:

```
LABEL      L_Begin

snd_str    C_Interface1        T_Text

posf       X_Axis1             1000

jmp        L_Begin
```

6.1.5 Send texts for operator dialogs

Send texts are text lines which can be output on a VT52 terminal, e.g. FT2000, from the SEQUENCE program via the serial interface.
Send texts are created with the text editor.

The text editor can be used for defining up to 97 text lines. Each text line is assigned a number and can be up to 59 characters long.
Send texts can be output via the serial interface <c> using the “snd_str <c> <k>” command. The number of the send text is defined by the <k> constant.

Examples of send texts:

```
21: Please enter a position value
22: Incorrect value entered
```

Programming example 1:

```
snd_str    c1 21  ;Text line 21 is output via
                  ;the serial interface c1.
:
```

Programming example 2:

```
ld         22     ;Text line 22 is output via
st         v1     ;the serial
snd_str c1 v1     ;interface c1.
```

NOTE
Chapter 6.3.5 contains a detailed description of the commands and control codes for programming the serial interface.

6.2 Basic function programming

The following table contains a summary of the commands for basic function programming.

Functional group	Commands Operator <avail. operands>		Function
Loading and storing*	ld	<fknvwiqmxtp>	Loading or reading into the CR
	ldn	<iqm>	Loading or reading the negated value into the CR
	st	<fnvwqmxtp>	Storing
	stn	<qm>	Storing the negated value
Setting and resetting*	r	<qm>	Setting output or flag to 0
	s	<qm>	Setting output or flag to 1
Timer (only in PLC program component)	stimer	<t>	Start timer (only available for PLC program component)
Logical operation commands*	and	<iqm>	AND operation
	andn	<iqm>	Negated AND operation (NAND)
	or	<iqm>	OR operation
	orn	<iqm>	Negated OR operation (NOR)
Arithmetic commands*	add	<fknvw>	Adding
	div	<fknvw>	Dividing
	mul	<fknvw>	Multiplying
	sub	<fknvw>	Subtracting
Relational commands*	eq	<fknvwiqm>	Equal to
	gt	<fknvw>	Greater than
	lt	<fknvw>	Less than
Program jumps and labels*	jmp	<lk>	Unconditional jump
	jmpc	<lk>	Conditional jump, CR = 1 (TRUE)
	jmpn	<lk>	Conditional jump, CR = 0 (FALSE)
	label	<L>	Label for program jump operation
	end		Program end and jump to program beginning
Subprogram calls (in SEQUENCE program component only)	cal	<L>	Unconditional subprogram call
	calc	<L>	Conditional subprogram call, CR = 1 (TRUE)
	caln	<L>	Conditional subprogram call, CR = 0 (FALSE)
	ret		Return from subprogram

* These functions are available for programming in both the PLC and SEQUENCE program components.

6.2.1 Loading and storing

The load commands “ld” and “ldn” can be used for reading a value and storing it in the CR.
The commands “st” and “stn” can be used for transferring the contents of the CR to a different storage element.

Commands for loading and storing (PLC/SEQUENCE)		
ld	<fknvwiqmxtp>	Loading or reading into the CR
ldn	<iqm>	Loading or reading the negated value into the CR
st	<fnvwqmxtp>	Storing
stn	<qm>	Storing the negated value

Programming examples:

```
ld      100     ;Load the value 100 into the CR
st      v10     ;and store it in the variable v10.

ld      i10     ;Read input i10 into the CR
st      q1      ;and output it to output q1.

ld      x1      ;Read actual position of axis x1.

ld      0       ;Dimension setting on axis x1,
st      x1      ;the current position of the axis
                ;is defined as the zero point.

ld      p2      ;Read encoder position from
                ;p2 encoder.

ld      100     ;Load timer t1.
st      t1

ld      t1      ;Read time from timer t1.
```

6.2.2 Setting and resetting

The commands "**s**" and "**r**" can be used for setting an output (q) or flag (m) to 1 or resetting it to 0.

NOTE
These two commands are only executed if the CR contains a value which is not equal to 0 (TRUE).

Commands for setting and resetting (PLC/SEQUENCE)		
r	<qm>	Setting output or flag to 0
s	<qm>	Setting output or flag to 1

Programming examples:

```
ld      i10     ;Load input i10 into CR.
r       q5      ;Output 5 is only reset if the
                ;input i10 = 1.

ld      i11     ;Load input i11 into CR.
s       m500    ;Flag 500 is only set to 1 if the
                ;input i11 = 1.
```

6.2.3 Logical operations

Logical operations can be performed with the "**and**" and "**or**" operations. The result of such an operation is stored in the CR.

Logical operation commands		
and	<iqm>	AND operation
andn	<iqm>	Negated AND operation (NAND)
or	<iqm>	OR operation
orn	<iqm>	Negated OR operation (NOR)

Programming examples:

```
ld      m200    ;Jump to label L10 if
and     m201    ;flag m200 and flag m201
andn    m202    ;are set and flag m202
jmpc    L10     ;is not set.
```

6.2.4 Relational operations

The commands “**eq**”, “**gt**” and “**lt**” can be used for relational operations. The relational operation is performed with the CR and the operand of the relational operation command. The result of such an operation is either 0 (FALSE) or 1 (TRUE) and stored in the CR.

NOTE
After a relational operation, conditional jumps or subprogram calls which depend on the CR can be executed.

Relational commands (PLC/SEQUENCE)		
eq	<fknvwiqm>	Equal to
gt	<fknvw>	Greater than
lt	<fknvw>	Less than

Programming example 1:

```
ld      i10     ;Jump to label L10 if
eq      i11     ;the inputs i10 and i11 have
jmpc    L10     ;the same status.
```

Programming example 2:

```
ld      v10     ;Call a subprogram if
lt      100     ;variable v10 is less than 100.
calc    L11
```

6.2.5 Arithmetic calculations

The four basic arithmetic operations are available for arithmetic calculations.
The result is produced from the operation of the CR with the operand and is stored in the CR.

NOTE
You can only use integer numbers (constants and variables) for calculations. Calculations with fixed-point numbers are not possible.

Arithmetic commands (PLC/SEQUENCE)		
add	<fknvw>	Adding
div	<fknvw>	Dividing
mul	<fknvw>	Multiplying
sub	<fknvw>	Subtracting

Programming example 1:

```
;(356 + 744) * 3
ld      356
add     744
mul     3
```

Programming example 2:

```
;356 + 744 * 3
ld      744
mul     3
add     356
```

6.2.6 Jump instructions

The commands "**jmp**", "**jmpc**" and "**jmpn**" are used for programming branches in programs.
The unconditional jump instruction "jmp" is always executed.
The execution of the conditional jump instructions "jmpc" or "jmpn" depends on the CR.

There are two ways of specifying a jump destination:

– Absolute, by way of a label e.g.: jmp L10
– Relative, by a certain number of lines e.g.: jmp -7

Labels are set with the "label" command.

NOTE
To execute a relative jump backwards, specify a negative number.

Jumps and labels (PLC/SEQUENCE)		
jmp	<lk>	Unconditional jump
jmpc	<lk>	Conditional jump, CR = 1 (TRUE)
jmpn	<lk>	Conditional jump, CR = 0 (FALSE)
label	<L>	Label
end		Program end and jump to program beginning

Programming example 1:

```
;Absolute conditional jump instruction
ld       i10      ;Jump to label L30 if
jmpc     L30      ;input i10 is set (=1).
...
LABEL    L30      ;Label L30
```

Programming example 2:

```
;Loop programming
ld       1        ;Loop counter v15
st       v15      ;initialization.

LABEL    20       ;Label for the loop.
...
ld       v15      ;Increment the loop counter.
add      1
st       v15
eq       100      ;Check for loop end.
jmpn     L20      ;100 loop passes.
```

Programming example 3:

```
;Relative conditional jump instruction
ld       m1001    ;Wait until axis 1 has stopped
jmpc     -1       ;Jump back by one line
```

6.2.7 Subprograms

The SEQUENCE program can be structured by the use of subprograms. The commands "**cal**", "**calc**", "**caln**" can be used for programming subprogram jump operations.
The "ret" command is used for ending a subprogram and returning to the calling program line.
The unconditional subprogram call "cal" is always executed.
Execution of the conditional subprogram calls "calc" or "caln" depends on the CR.

NOTE
Subprograms can only be used in the SEQUENCE program in the same program text as the SEQUENCE program. There are no separate subprogram files.

Subprogram calls (SEQUENCE)		
cal	<L>	Unconditional subprogram call
calc	<L>	Conditional subprogram call, CR = 1 (TRUE)
calcn	<L>	Conditional subprogram call, CR = 0 (FALSE)
ret		Return from subprogram

Programming example:

```
;Conditional subprogram call
ld       i10      ;Jump to label L30 if
calc     L30      ;input i10 is set (=1).
...
;Subprogram
LABEL    L30      ;Subprogram label
...               ;Subprogram commands
ret               ;Return from the subprogram
```

6.3 Controller function programming

This chapter describes how to program controller-specific functions.

Controller-specific functions are programmed using
– control parameters and
– commands

NOTE
Controller-specific functions can only be programmed in the SEQUENCE program.

Commands for SEQUENCE program component				
Functional group	**Commands Operator <avail. operands>**			**Function**
Subprogram calls	cal	<L>		Unconditional subprogram call
	calc	<L>		Conditional subprogram call, CR = 1 (TRUE)
	caln	<L>		Conditional subprogram call, CR = 0 (FALSE)
	ret			Return from subprogram
Axis operating mode	mode	<x>	<k>	Set the axis operating mode
Positioning in point-to-point mode	vel	<x>	<fknvw>	Set the set speed
	acc	<x>	<fknvw>	Set the acceleration ramp
	move	<x>	<fknvw>	Relative positioning without waiting
	movef	<x>	<fknvw>	Relative positioning, and wait until position reached
	pos	<x>	<fknvw>	Absolute positioning without waiting
	posf	<x>	<fknvw>	Absolute positioning, and wait until position reached
	ref	<x>		Reference movement without waiting
	reff	<x>		Reference movement, and wait until position reached
	stop	<x>		Stop axis x
	stopa*	<x>		Stop all axes
	setsiglist*	<x>	<knvw>	Activate position list
	settrigger*	<x>	<k>	Position trigger (q1)
Electronic gear in position following mode	gearn	<x>	<fknvw>	Set gear ratio denominator
	gearz	<x>	<fknvw>	Set gear ratio numerator
	goff	<x>	<fknvw>	Set position offset
Linear interpolation for multi-axis controllers	linpos*			Start linear interpolation (absolute)
	linposf*			Start linear interpolation (absolute), and wait until position reached
	linmove*			Start linear interpolation (relative)
	linmovef*			Start linear interpolation (relative), and wait until position reached
	setipos*	<x>	<knvw>	Prepare linear interpolation

Commands for SEQUENCE program component				
Functional group	**Commands Operator <avail. operands>**			**Function**
Communication via serial interface	cursor	<c>	<fknvw>	Position the cursor
	rec_char	<c>	<fnvw>	Receive character
	rec_char_n	<c>	<fnvw>	Receive character and check
	rec_dez	<c>	<fnvw>	Receive decimal number
	rec_var	<c>	<fnvw>	Output and edit number
	rec_var_n	<c>	<fnvw>	Receive and store number
	screen	<c>	<fknvw>	Screen control
	snd_char	<c>	<fknvw>	Output character
	snd_dez	<c>	<fnvw>	Output decimal number
	snd_str	<c>	<fknvw>	Output string
	snd_var	<c>	<fnvw>	Send number
Analog output	getanalog*	<a>	<knvw>	Read analog voltage
	setanalog*	<a>	<knvw>	Output analog voltage
Lauer operating panel**	ld_LKey*	<knvw>		Read key status of Lauer operating panel
	ld_LBit*	<knvw>	<knvw>	Read bit from data interface of Lauer operating panel
	ld_LInt*	<knvw>		Read word from data interface of Lauer operating panel
	ld_LDint*	<knvw>		Read double word from data interface of Lauer operating panel
	st_LBit*	<knvw>	<knvw>	Write bit to data interface of Lauer operating panel
	st_LInt*	<knvw>		Write word to data interface of Lauer operating panel
	st_LDint*	<knvw>		Write double word to data interface of Lauer operating panel
Miscellaneous commands	amp	<x>	<k>	Switching the power controller on and off
	brake*	<x>		Activate output for brake
	clrerror	<x>		Reset axis signal error
	getport	<im>	<fknvw>	Read inputs or flags and convert to integer number
	handshake	<i>	<q>	Synchronization with master controller
	restart			Restart SEQUENCE program
	setcurrent	<x>	<fknvw>	Set the motor current
	wait	<fknvw>		Suspend program execution for a specific time (ms)
	wsave			Save position variables
	nop			Dummy command (500 µs to 1000 µs execution time)

* Series 300 only
** This function is available for programming in both the PLC and SEQUENCE program components.

6.3.1 Controller initialization

Each time a program is started, the controller is initialized using the values set as control parameters.

The following controller components can be adjusted for the specific application by the initialization process:

– Indexer (axis control)
– Power controller
– Encoder
– Serial interfaces
– System characteristics

NOTE
The control parameters and their functions are explained in the "Control parameters" chapter and in the following sections.

6.3.2 Movement programming

6.3.2.1 Axis operating modes

There are two axis operating modes for Series 300 and WDP3-014/018 controllers with OED3 installed:

– Point-to-point mode
– Position following mode

The axis operating modes are set using the "mode" command.

Command			
mode	<x>	<fknvw>	Set axis operating modes

NOTE
The characteristics required for a positioning operation (speeds, acceleration, etc.) are also determined by the control parameters.

Programming example:

```
;Set point-to-point mode for axis x1
mode      x1      0
...
;Set position following mode for axis x1
mode      x1      1
```

6.3.2.2 Point-to-point mode

In point-to-point mode, a positioning command is used for moving an axis from point A to point B.

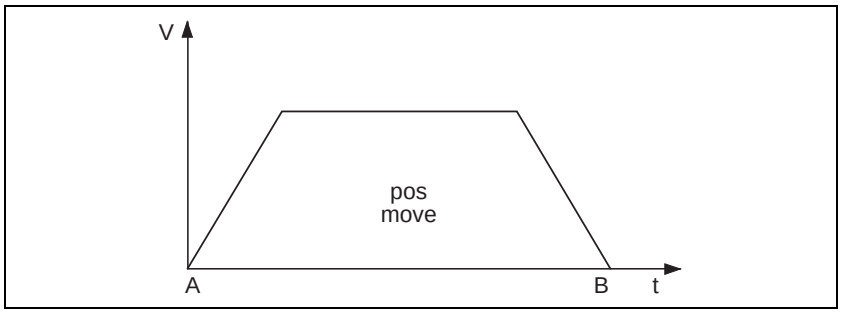


Fig. 6-5 Point-to-point mode

Absolute positioning

The “pos” and “posf” commands are used for positioning an axis in an absolute way, i.e. relative to the reference point (zero point) of the axis.

Commands			
pos	<x>	<fknvw>	Absolute positioning without waiting
posf	<x>	<fknvw>	Absolute positioning, and wait until position reached

Relative (incremental) positioning

The “move” and “movef” commands are used for positioning an axis in an incremental way, i.e. relative to the current position of the axis.

Commands			
move	<x>	<fknvw>	Relative positioning without waiting
movef	<x>	<fknvw>	Relative positioning, and wait until position reached

Positioning with and without waiting

The “move” and “pos” commands are used for starting a positioning operation and continue executing the SEQUENCE program without waiting for the positioning operation to be completed.

NOTE
The end of a “positioning operation without waiting” can be monitored using the flags m1001 to m1004.

The “movef” and “posf” commands are also used for starting a positioning operation, however, the SEQUENCE program will wait until the specified position is reached before it continues to execute.

Programming example:

```
;Absolute positioning without waiting
pos       x1      1000

;Absolute positioning with waiting
posf      x1      1000

;Relative positioning without waiting
move      x1      1000

;Relative positioning with waiting
movef     x1      1000
```

Stopping the axis

You can use the “stop” and “stopa” commands for stopping any or all axes under program control. The axes are braked using the currently set acceleration curve; see the following pages.

Commands		
stop	<x>	Stop axis x
stopa*		Stop all axes

* Series 300 only (multi-axis system)

Monitoring the movement status of an axis

The flags m1001 to m1004 can be used for monitoring the movement status of an axis during positioning or reference movement.

Flag	**Function**
m1001	Status of axis x1: 0 = Axis stopped 1 = Axis moving
m1002	Status of axis x2: 0 = Axis stopped 1 = Axis moving
m1003	Status of axis x3: 0 = Axis stopped 1 = Axis moving
m1004	Status of axis x4: 0 = Axis stopped 1 = Axis moving

Programming example 1:

```
;Absolute positioning, and wait for end of
;positioning
pos      x1     1000   ;Start positioning
...
ld       m1001         ;Wait until axis x1 has
jmpc     -1            ;stopped for any additional
                       ;positioning operations.
```

Programming example 2:

```
;Stop axis x1, and wait until axis has stopped
pos      x1     10000  ;Start positioning.
...
stop     x1            ;Stop axis x1.
ld       m1001         ;Wait until axis x1
jmpc     -1            ;has actually stopped.
                       ;m1001 = 1 : Axis moves
                       ;m1001 = 0 : Axis has
                       ;stopped
```

Monitoring limit switches

The control parameter “Active limit switches xn” can be used for defining the limit switches to be monitored during a positioning operation.
If any of the monitored limit switches is actuated, the controller reports an error.

Parameter	Meaning
Active limit switches x1 to x4	When a limit switch is reached, the drive is brought to a standstill using the set ramp. The controller is then in error status. 0 = No limit switch monitoring 1 = Negative limit switch monitoring 2 = Positive limit switch monitoring 3 = Both limit switches are monitored

Programming example:

The positive and the negative limit switches of the axes x1 and x2 are to be monitored.

Control parameters:

```
"Active limit switches x1" = 3
"Active limit switches x2" = 3
```

Position values in user-defined units

All position values are given in user-defined units. User-defined units allow you to specify positions in common units of measurement (mm, cm, inch, etc.).
Conversion of user-defined units to drive units (motor steps) is effected using a normalizing factor. The normalizing factor is determined by the control parameters “normalizing factor denominator” and “normalizing factor numerator”.

Normalizing factors can be defined individually for each axis.

$$\text{Normalizing factor} = \frac{\text{Normalizing factor numerator}}{\text{Normalizing factor denominator}}$$

Drive units = User-defined units x Normalizing factor

NOTE
The factory default for the normalizing factor is 1, i.e. “normalizing factor denominator” = “normalizing factor numerator”. The basic setting for position values is therefore:

User-defined units = Drive units

Programming example:

A motor is to move a transport slide by means of a spindle drive. The positions for slide positioning are to be specified in centimeters (cm). One motor revolution moves the slide by 10 cm, i.e. 10 cm are equivalent to 1000 steps. Thus, one centimeter is equivalent to 100 steps.
The normalizing factor must be set to 100.

Control parameters:

```
"Normalizing factor numerator x1"   = 100
"Normalizing factor denominator x1" = 1
```

Program:

```
;Move axis x1 by 12 cm (1200 steps)
movef    x1     12
```

Speeds

An axis moves at the speed which is preset with the “Standard speed” control parameter.
The “vel” command can be used for programming the set speed of an axis. The set speed may be changed before or during a positioning operation.

Speeds are specified in hertz (Hz).

1 Hz = 1 step/s.

The start/stop speed and the maximum system speed of an axis are set with control parameters.

Command			
vel	<x>	<fknvw>	Set the set speed

Programming example:

The axis x1 is to start and stop at a start/stop speed of 500 Hz. Normal positioning movements are to be performed at 6000 Hz. The maximum system speed of 17000 Hz must not be exceeded.

Control parameters:

```
"Start/stop speed x1" = 500
"System speed x1"     = 17000
```

Program:

```
;Setting a set speed of 6000 Hz on axis x1
vel      x1     6000
movef    x1     2000

;or
ld       6000
st       v10
vel      x1     v10
movef    x1     2000
```

Acceleration (Hz/ms)

Various acceleration ramps can be defined for each axis for movement sequences which are optimized with respect to the timing or the application.
The same acceleration ramps are used for braking and accelerating.
An axis accelerates and decelerates at the ramp shape set with the "Ramp" parameter.

There are 4 ramp shapes:
– Linear ramp (default)
– Exponential ramp
– Sine square ramp
– Optimized stepping motor ramp (time-optimized acceleration or deceleration)

The following control parameters are used for calculating the current acceleration curve:
– "Ramp", the ramp shape
– "System speed", the maximum system speed
– "Standard acceleration", the maximum acceleration

These parameters can be set separately for each axis.

See also chapter 6.1.2, "Control parameter setting".

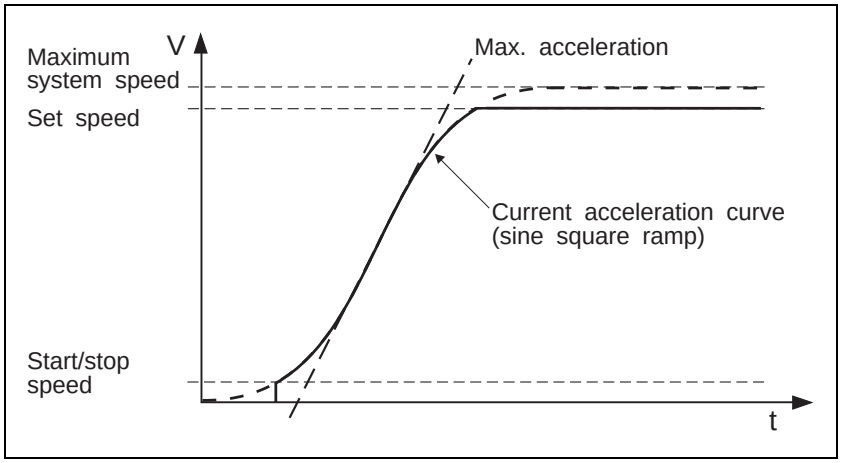


Fig. 6-6 Acceleration curve

The current acceleration curve is only calculated up to the maximum system speed.
The actual movement curve is in the range between start/stop speed and set speed.

NOTE
In the case of non-linear ramps, note that the calculated curve is only utilized to an optimum degree if:
– *the set speed corresponds to the maximum system speed,*
– *the acceleration curve is designed in such a way that the set speed is actually reached during a positioning operation.*

Modifying the acceleration curve

The “acc” command can be used for specifying the maximum acceleration of the ramp.

NOTE
This command is only valid when the axis is at a standstill.

The maximum acceleration is always specified in Hz/ms.

1 Hz/ms = 1000 steps/s^2 = 1 revolution/s^2

Programming example:

A sine square ramp with a maximum acceleration of 200 Hz/ms is to be programmed as the acceleration ramp for axis x1. Positioning operations are executed at a maximum set speed of 11000 Hz.

Control parameters:

```
"Ramp x1"          = 2
"System speed x1" = 11000
"Standard speed"   = 11000
```

Program:

```
;Calculating the acceleration or deceleration ramp
;for axis x1
acc      x1     200    ;Maximum acceleration
                       ;200 Hz/ms.


;or
ld       200
st       v10
acc      x1     v10
```

Reference movement

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

NOTE
Reference movements are only possible in point-to-point mode.

In a reference movement, a reference point is approached which is to be the zero point for all subsequent positioning operations.
The parameter “Type of reference movement” can be used for defining a reference movement to be executed towards the

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

The principles of the different reference movements are illustrated in the figures 6-7 and 6-8. The “Clearing distance limit switch” parameter can be used for programming a safety distance to the limit or reference switch.
The control parameter “Clearing speed limit switch” defines the speed at which the movement away from the limit switch is performed.

The “Max. allowed dist. limit switch” parameter specifies the maximum distance after which the axis must have left the limit switch (reference switch) again. Otherwise the reference movement is cancelled.

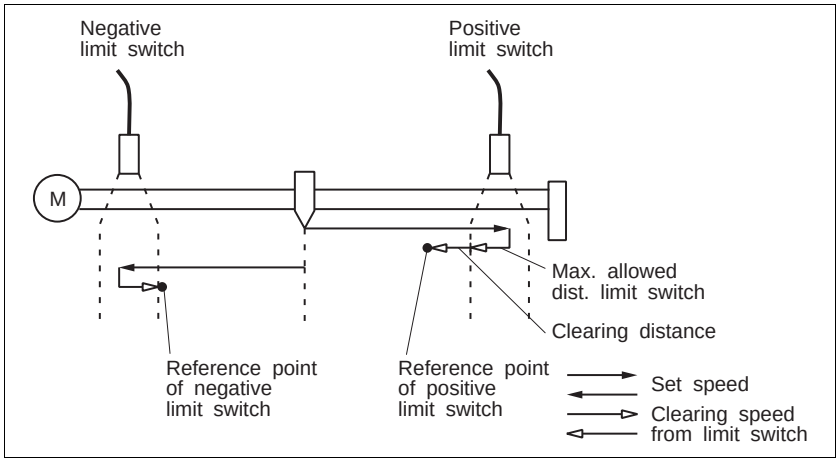


Fig. 6-7 Reference movement towards limit switch

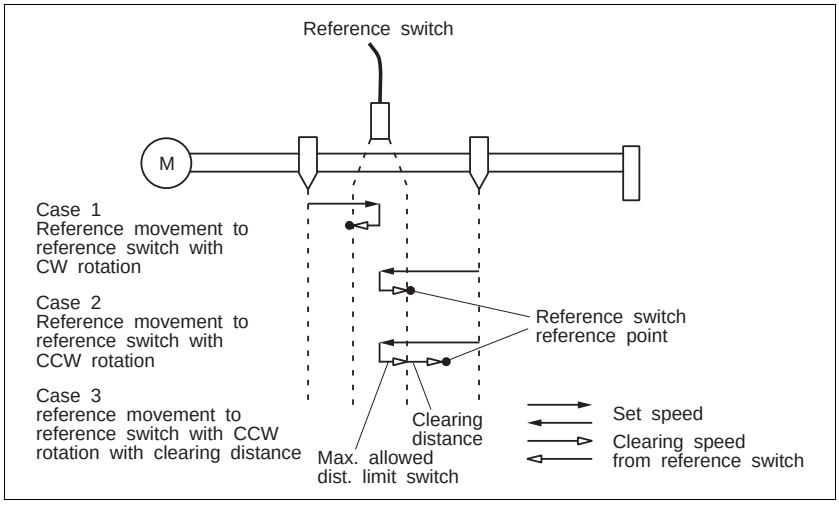


Fig. 6-8 Reference movement towards reference switch

Commands			
ref	<x>	<fknvw>	Reference movement without waiting
reff	<x>	<fknvw>	Reference movement, and wait until position reached

NOTE
When moving towards the reference switch, the acceleration curve and set speed which are set by control parameters or by the program are used.

Control parameters:

– "Type of reference movement"
– "Clearing distance limit switch"
– "Clearing speed limit switch"
– "Standard speed"
– "Max. allowed dist. limit switch"

See also chapter 6.1.2, "Control parameter setting".

Programming example:

A reference movement towards the positive limit switch is to be executed. The axis is to approach the limit switch at a speed of 2000 Hz. The axis is to clear from the limit switch at 100 Hz. The clearing distance is 2 user-defined units.

Control parameters:

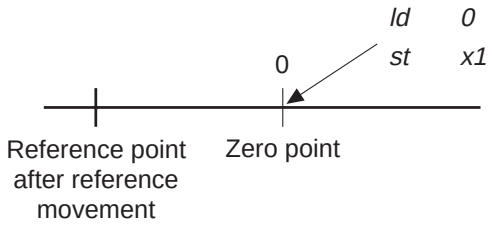
```
"Type of reference movement x1" = 1
"Clearing distance limit switch" = 2
"Clearing speed limit switch" = 100
"Standard speed" = 2000
"Max. allowed dist. limit switch" = 10000
```

Program:

```
reff      x1      ;Reference movement with axis x1
```

Setting dimensions

Usually, the reference point (zero point) for absolute positioning operations is the reference point determined by a reference movement.
However, also any other position in the system can be defined as the reference point. All subsequent absolute positioning operations refer to this point.
The process of defining a reference point is called “setting dimensions”.



Programming example:

```
reff      x1       ;Reference movement
movef     x1 2000  ;Positioning
ld        0        ;Setting dimensions
st        x1       ;Current position of axis x1 = 0
```

Reading the current position

The current position of an axis can be determined at any time in the SEQUENCE program.

Programming example:

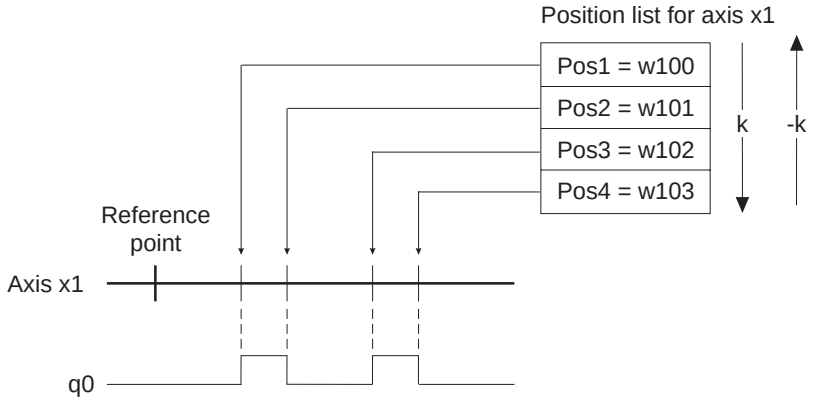
The current position of axis x1 is to be read and stored in the variable v10.

Program:

```
ld        x1       ;Read the current position of
                   ;axis x1 and
st        v10      ;store it in the variable v10.
```

Position-dependent signal output toggling (Series 300 only)

The “setsiglist” command can be used on Series 300 controllers for activating a position list for setting or resetting (toggling) the signal output q0.
Whenever the axis passes a position from this list, the signal level on the output q0 changes. If the output is active, it is deactivated, and vice versa.
At the first position value in the list, the output is set to 1, regardless of whether the output was already set to 1 or not.
Only one position list can be active at a time, i.e. you cannot monitor position lists for several axes simultaneously.
When a position in the list has been passed, this position is no longer monitored.



Command			
setsiglist*	<x>	<knvw>	Activate a position list for toggling the signal output q0

* Series 300 only

The position list must be stored in the position variables of the corresponding axis. The first position of the position list for axis x1 must be stored in the variable w100. For the other axes, w200 must be used for x2, w300 for x3 and w400 for x4.

The <k> operand in the “setsiglist” command specifies the number of positions in the list. If <k> has a negative value, position list monitoring is effected in reverse order.

NOTE
You can use the ProOED3 editor “Position Variables” for creating position lists.

Programming example:

Four positions of axis x1 are to be monitored. Toggling of the output q1 is to be effected at the positions 2000, 3000, 5000 and 6000.

Position list:

```
w100    2000
w101    3000
w102    5000
w103    6000
```

Program:

```
;Monitoring the position list of axis x1
;is activated. The position list contains
;4 entries.
...
reff        x1          ;Setting the zero point.
setsiglist  x1    4     ;Position list active.
posf        x1    10000 ;Start positioning.
...
;To monitor the position list in reverse order,
;the number of positions in the list
;must be specified as a negative number.
...
setsiglist  x1    -4
posf        x1    0
...
```

Activating a position trigger signal (Series 300 only)

On Series 300 controllers, a trigger input signal (trig 1 to trig 4) is available for each axis. Trigger inputs are used for applications which require particularly short response times.
The “settrigger” command activates monitoring a trigger input. This can be used for determining the position of an axis at the point of time when a trigger event occurs.
The occurrence of a trigger event is registered in the flags m1011, m1012, m1013 and m1014. As long as the trigger signal is active, the positions of the respective axis are stored in the variables v111 to v114. If the trigger signal is inactive, the position of the axis is kept in the respective variable.

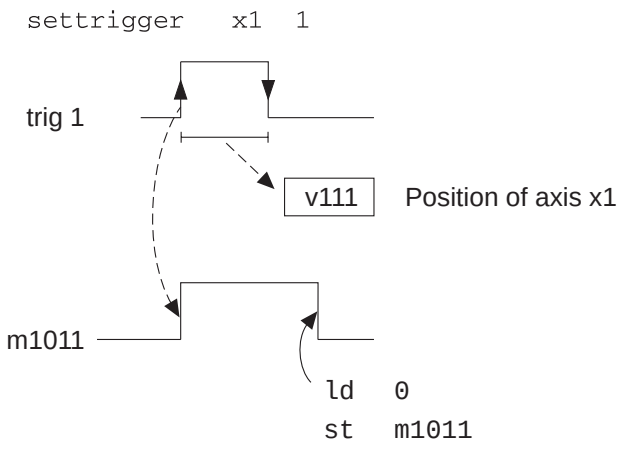
NOTE
The flags m1011 to 1014 must be reset after the response to the trigger signal.

Command			
settrigger*	<x>	<k>	Activate monitoring of the trigger input of an axis

* Series 300 only

Principle:

Monitoring the trigger input of axis 1 (trig 1, rising edge)



The <k> constant selects whether the trigger input of axis <x> is to respond to a rising (k = 1) or falling (k = 0) edge.

Axis	Trigger flag	Trigger position
x1	m1011	v111
x2	m1012	v112
x3	m1013	v113
x4	m1014	v114

Programming example:

The position of the axis is to be determined at a rising edge on the trigger input of axis x1 (useful for trigger signals in the millisecond range).

Program:

```
settrigger  x1     1     ;Start trigger monitoring
                         ;for axis x1 (rising edge)
move        x1     1000  ;Position axis
ld          m1011       ;Check whether trigger
jmpn        -1          ;event occurred
ld          0           ;Reset trigger flag after
st          m1011       ;trigger event
ld          v111        ;Determine the trigger
                         ;position and
st          v10         ;store it in variable v10
```

6.3.2.3 Position following mode (electronic gear)

In position following mode, set positions are predefined by external pulses (via encoder) or by the position values of a variable. The axis moves when pulses are present at the encoder input or when the position value in the variable changes.
The external pulses or the position value in the variable are called reference variables.

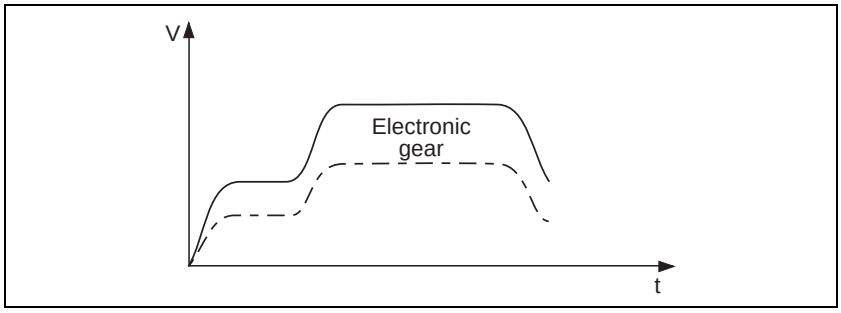


Fig. 6-9 Position following mode

In position following mode, the movement range of the axis is not limited, i.e. the motor can turn in one direction for an unlimited period.
The axis movement can be stopped with the “stop” or “stopa” commands.

Selecting the reference variable

The “mode” command can be used for selecting the type of the reference variable (position presetting type).

Command			
mode	<x>	<k>	Set the axis operating mode

<k> = 1: External pulses
<k> = 2: Variable

External pulses *External pulses supplied from encoder connection*

Usually the motor follows precisely the pulses supplied to the encoder input. The frequency of the external pulse signal determines the acceleration and the speed of the axis.

NOTE
If the frequency of the reference variable is greater than the maximum system speed, the motor moves at system speed.

If the acceleration of the reference variable is greater than the set acceleration, the motor accelerates at the currently set maximum acceleration.

If no pulses are present at the encoder input, this is equivalent to axis standstill.
The characteristics of the encoder, the resolution (in encoder marks) and the evaluation factor (single, double, quadruple evaluation) must be set with control parameters.

The following applies to motor movements:

"Drive units" = Encoder pulses (motor steps) x $\frac{\text{Evaluation factor}}{\text{Encoder resolution}}$

If the reference variable is subject to excessive variation, the acceleration curve set in point-to-point mode is used for the movements (see chapter 6.3.2.2). A temporary storage space is provided to ensure that no pulses are lost.

NOTE
If the drive is stopped and a different axis operating mode selected, the supplied pulses (positions) are not collected.

Variable *Position presetting by a variable*

The value of a variable can be used as the setpoint for an axis. When the variable value changes, the axis moves at the set acceleration curve and at the set speed (see chapter 6.3.2.2).

For this purpose, a special variable is reserved for each axis:

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

Programming example:

Axis x1 is to operate in position following mode and process external pulses from an encoder DG2 as the reference variable.

Control parameters (see page 6-53):

```
"Encoder setting"          -1 (Series 300 only)
"Encoder evaluation DG2"   5
"Gear interface signals"   0
```

Program:

```
;Position following mode via encoder
mode     x1     1
...
label    L1            ;Supply external pulse via
jmp      -1            ;encoder connection 2
```

NOTE
The functions for electronic gear with encoder as well as the parameter settings for position following mode are described on the following pages.

Electronic gear

In position following mode, an electronic gear can be implemented. For this purpose, the “gearz” and “gearn” commands are used for setting a gear ratio which is multiplied with the supplied reference variable (A/B signals, pulse/direction signals or value of a variable).

The following applies:

Motor steps = Reference variable x Gear ratio x Evaluation factor

The “goff” command can be used for superimposing the reference variable with a position offset. An offset is a position which is added to the reference variable.
When changing to position following mode, the offset is set to zero.
Changing the offset with “goff” accelerates or decelerates the axis. When the offset has been processed, the axis continues to run normally.

The following applies (only when changing the offset):

Motor steps = Offset + (Reference variable x Gear ratio)

NOTE
Programming the gear ratio for an electronic gear must be effected before changing over to position following mode.

Programming example:

Axis x1 is to be used for implementing an electronic gear via encoder connection 2. The gear ratio is to be set to 7.5.

Control parameters (see page 6-53):

```
"Encoder setting"          -1 (Series 300 only)
"Encoder evaluation DG2"    5
"Gear interface signals"    0
```

Program:

```
gearn    x1     10      ;Gear ratio denominator = 10
gearz    x1     75      ;Gear ratio numerator = 75
mode     x1     1       ;Position following mode
                        ;via encoder
...
label    L1             ;Supply external pulses via
jmp      -1             ;encoder connection 2
```

Setting the encoder

The following control parameters are used for programming the encoder:
– Gear interface signals
– Encoder setting (only required for Series 300)
– Encoder evaluation

Parameter	Default value	Range of values	Description
Gear interface signals	1	0, 1	Set type of input signal at encoder input 0 = Pulse/direction signal 1 = A/B signal
Encoder setting*	0	-1 to 2	Usage of the encoder connections -1 = Connection 1 (p1) not used Connection 2 (p2) for position following mode 0 = Connection 1 (p1) for position following mode Connection 2 (p2) not used 1 = Connection 1 (p1) for rotation monitoring Connection 2 (p2) for position following mode 2 = Connection 1 (p1) for position following mode Connection 2 (p2) for rotation monitoring Encoder connection 1 (p1) is not available on WDP3-314/ WDP3-318 controllers. The following settings are possible: -1 = Connection 2 (p2) for position following mode 0 = Connection 2 (p2) not used 2 = Connection 2 (p2) for rotation monitoring
Encoder evaluation DG1 or DG2 (Encoder 1 is only available on WP-311 or WDP5-318.)	3	0 to 5	Encoder resolution (increments/revolution) and encoder evaluation (single, double, quadruple) 0 = ’500 encoder, single evaluation 1 = ’500 encoder, double evaluation 2 = ’500 encoder, quadruple evaluation 3 = ’1000 encoder, single evaluation 4 = ’1000 encoder, double evaluation 5 = ’1000 encoder, quadruple evaluation

* Series 300 only

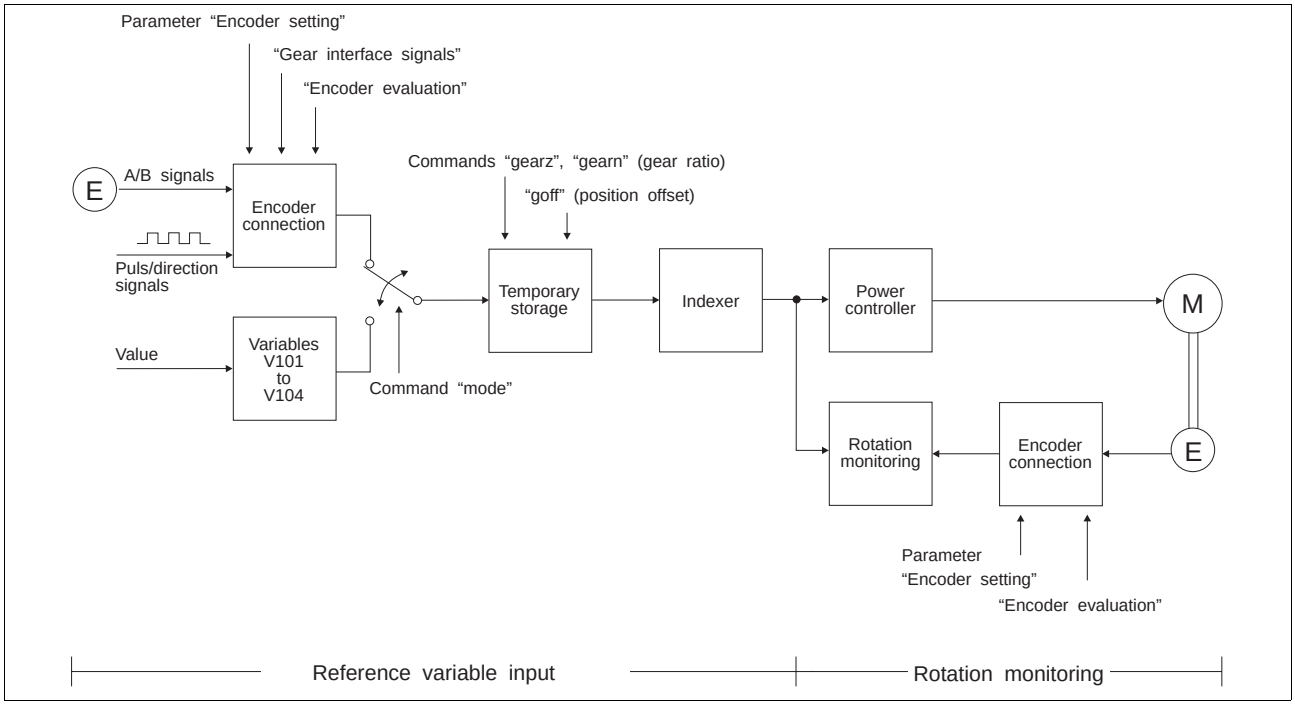


Fig. 6-10 Encoder programming

Programming example:

Axis x1 is to be used for implementing an electronic gear via encoder p2 (DG2). The gear ratio is to be set to 10.
The encoder has 1000 marks per revolution, and quadruple evaluation is performed. Pulse/direction signals are used on the encoder.

Control parameters:

```
"Encoder setting"        -1 (Series 300 only)
"Encoder evaluation DG2"  5
"Gear interface signals"  0
```

Program:

```
gearn    x1     1      ;Gear ratio denominator = 1
gearz    x1     10     ;Gear ratio numerator = 10
mode     x1     1      ;Position following mode
                       ;via encoder
```

Reading the encoder position

The current encoder position can be read and processed by the program as follows:

```
ld       p2             ;Read current encoder
                        ;position
st       v10            ;and store it in the
                        ;variable V10
```

6.3.2.4 Rotation monitoring

Rotation monitoring for controllers with internal power controller

Rotation monitoring is used for detecting and avoiding any positional deviations occurring during a motor movement.
Positional deviations can occur when the motor cannot reach the preset position because of an obstacle or a load variation; this results in a “loss of synchronicity”.
With rotation monitoring, the actual position is detected by an encoder and then compared with the setpoint. If the difference between set and actual position (the following error) exceeds the following error limit (19 steps), a contouring error is generated.

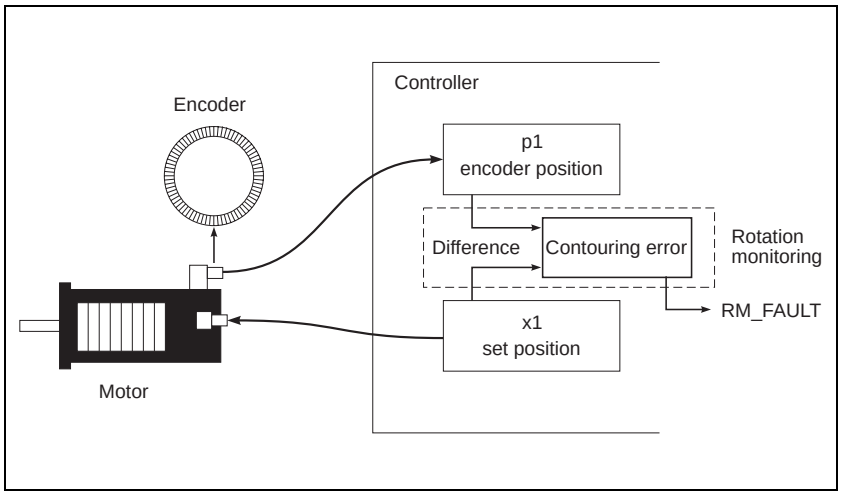


Fig. 6-11 Rotation monitoring principle

When a contouring error occurs, the following happens:

1. The motor decelerates at the set ramp.
2. The outputs are deenergized.
3. The error message “12” appears in the front panel status display. The error menue display depends on the setting of the control parameter “Error handling”.

Parameter	Default value	Range of values	Description
Encoder setting*	0	-1 – 2	Usage of encoder connections -1 = Connection 1 (p1) not used Connection 2 (p2) for position following mode 0 = Connection 1 (p1) for position following mode Connection 2 (p2) not used 1 = Connection 1 (p1) for rotation monitoring Connection 2 (p2) for position following mode 2 = Connection 1 (p1) for position following mode Connection 2 (p2) for rotation monitoring Encoder connection 1 (p1) is not available on WDP3-314/WDP3-318 controllers. The following settings are possible: -1 = Connection 2 (p2) for position following mode 0 = Connection 2 (p2) not used 2 = Connection 2 (p2) for rotation monitoring
Encoder evaluation DG1 or DG2 (Encoder 1 is only available for WP-311 and WDP5-318.)	3	0 – 5	0, 1 or 2 for encoders with a resolution of 500 3, 4 or 5 for encoders with a resolution of 1000
Rotation monitoring x1 to x4	0	0, 1	0 = Deactivate rotation monitoring 1 = Activate rotation monitoring

* Series 300 only

NOTE
When using the rotation monitoring function, the A/B signals of an encoder are evaluated. If an encoder with a resolution of 500 marks is used, double evaluation is always applied, regardless of the value of the "Encoder evaluation" parameter.
If an encoder with a resolution of 1000 marks is used, single evaluation is always applied, regardless of the value of the "Encoder evaluation" parameter.
The "Gear interface signals" control parameter does not have an effect with the rotation monitoring function.

Troubleshooting after a contouring error

1. Eliminate the fault, e.g. mechanical blocking, dirt.
2. Restart the program and begin with a reference movement.

Programming example:

The encoder p2 (DG2) is to be used for rotation monitoring on axis x1. It is a 1000-mark encoder. A/B signals are supplied to the encoder input.

Control parameters:

```
"Encoder setting"         2
"Encoder evaluation DG2" 3 (Series 300 only)
"Rotation monitor x1"     1
```

Rotation monitoring for controllers with external power controller

If rotation monitoring is effected by an external BERGER LAHR power controller (e.g. WD3-008), a rotation monitoring error is reported by the RM_FAULT signal from of the controller (e.g. WPM-311).
The control parameter “Rotation monitor” can be used for enabling or disabling RM_FAULT signal evaluation on the controller.

Parameter	Default value	Range of values	Description
Rotation monitor x1 to x4	0	0 – 1	Toggle signal evaluation for “RM_FAULT” 0 = Disable evaluation of error signal “RM_FAULT” 1 = Enable evaluation of error signal “RM_FAULT”

NOTE
When using rotation monitoring with an external power controller, no further settings are possible for the encoder.

6.3.2.5 Controlling a brake (Series 300 only)

The "brake" command can be used on Series 300 controllers for defining an output for controlling a brake. Figure 6-13 shows the relationship between the ENABLE (power controller enable) and READY (power controller ready) signals and the output signal for the brake.

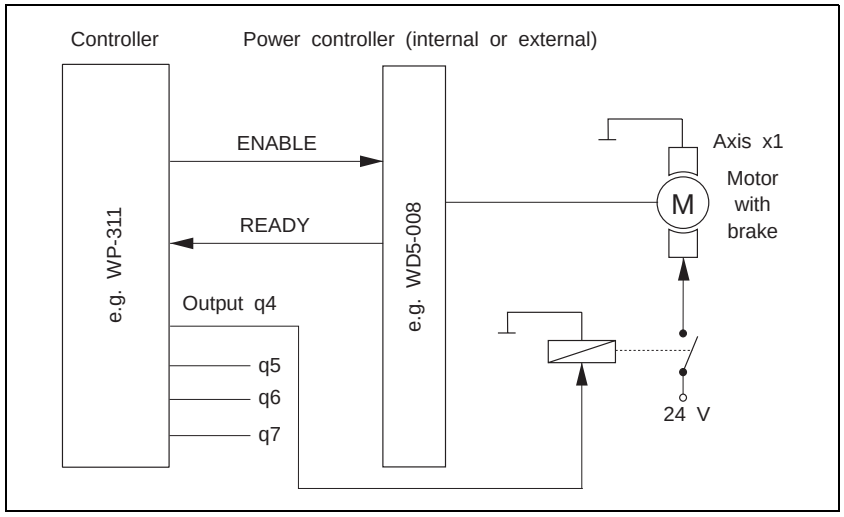


Fig. 6-12 Signals for the brake function

The brake is opened (q = high) when the power controller has been enabled and is ready. The brake is applied (q = low) when the power controller readiness fails (READY = low).

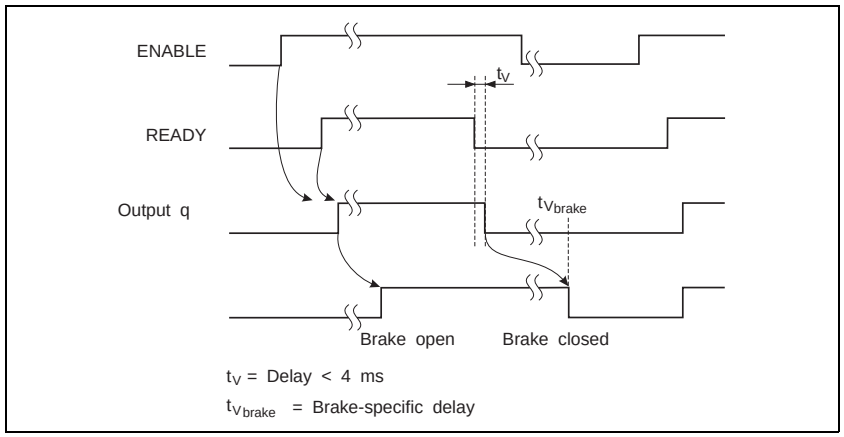


Fig. 6-13 Brake function timing diagram

Command		
brake*	<x>	Activate output for brake

* Series 300 only

NOTE
The "brake" command can only be cancelled by switching off the controller. The brake output remains active as long as the controller is energized.

Programming example:

```
;The output q4 is to be used for controlling the
;motor brake of the axis x1.
...
brake    x1
```

6.3.2.6 Interpolation for multi-axis positioning units (WPM)

On Series 300 multi-axis positioning units (e.g. WPM-311), several axes can be controlled simultaneously. Axis movements can be performed independently or interdependently (interpolated). Linear interpolation is possible with two or three axes.

NOTE
Linear interpolation can be effected with absolute values (position values referring to the zero points of the axes) or with relative values (position values referring to the current axis position), with or without a waiting period (program execution is suspended until interpolation has been completed).

Commands			
linpos*			Start linear interpolation (absolute)
linposf*			Start linear interpolation (absolute) and wait until position reached
linmove*			Start linear interpolation (relative)
linmovef*			Start linear interpolation (relative) and wait until position reached
setipos*	<x>	<knvw>	Prepare linear interpolation

* Series 300 only (multi-axis system)

The “stopa” command can be used for stopping linear interpolation.

The following points must be observed for interpolation:

– The axes involved in interpolation must be set to point-to-point mode (see “mode” command).
– The speeds and accelerations of the axes involved must be set before the interpolation process (see “vel” and “acc” commands).
– The target positions, speeds and accelerations of the axes involved cannot be changed during an interpolation process.
– You cannot perform several interpolation processes at the same time.
– An electronic gear is affected by interpolation (it may be necessary to reinitialize it after the interpolation process).

Principle of linear interpolation

Figure 6-14 illustrates the principle of linear interpolation by way of an example:

Two axes (x1 and x2) are to move from point A to point B with linear interpolation.

The setpoints for the axes are to be input (e.g. with the commands “setipos x1 500” and “setipos x2 200”) and the interpolation started with the “linmove” command.

The linear interpolator calculates the speeds and accelerations required for the interpolation and controls the individual axes. It is ensured that the preset speeds and accelerations of the individual axes are not exceeded.

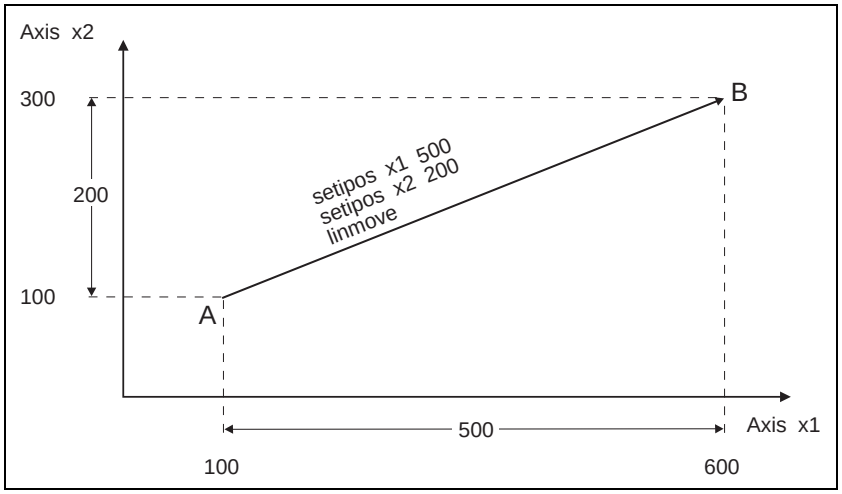


Fig. 6-14 Principle of linear interpolation

Programming example:

Axes x1 and x2 to move from position A (x1 = 100, x2 = 100) to position B (x1 = 600, x2 = 300) with linear interpolation.

Program:

```
setipos     x1    500    ;Relative position for x1
setipos     x2    200    ;Relative position for x2
linmove                  ;Start positioning
```

6.3.2.7 Manual movement via signal inputs

Manual positioning of axes can be controlled with special flags.

Function of the flags

Flag	Function			
m0	Manual movement to the right (m0 = 1)			
m1	Manual movement to the left (m1 = 1)			
m2	Rapid speed (m2 = 1)			
m3	Axis selection	Axis	m4	m3
m4	Axis selection	x1	0	0
		x2	0	1
		x3	1	0
		x4	1	1
m10	Activate manual movement			
m11	Manual movement active? m11 = 0: no m11 = 1: yes			

A manual movement is programmed by addressing the signal inputs with these flags.

In the example, the signal inputs are used as follows:

i1	m0	Manual movement to the right
i2	m1	Manual movement to the left
i5	m2	Select rapid speed
i6	m3	Select axis
i7	m4	Select axis
i8	m10	Activate manual movement

Programming example:

```
;Inputs control manual movement via flags
ld      i1        ;Manual movement to the right
st      m0
ld      i2        ;Manual movement to the left
st      m1
ld      i5        ;Rapid speed
st      m2
ld      i6        ;Axis selection
st      m3
ld      i7        ;Axis selection
st      m4
ld      i8        ;Activate manual movement
st      m10
```

6.3.3 External inputs/outputs with MP 926 (Series 300 only)

Series 300 controllers can control up to 5 external MP 926 input/output cards via an RS 485 HS interface. An MP 926 input/output card features 16 inputs and 16 outputs.

The control parameter “External I/O modules” is used for setting the number of connected input/output cards (up to a maximum of five cards).

The inputs are assigned the flags m32 to m111, the outputs are assigned the flags m112 to m191 (fig. 6-15). The flags can be read and written in the SEQUENCE and PLC program.

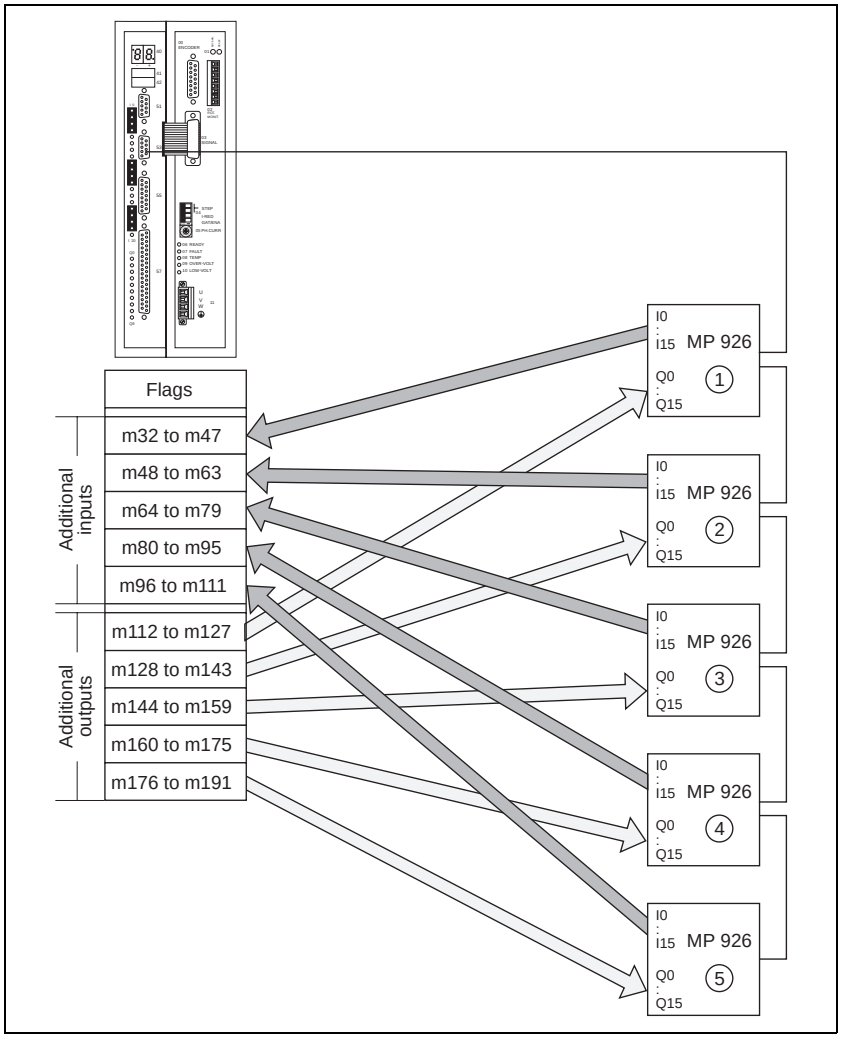


Fig. 6-15 Flag ranges for external inputs/outputs

Programming example:

Input i0 of the MP 926 I/O extension to be read and output to the output q0.

Control parameters:

```
"External I/O modules"  1
```

Program:

```
ld       m32      ;Read input i0 of MP 926.
st       m112     ;Set output q0 of MP 926.
```

6.3.4 Analog signals (Series 300 only)

The “getanalog” command can be used on Series 300 controllers for reading voltage values (millivolts) from an analog module. The “setanalog” command can be used for outputting an output voltage to an analog module.

Commands			
getanalog*	<a>	<knvw>	Read analog voltage
setanalog*	<a>	<knvw>	Output analog voltage

* Series 300 only

Programming example:

Voltage of analog input 1 of analog module a2 to be read and stored in the variable v10.

Program:

```
getanalog   a2    1
st          v10
```

6.3.5 Communication via the serial interface

The communication between the controller and a VT52 operating terminal, e.g. FT2000, via the serial interface can be freely programmed by the user.

NOTE
The communication via the serial interface can be tested using ProOED3; see VT52 simulation or FT2000 simulation.

NOTE
WDP3-014/018 controllers have only one serial interface c1, which is used for programming and communication.

The transmission parameters of the two interfaces are set as follows:

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

A number of commands are available for programming the serial interfaces:

Commands			
cursor	<c>	<fknvw>	Position the cursor
rec_char	<c>	<fnvw>	Receive character
rec_char_n	<c>	<fnvw>	Receive character and check
rec_dez	<c>	<fnvw>	Receive decimal number
rec_var	<c>	<fnvw>	Output and edit number
rec_var_n	<c>	<fnvw>	Receive and store number
screen	<c>	<fknvw>	Screen control
snd_char	<c>	<fknvw>	Output character
snd_dez	<c>	<fnvw>	Output decimal number
snd_str	<c>	<fknvw>	Output string
snd_var	<c>	<fnvw>	Send number

The “snd_str” command can be used for sending complete texts through the interface. The texts must have been created previously with the “Send Texts” editor.

Characters are interpreted as follows:

Character in controller input/output buffer	**Sense of transmission**	**Character via interface (hexadecimal code)**	
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)	*
$N	↔	(16#0A) New Line	*
$R	↔	<CR> (16#0D) Carriage Return	*
$00 - $1F	←	16#00 - 16#1F	*
$00 - $FF	→	16#00 - 16#FF	*

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

NOTE
Control codes must be specified in the “Send Texts” editor with a $ character and a special letter, e.g. $R for Carriage Return.

Individual characters and variables can be transmitted with the commands “snd_char” and “snd_var”.
The “rec_var” command allows you to read in characters conveniently via a VT52 terminal, e.g. FT2000.
Suitable commands for cursor positioning and screen control on a VT52 terminal are “cursor” and “screen” .

NOTE
The commands for communication via the serial interface are described in detail in the “Command list” in the appendix.

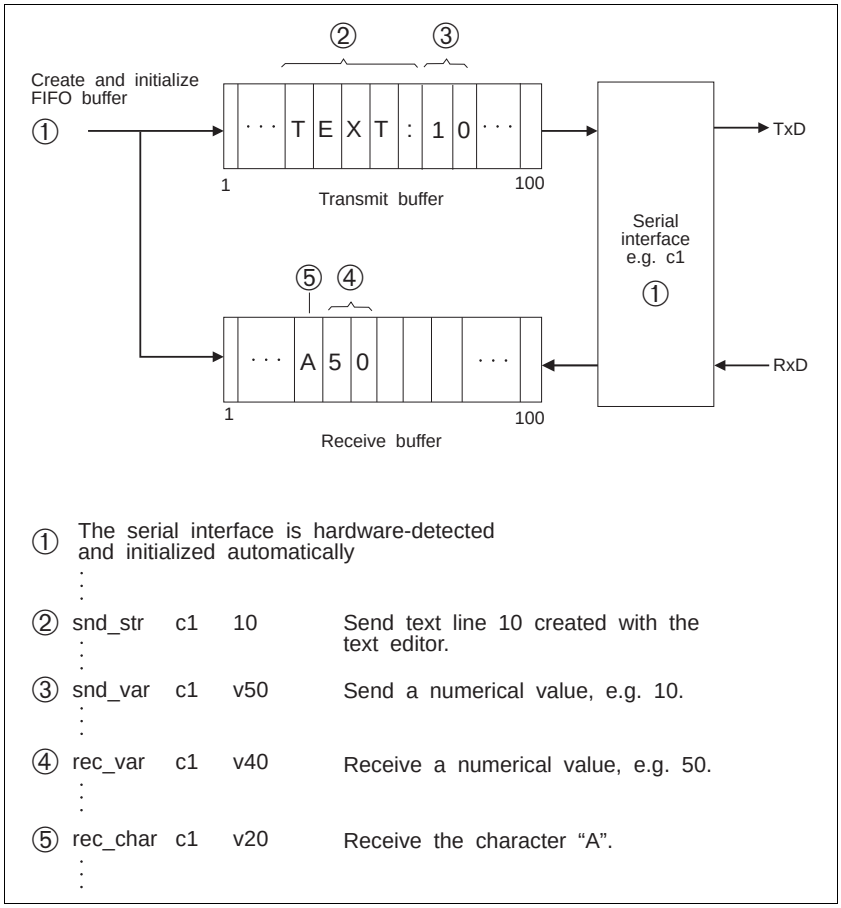


Fig. 6-16 Interface programming

Programming example 1:

Five lines of text to be output line by line on a terminal.
Before outputting each line, a line feed (or “Newline”) and a carriage return are to be performed.
Text lines are entered using the “Send Texts” editor.

“Send Texts” editor:

```
#14$R
$R$NThis is the first of 5 lines.
$R$NThese 5 lines are
$R$Nsent one after the other through
$R$Nthe serial interface.
$R$NThis is the last of 5 lines.
```

NOTE
When using the FT2000 operating terminal, the device polling command #<address><CR> must be sent to the terminal before transmitting or receiving any data.
Example:
The command #14<CR> polls the FT2000 with the address 14.

The device address can be selected when setting up the FT2000; see FT2000 documentation.

Program:

```
snd_str  c1     1      ;Poll FT2000
;Output text line by line on a terminal
ld       2             ;Initialize v10 with the
st       v10           ;first line number.
snd_str  c1     v10    ;Output line.
ld       v10           ;Increment line number.
add      1
st       v10
eq       6             ;Last line ?
jmpn     -5            ;Output next line.
```

Programming example 2:

A number to be read from a terminal into a variable.
The number is to be edited and the range of values (0 to 1000) should be checked.
The prompt “Please enter number:” and the number stored in the variable are to be displayed.
The input operation is not to be considered complete until a correct number has been entered.

“Send Texts” editor:

```
#14$R
$R$NPlease enter number:
$R$NIncorrect input
$R$NInput correct
```

Program:

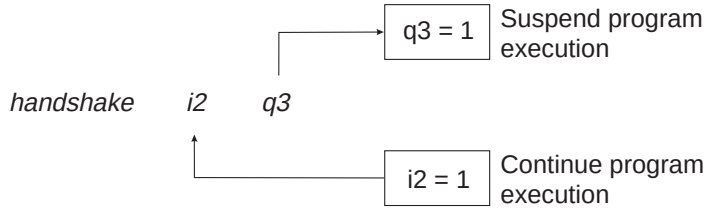
```
snd_str  c1     1      ;Poll FT2000
;Comfortable number input on a terminal
label    L10
snd_str  c1     2      ;"Please enter number:"
rec_var  c1     v10    ;Start number input.
ld       v10           ;Check range of values.
lt       0
jmpc     L20           ;Too small: New input
ld       v10
gt       1000
jmpc     L20           ;Too great: New input
snd_str  c1     4      ;"Input correct:" Continue
jmp      L30
...
label    L20           ;"Incorrect input"
snd_str  c1     3
jmp      L10

label    L30
```

6.3.6 Synchronization with a master controller

The “handshake” command can be used for synchronizing execution of a SEQUENCE program with a master controller.

Command			
handshake	<i>	<q>	Synchronization of program execution with a master controller

The controller indicates on output <q> that it has executed a “handshake” command. Execution of the SEQUENCE program is suspended until the input <i> becomes active (= 1). Program execution is then resumed and the output <q> reset (= 0).



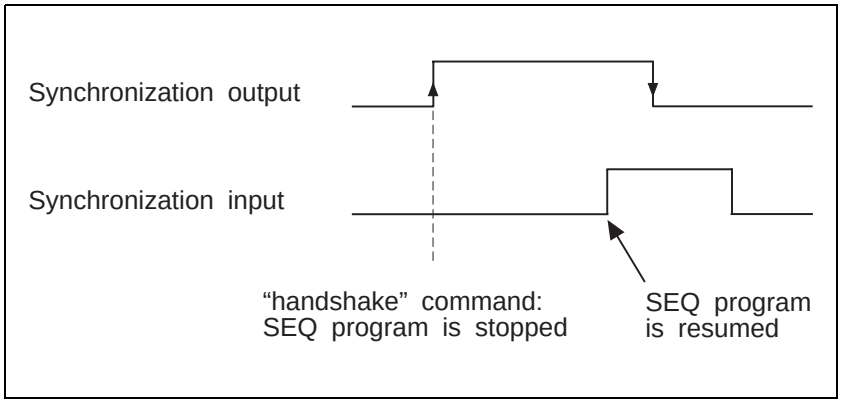


Fig. 6-17 Synchronization

Programming example:

A positioning sequence is to be synchronized with a master PLC.

```
Program:
...
posf      x1   1000
handshake i2   q3
posf      x2   2000
handshake i2   q3
...
```

6.3.7 Lauer operating panel (Series 300 only)

Operating panels of the PCS-topline series made by Lauer can be connected to the Series 300 controllers.

The following operating panels can be used:

- topline micro PCS009
- topline mini PCS090 and PCS095
- topline midi PCS900, PCS920 and PCS950
- topline maxi PCS9000 and PCS9100

NOTE
The operating panels must feature an Interbus-S interface for connection to a BERGER LAHR controller.

Connecting a Lauer operating panel to a BERGER LAHR controller

The Lauer operating panels are connected to a BERGER LAHR controller either directly via the RS 485 HS interface or via the MP926 I/O module.

ATTENTION
If one or more I/O modules are connected to the controller, the Lauer operating panel must be the last device in the configuration (figure 6-18).

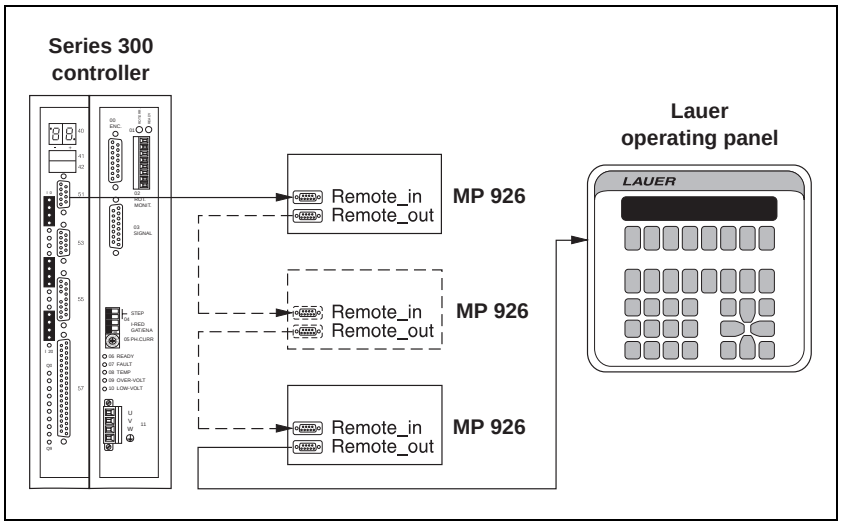


Fig. 6-18 Connecting a Lauer operating panel to a BERGER LAHR controller

The wiring between a Series 300 BERGER LAHR controller and a Lauer operating panel is as follows:

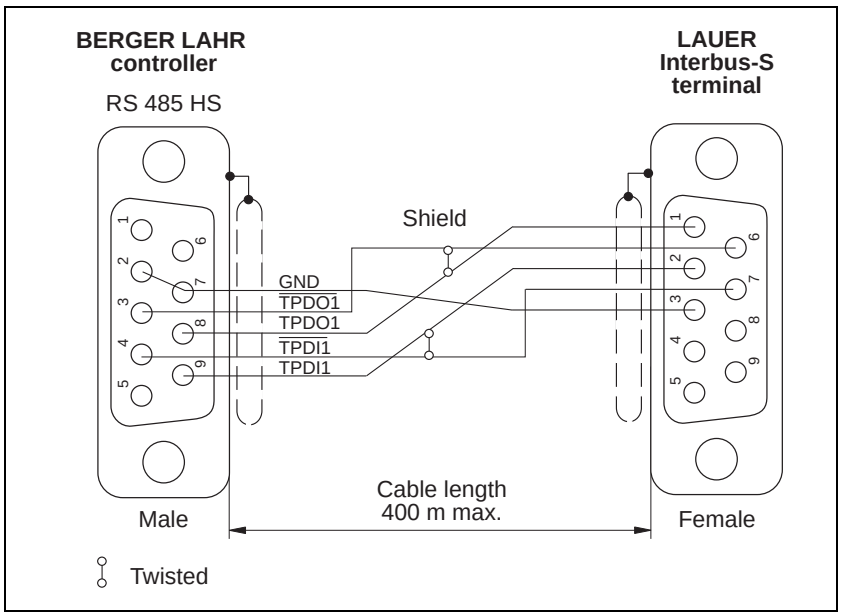


Fig. 6-19 Wiring between controller and Lauer operating panel

Communication between controller and operating panel

Communication between the controller and the operating panel is effected via a data interface. The data interface comprises a range of 256 words (16 bits).
This data range can be read and written with commands.
The structure and the contents of the data interface are determined by the Lauer operating panel. Refer to the Lauer documentation for more details.

Control parameters

The following control parameters must be set correctly in order to enable communication between a Series 300 controller and a Lauer operating panel:

– “Lauer operating panel”
– “External I/O modules”

ATTENTION
The setting of the “External I/O modules” parameter depends on the setting of the “Lauer operating panel” parameter.

Parameter	Default value	Range of values	Description
Lauer operating panel	0	0 – 8	1 = 8 data bytes of type micro 2 = 8 data bytes of type mini 3 = 8 data bytes of type midi 4 = 8 data bytes of type maxi 5 = 16 data bytes of type micro 6 = 16 data bytes of type mini 7 = 16 data bytes of type midi 8 = 16 data bytes of type maxi
External I/O modules	0	0 – 5	If either the value 5, 6, 7 or 8 has been set for "Lauer operating panel", the "External I/O modules" parameter may only be set to 0, 1 or 2.

NOTE
If only the Lauer operating panel is connected to the RS 485 HS interface, the "External I/O modules" parameter must be set to 0.

Commands for communication between controller and operating panel

The data exchange between controller and operating panel is programmed using the following commands:

`ld_LKey`	`<knvw>`		Read the status of a key (keyboard bit) directly from the Lauer operating panel
`ld_LBit`	`<knvw>`	`<knvw>`	Read a bit from the data interface of the Lauer operating panel
`ld_LInt`	`<knvw>`		Read a word from the data interface of the Lauer operating panel
`ld_LDint`	`<knvw>`		Read a double word from the data interface of the Lauer operating panel
`st_LBit`	`<knvw>`	`<knvw>`	Write a bit to the data interface of the Lauer operating panel
`st_LInt`	`<knvw>`		Write a word to the data interface of the Lauer operating panel
`st_LDint`	`<knvw>`		Write a double word to the data interface of the Lauer operating panel

6.4 Program testing

ProOED3 offers several options for debugging and program testing. For a more detailed description of these options, refer to chapter 5.4, “On-line functions”.

I/O test

The I/O test checks that the signal states of the inputs and outputs are set correctly. You can also manipulate outputs manually.

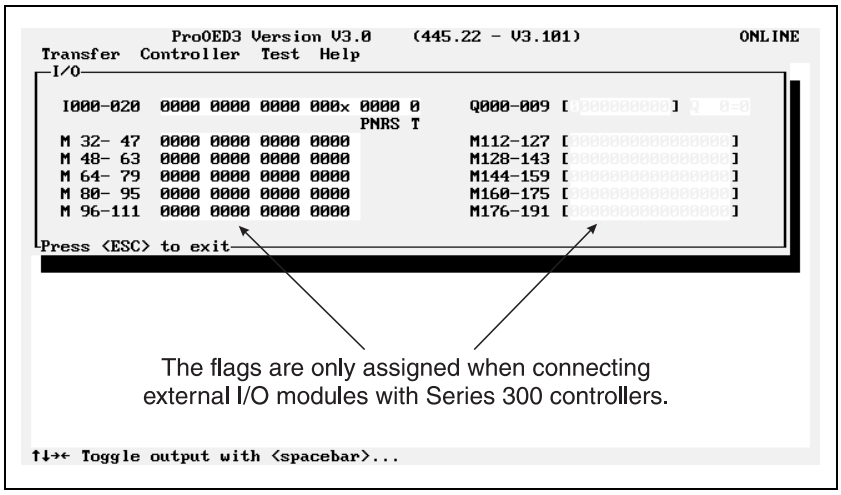


Fig. 6-20 I/O test

Viewing a program during execution (Debug)

ProOED3 echoes the actual program execution on the screen. Current results, variables, flags, etc. can also be displayed. The program can also be executed step by step; see chapter 5.4.3.5, “Debugging the SEQUENCE and PLC program”.

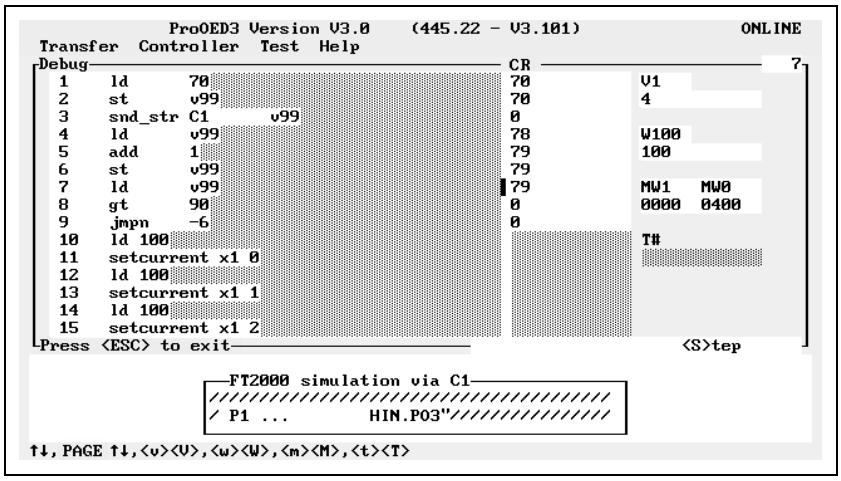


Fig. 6-21 Debug window

Terminal simulation

ProOED3 offers two options for testing data input/output on terminals:

– Simulation of the FT2000 on the screen
– Simulation of a VT52 terminal

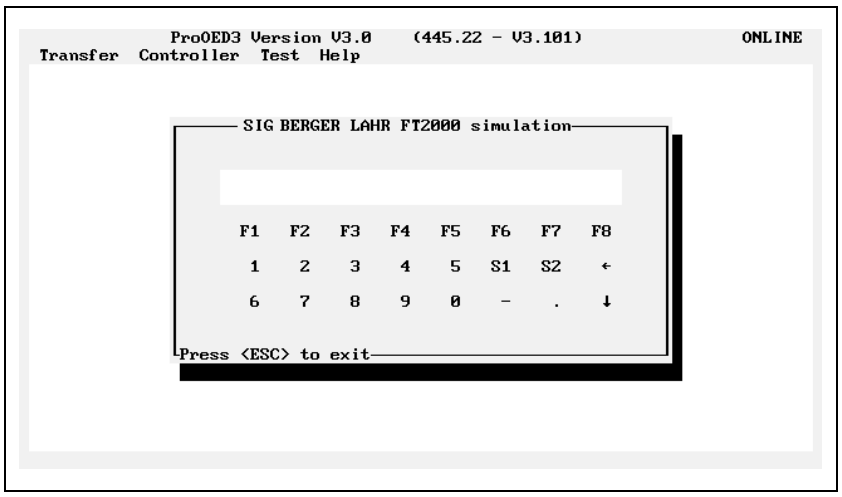


Fig. 6-22 FT2000 simulation

7 Error messages

7.1 Program errors

7.1.1 Compiler errors

Error display	Rectification
No memory available	Exit and restart ProOED3. 555 Kbytes of DOS memory must be available.
Compiler not initialized	Select available controller type from the “Project/Controller Type” menue option
Preprocessor not initialized	Too many symbolic names in assignment list =>Reduce number of symbolic names
Input line too long	Shorten the program line
Output line too long	Shorten the program line
1st macro statement too long	Shorten symbolic names (30 characters max.)
2nd macro statement too long	Shorten symbolic names (30 characters max.)
Too many lines	Reduce number of symbolic names
Too many errors	Eliminate programming errors
$ABL_START not found	Insert $ABL_START at the beginning of the SEQUENCE program component (in second line)
$ABL_END not found	Insert $ABL_END at the end of the SEQUENCE program component (line before last)
$ABL_START already exists	Delete $ABL_START from SEQ program component
$ABL_END already exists	Delete $ABL_END from SEQ program component
Too many lines in SEQ section	Delete lines from SEQUENCE program component
$SPS_START not found	Insert $SPS_START at the beginning of the PLC program component (in second line)
$SPS_END not found	Insert $SPS_END at the end of the PLC program component (line before last)
$SPS_START already exists	Delete $SPS_START from PLC program component
$SPS_END already exists	Delete $SPS_END from PLC program component
Too many lines in PLC section	Shorten PLC program component
$START_OF_OED3_UPLOAD not found	Insert $START_OF_OED3_UPLOAD at the beginning of the program component
$END_OF_OED3_UPLOAD not found	Insert $END_OF_OED3_UPLOAD at the end of the program
$START_OF_OED3_UPLOAD already exists	Delete $START_OF_OED3_UPLOAD from the beginning of the program
$END_OF_OED3_UPLOAD already exists	Delete $END_OF_OED3_UPLOAD from end of program
Unrecognized compiler error	Call Technical Services department, phone: (07821) 946-257
Invalid operator	See chapter 6.1.3.2, command list
Invalid operand	See chapter 6.1.3.2, command list
Operands missing	See chapter 6.1.3.2, command list
Too many operands	See chapter 6.1.3.2, command list
Invalid range of values	See chapter 6.1.3.3, operand list
Invalid operand range of values	See chapter 6.1.3.3, operand list
Invalid operand type	See chapter 6.1.3.2, command list

7.1.2 Errors during program download

The SEQUENCE and PLC programs are tested again for compatibility with the controller during download to the controller.

Download errors are displayed as follows:

`[E####] [L#] $END_OF OED3_DOWNLOAD [Download Error] <ESC>`

[E####] Error code, error type

[L#] Program line in which an error was detected

Error code	Meaning
[E4296]	Invalid operand
[E4297]	Invalid command in SEQUENCE or PLC program
[E4300]	Invalid operand value

NOTE
These errors can occur if the set controller type differs from the controller actually used.

Timeout during reception

A timeout error occuring during program download may be due to any of the following causes:

– Project file corrupted
– Too many lines in SEQUENCE or PLC program
– Link to the controller interrupted

7.1.3 MS-DOS operating system errors

Error display	Rectification
MS-DOS error (...)	ProOED3 file on hard disk is defective. Check file system and hard disk (e.g. using “SCANDISK”). It may be necessary to reinstall ProOED3.

7.1.4 Other ProOED3 error messages

Error message	Possible cause
Connected with controller with old software	OED3 version of controller and ProOED3 version do not match (e.g. OED3 version 2 and ProOED3 version 3)
Controller does not answer	Supply voltage not available or link to controller disrupted
Other error messages	Call Technical Services department, phone: (07821) 946-257

7.2 Runtime errors

Errors occurring during program execution are indicated to the user in two ways:

– On the 7-segment display of the controller (see chapter 7.2.1, “Error codes in the 7-segment display”)
– In an error menue sent via the serial interface c1 to the PC monitor or on the FT2000 operating terminal (see chapter 7.2.2, “Error menue”)

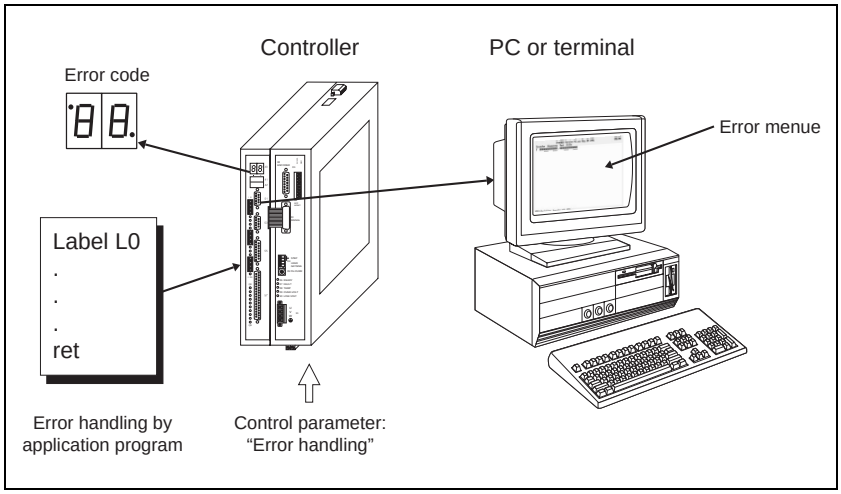


Fig. 7-1 Runtime errors

On controllers with ProOED3 installed, the user can partially determine the error display and the controller response to an error himself.

The control parameter “Error handling” can be used for defining various options for error display and error handling.

Control parameter: “Error handling” (for axis signal errors only)

Parameter: Error handling	Description	Rectification
0	**Error display:** 1. In the 7-segment display on the front panel (see chapter 7.2.1) 2. Error menue via serial interface c1 on FT2000 or PC (see chapter 7.2.2). **Controller:** 1. Motor is stopped. 2. Outputs are reset.	Eliminate the error and acknowledge as follows. **Series 300:** Press key 41 on controller front panel. **WDP3-014/018:** Press “–” key on controller front panel.
1	**Error display:** In the 7-segment display on the front panel (see chapter 7.2.1) **Controller:** Motor stops.	Eliminate the error and acknowledge as follows. **Series 300:** Press key 41 on controller front panel. **WDP3-014/018:** Press “–” key on controller front panel. If “99” is displayed as the error code, you can display the error via the error menue and analyze it (chapter 7.2.2).
2	**Error display:** 7-segment display on front panel shows “98” (see chapter 7.2.1) **Controller:** If the label L0 does not exist, the motor stops. **Program:** Jump to label L0, if it exists.	The axis signal error (see table in chapter 7.2.2) must be handled by the application program (see chapter 7.2.3).

7.2.1 Error codes in the 7-segment display

Runtime errors are identified by an error code in the 7-segment display of the controller. The following table summarizes the possible errors, their causes and methods for rectification.

Display	Cause	Rectification
A ...	Self-test error	Call Technical Services department, phone: (07821) 946-257
03	Motor lead short-circuit	Check the motor wiring
04	Power controller not ready	See power controller troubleshooting table
	Line interruption	Disconnect the unit and check the cable
05	Overvoltage on power controller	Connect a bleed resistor; see controller manual
07	Power controller overtemperature	Let the power controller cool down while the motor is at a standstill
		Install a fan set; see controller manual
08	Error on encoder for electronic gear Line broken	Check encoder wiring
09	Motor overtemperature	Reduce the phase current
		Reduce the load
11	Power controller undervoltage (<200 V)	Check the voltage supply
12	Rotation monitoring active, contouring error	Check mechanical components for ease of movement
14	Power controller without voltage supply	Check voltage supply. Switch on the voltage supply for the power controller first before switching on the voltage supply for the processor unit.
	Internal power controller defective	If switching on is not possible, call Technical Services department, phone: (07821) 946-257.
16	Short-circuit on one output q	Check signal connector wiring
20	Incorrect limit switch LIMP or limit switch malfunction	Check wiring and function of the limit switch or the sense of rotation of the motor; see controller manual. LIMP must be approached with motor running in a clockwise direction.
21	Incorrect limit switch LIMN or limit switch malfunction	Check wiring and function of the limit switch or the sense of rotation of the motor; see controller manual. LIMN must be approached with motor running in a counterclockwise direction.
22	CW limit switch LIMP actuated	Move out of the limit switch range
23	CCW limit switch LIMN actuated	Move out of the limit switch range
26	Reference switch defective or disconnected	Check the reference switch
30	STOP input active	Deactivate the STOP input
40 41 42	Internal errors: 40 = Error during initialization 41 = Error in SEQUENCE component 42 = Error in PLC component	Controller errors, call Technical Services department, phone: (07821) 946-257

Error messages

Display	Cause	Rectification
48	OED3 operating system not found on controller	Call Technical Services department, phone: (07821) 946-257
52	No link via RS 485 HS interface	Check wiring
		Specify the correct number of input/output cards
55	System defective	Call Technical Services department, phone: (07821) 946-257
56	No EEPROM available	Call Technical Services department, phone: (07821) 946-257
57	EEPROM write error	Call Technical Services department, phone: (07821) 946-257
80	Battery voltage low, battery used up  ***ATTENTION*** ***After switching off the controller, data or the application program may be lost!***	Replace the battery; see controller manual
98	Error handling by OED3 application program	Error rectification by OED3 application program.
		Change the “Error handling” control parameter with ProOED3.
99	Error display by ProOED3 error menue	Display error with ProOED3 error menue and eliminate it.

NOTE
If error handling is not effected by the application program, errors can be acknowledged via the front panel keys.

7.2.2 Error menue on operating terminal

Depending on the setting of the “Error handling” parameter, errors can also be displayed in an error menue on a VT52 operating terminal, e.g. FT2000 or PC. The operating terminal or the PC must be connected to the serial interface c1.

The 7-segment display of the controller displays the number “99” while the error menue is output.

The error display in the error menue on the operating terminal shows the error in binary format.

Two lines of the error menue are displayed at a time. To display the next two lines, press the <↵> key.

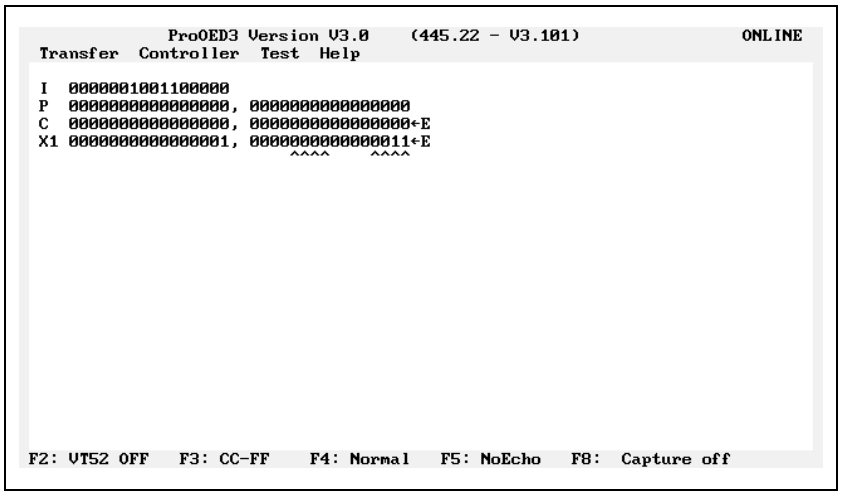


Fig. 7-2 Error menue in ProOED3 terminal

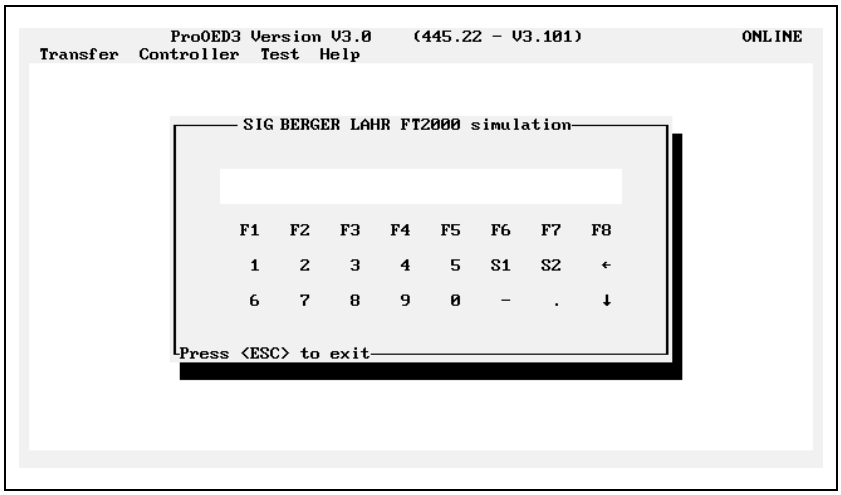


Fig. 7-3 Error menue in FT2000

NOTE
In debugging mode, the FT2000 terminal emulation is automatically displayed on the PC screen in case of an error.
Error display is only effected via the interface c1.

Error display	Meaning
Controller error xxx	**Controller error code** 000 = Program error in line XXX; see page 7-7 219 = Error during RS 485 HS interface initialization 222 = Error in parameter data 233 = Error at “setipos” (too many axes specified) 234 = No EEPROM available 236, 237 = Incorrect data type when reading data with the command “ld” 238, 239 = Incorrect data type when storing data with the command “st” 249 = Label does not exist 250 = Multiple label 251 = Jump outside of application program (e.g. “jmp +3000”) 252 = Incorrect operand data type specified 253 = Unrecognized command 254 = Suprogram call depth exceeded 255 = One or more signals of axis x1 active (see bit template of axis signal errors on page 7-7) 256 = One or more signals of axis x2 active (see bit template of axis signal errors on page 7-7) 257 = One or more signals of axis x3 active (see bit template of axis signal errors on page 7-7) 258 = One or more signals of axis x4 active (see bit template of axis signal errors on page 7-7) 259 = Error on serial interface 260 = Invalid value with “snd_char” 263 = Error with “rec_var_n” or “rec_dez” 264 = “setipos” error: axis specified twice
PLC : xxx SEQ : xxx	Program component and line number in which the error occurred => Display line number in debug mode
I 00000000000000000000 ↑ ↑ Bit 20 Bit 0	**Input signals** Bit 0 = Input 0 (i0) : 15 = Input 15 (i15) 16 = limp 17 = limn 18 = ref 19 = stop 20 = trig
P 0000000000000000, 0000000000000000 ↑ ↑ ↑ ↑ Bit 15 Bit 0 Bit 15 Bit 0 p1 p2	**Encoders p1, p2** Bit 8 = Contouring error 9 = Encoder error

Error display	**Meaning**
C 0000000000000000, 0000000000000000 ↑ ↑ ↑ ↑ Bit 15 Bit 0 Bit 15 Bit 0 C1 C2	**Serial interfaces c1, c2** Bit 2 = Non-initalized interface accessed 4 = Hardware interface already assigned 5 = Error when allocating or deallocating memory 6 = Receive buffer too small 7 = Send string too short 8 = Invalid order in send string 11 = Receive buffer overflow 12 = Overrun error 13 = Parity error 14 = Framing error
X1 0000000000000000, 0000000000000000 X2 0000000000000000, 0000000000000000 X3 0000000000000000, 0000000000000000 X4 0000000000000000, 0000000000000000 ↑ ↑ ↑ ↑ Bit 15 Bit 0 Bit 15 Bit 0 Program error Axis signal error	**Program error in line XXX (axes x1 to x4)** Bit 0 = One or more signals active (see axis signals) 2 = Command not permitted in position following mode 3 = Wait positioning already active (e.g. “posf”, “movef”) 4 = Command not permitted with interrupted/blocked axis 5 = Invalid master curve 6 = Insufficient information on reference variable 7 = Command not permitted during an axis movement 8 = Command not permitted during a reference movement 10 = Incorrect parameter value 11 = Value cannot be calculated 12 = Invalid parameter value specified 13 = Command cannot be executed under the prevailing conditions 14 = Undefined value requested **Axis signals x1 to x4 (axis signal errors)** Bit 0 = Positive hardware limit switch (20, 22) 1 = Negative hardware limit switch (21, 23) 2 = Reference switch defective (26) 3 = Hardware stop (30) 4 = Reserved 5 = Reserved 6 = Reserved 7 = Reserved 8 = Contouring error (12) 9 = Encoder error (08) 10 = Power controller not ready (04) 11 = Power controller overtemperature (05) 12 = Motor overtemperature; only for motors with encoders (09) The numbers in parentheses reflect the error code in the 7-segment display of the controller.

NOTE
Positioning operations are interrupted by axis signal errors.
Axis signal errors can be handled by the application program; see chapter 7.2.3.

7.2.3 Error handling by application program

The control parameter “Error handling” can be used for enabling the user to program the response to an axis signal error himself (see chapter 7.2.2, “Error menue”, table).
When an axis signal error occurs, the corresponding axis and the SEQUENCE program are stopped and the subprogram “L0” is called automatically.

During subprogram processing, the 7-segment display of the controller displays the code “98”. Subsequently the SEQUENCE program is resumed from the point where it was interrupted.

NOTE
The PLC program is not stopped by an axis signal error.

If the subprogram label L0 does not exist, the code “99” is displayed in the 7-segment display. You can then analyze the error using the error menue.

The variable v0 contains the number of the axis on which the error occurred.

The variable v100 contains the error code from which you can determine the cause of the error (see below).

Subprogram “L0”

The error handling program must be programmed in a subprogram with the label L0.

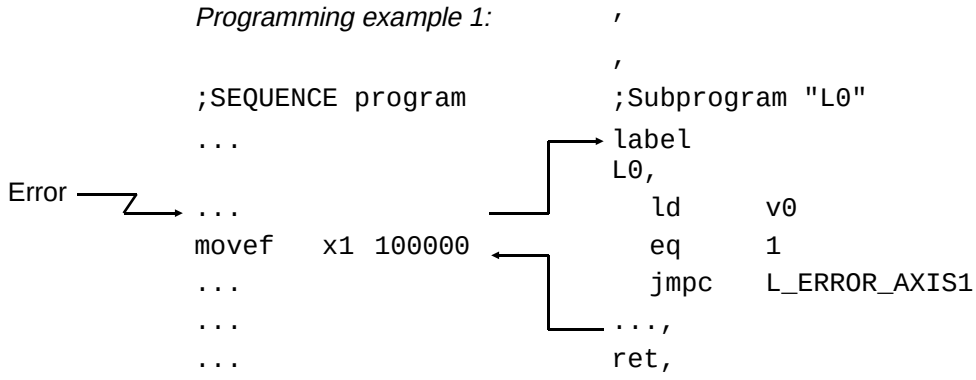
Procedure:

1. Set the control parameter “Error handling” to 2.
2. Program the response to the error in the subprogram “L0” within the SEQUENCE program.
3. Use “clrerror” to clear the axis signal error and the error display.

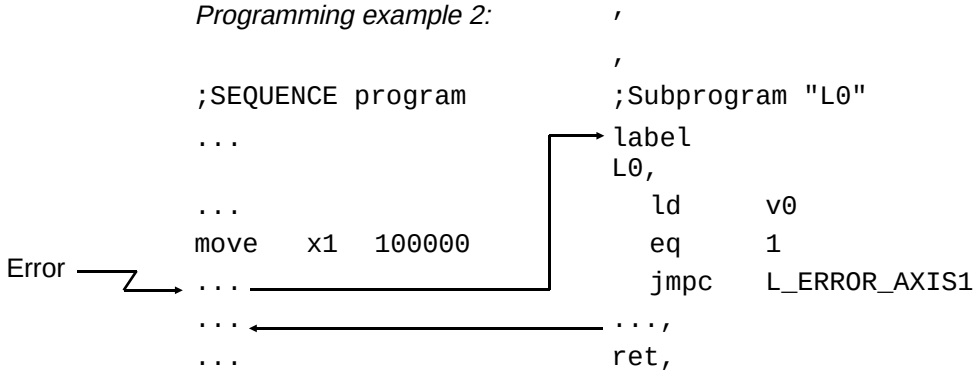
The subprogram jump to the label L0 is executed automatically. At the end of the subprogram (“ret” command), program execution is resumed at the point where it was interrupted.

NOTE
In case of positioning commands with waiting phase, the point of program interruption is known (one program line after the command which triggered the error). With all other positioning commands, the program continues to be executed in parallel to axis positioning; it is therefore not possible to determine the exact point of interruption in the program.

Programming example 1:



Programming example 2:



The variables v0 and v100 can be used for determining the cause of the error.

Variable v0

The variable v0 receives the number of the axis which caused the error in case of an error in the SEQUENCE program.
If an error occurs, the corresponding axis can be moved using the manual movement flags m0 to m4.

```
movef   v0   <position>
```

to move the axis.

Programming example:

```
;Determining the axis which caused the error.
ld     v0,
eq     1,
jmpc   L_ERROR_AXIS1    ;Error on axis x1
ld     v0,
eq     2,
jmpc   L_ERROR_AXIS2    ;Error on axis x1
...    ,
ret    ,
```

Variable v100

When an axis error occurs, the variable v100 is assigned an error code from which you can determine the cause of the error.
The error codes correspond to the codes in the 7-segment display of the controller (see above).

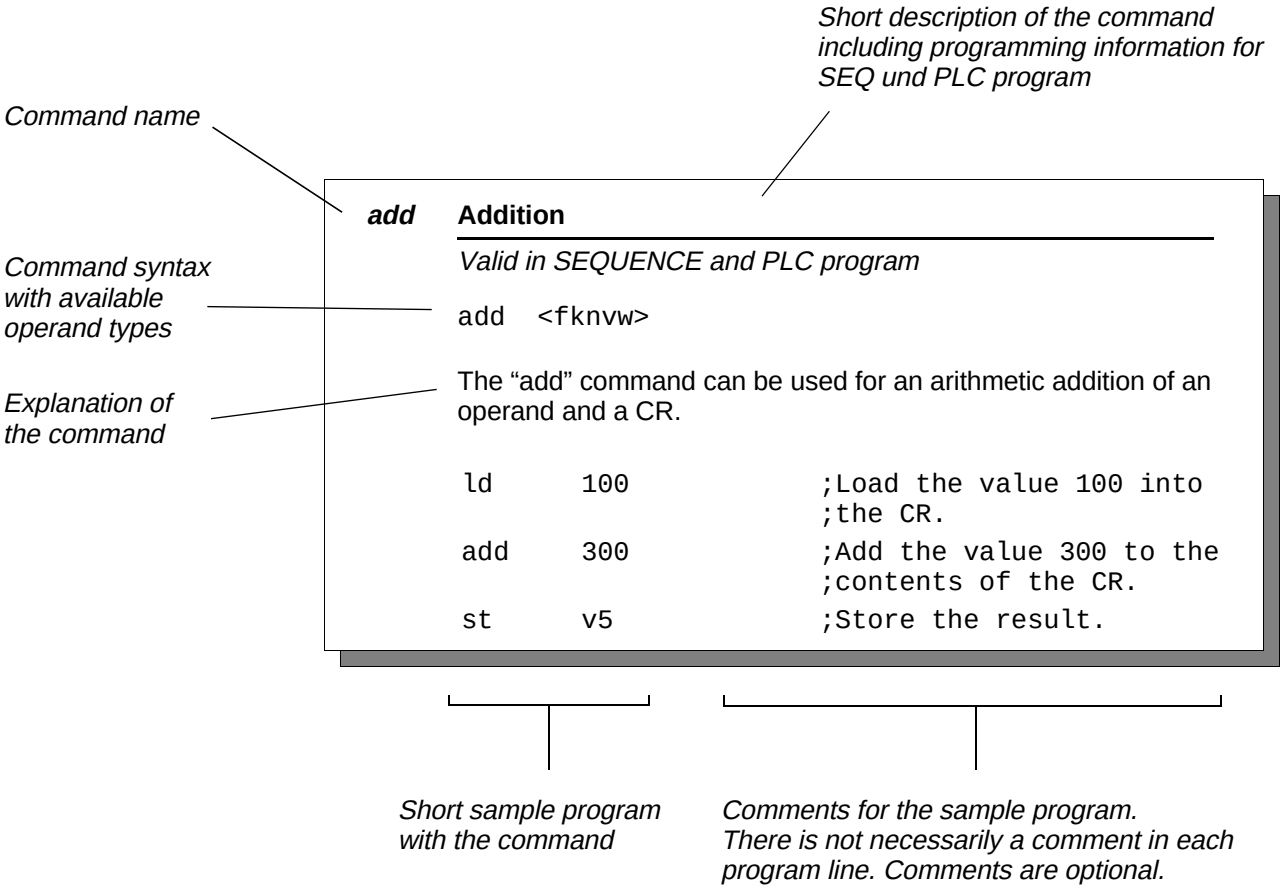
Programming example:

```
;Determining the cause of an error
label       L_ERROR_AXIS1
  ld        v100,
  eq        22              ;Positive limit switch
  jmpc      L_LIMITSWITCH   ;error
  ld        v100,
  eq        23              ;Negative limit switch
  jmpc      L_LIMITSWITCH   ;error
  ...       ,
;Error handling for limit switch error
label       L_LIMITSWITCH,
  clrerror  x1              ;Clear error display
  jmpc      -1,
  snd_str   c1              T_ERROR
  ...       ,
  ret       ,
```

8 Appendix

8.1 Command description

This chapter describes the commands in alphabetical order as illustrated in the example below.



acc Maximum acceleration

Valid in SEQUENCE program

```
acc  <x>  <fknvw>
```

The “acc” command is used for specifying the maximum acceleration for calculating the selected acceleration and deceleration curve (“Ramp” parameter). In the subsequent movements, the corresponding axis x1 to x4 is accelerated and decelerated using this ramp.
If an “acc” command is not specified, the ramps are calculated using the “Standard acceleration” control parameter by default.

NOTE
The “acc” command is only valid when the axis is at a standstill.

```
acc     x1      125     ;Ramp of axis x1 with
                        ;125 Hz/ms maximum
                        ;acceleration or
                        ;deceleration.
```

add Addition

Valid in SEQUENCE and PLC program

```
add  <fknvw>
```

The “add” command can be used for an arithmetic addition of an operand and a CR.

```
ld      100             ;Load the value 100 into the
                        ;CR.
add     300             ;Add the value 300 to the
                        ;contents of the CR.
st      v5              ;Store the result in the
                        ;variable v5.
```

amp *Switching the power controller on and off*

Valid in SEQUENCE program

```
amp  <x>  <k>
```

The “amp” command is used for switching the motor power controller on or off. The CR contains the current status of the power controller after executing the command.
TRUE (<> 0) = Power controller ready
FALSE (=0) = Power controller off

Programming example:

```
amp     x1      1       ;Switch power controller on
jmpn    -1              ;and wait
                        ;until it is active
...
amp     x1      0       ;Switch power controller off
jmpc    -1              ;and wait
                        ;until it is off
```

and AND operation

Valid in SEQUENCE and PLC program

```
and   <iqm>
andn  <iqm>
```

The “and” command can be used for an AND operation with a Boolean operand and the CR.
The “n” letter can by used for negating the operand before the operation.

Programming example 1:

```
ld       i1       ;Load input i1 into the CR.
and      i2       ;AND operation with input
                  ;i1 and i2.
                  ;CR = i1 and i2
st       m25      ;Store the contents of the CR
                  ;in the flag m25.
```

Programming example 2:

```
ld       i1       ;Load input i1 into the CR.
andn     i3       ;Negated AND operation
                  ;between input i3 and i1.
                  ;CR = i1 andn i3
st       m25      ;Store the contents of the CR
                  ;in the flag m25.
```

brake
(Series 300 only)

Defining an output for controlling a brake

Valid in SEQUENCE program

```
brake  <x>
```

The “brake” command is used for defining an output for controlling a brake.
The output is set or reset depending on the READY (power controller readiness) and ENABLE (power controller enable) signals.

ENABLE	READY	Output for brake
0	0	0
0	1	0
1	0	0
1	1	1

The outputs q4 to q7 are permanently assigned to the axes for the braking function as follows:

Axis	Output for brake
x1	q4
x2	q5
x3	q6
x4	q7

```
brake    x1       ;Output q4 used as an output for
                  ;the motor brake of axis x1
brake    x2       ;Output q5 used as an output for
                  ;the motor brake of axis x2
brake    x3       ;Output q6 used as an output for
                  ;the motor brake of axis x3
brake    x4       ;Output q7 used as an output for
                  ;the motor brake of axis x4
```

cal Call a subprogram

Valid in SEQUENCE program

```
cal   <L>
calc  <L>
caln  <L>
```

The “cal” command is used for calling a subprogram starting at a defined label. A maximum of seven subprograms can be nested. The “calc” and “caln” commands can be used for programming conditional subprogram calls.

Programming example 1:

```
cal     L10             ;Jump to subprogram at
                        ;label L10.
...
label   L10             ;Subprogram start
...
ret                     ;Return from subprogram
```

Programming example 2:

```
ld      i10             ;Call a subprogram
calc    L20             ;if i10 = 1 (TRUE)
ld      i10             ;Call a subprogram
caln    L20             ;if i10 = 0 (FALSE)
```

clrerror Reset axis signal error

Valid in SEQUENCE program

```
clrerror  <x>
```

You can use the “clrerror” command to reset the axis signal errors LIMP, LIMN, STOP, contouring error and power controller failure for an axis. Any flashing error display in the front panel display is cleared.
When calling the command, the CR indicates if there is still a signal error active on the axis (CR <>0) or if no signal errors are active (CR = 0).

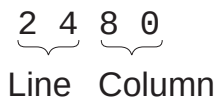
```
label       L0              ;Subprogram start
clrerror    x1              ;Reset signal error
                            ;on axis 1
jmpc        -1              ;Wait until no error
                            ;is active
ret
```

cursor Screen cursor positioning (VT52 terminal)

Valid in SEQUENCE program

```
cursor  <c>  <fknvw>
```

The “cursor” command positions the cursor of a connected VT52 terminal. The terminal may be either an FT2000 (2 lines, 40 characters) or a normal VT52 terminal (24 lines, 80 characters). The <c> parameter specifies the interface to which the terminal is connected. The value of the parameter <k, n, v or w> defines the position of the cursor. The digits representing hundreds and thousands specify the line position. The digits representing tens and ones define the column position.




```
cursor  c1      120     ;Positions the cursor of the
                        ;terminal connected to
                        ;interface c1 to line 1 in
                        ;column 20.
cursor  c1      101     ;Cursor position: line 1,
                        ;column 1
cursor  c1      2480    ;Cursor position: line 24,
                        ;column 80
```

div Division

Valid in SEQUENCE and PLC program

```
div  <fknvw>
```

The “div” command can be used for a division of an operand and the CR.

```
ld      100     ;Load the value 100 into the CR.
div     20      ;Divide the contents of the CR
                ;(100) by the value 20.
st      v4      ;Store the result in variable v4.
```

end Program end

Valid in SEQUENCE and PLC program

```
end
```

The “end” command is used for terminating the SEQUENCE or PLC program. The program pointer returns to the program beginning and the program restarts.

NOTE
The “end” command is automatically inserted at the end of the program when downloading.

```
end                ;Program end and return to
                   ;program beginning
```

eq Relation (equal to)

Valid in SEQUENCE and PLC program

```
eq  <fknvwiqm>
```

The “eq” command can be used for a relational operation (=) between an operand and the CR. After the operation, the CR is either 0 (FALSE) or 1 (TRUE).

CR = 0 (FALSE), if the result is not true or

CR = 1 (TRUE), if the result is true.

```
ld       v10       ;Load variable v10 into the CR.
eq       100       ;Perform relational operation.
                   ;CR = 100?
                   ;If yes, the CR is 1, otherwise
                   ;it is 0.
```

gearn Gear ratio denominator

Valid in SEQUENCE program

```
gearn  <x>  <fknvw>
```

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

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

The encoder units depend on the setting of the parameter “Encoder evaluation DG1 or DG2”.

NOTE
The gear ratio is accepted with the “gearz” command.

```
gearn    x1     4       ;Multiply the position
gearz    x1     3       ;value supplied to the
                        ;encoder input with
                        ;a gear ratio of 3/4
mode     x1     1       ;Position following mode
                        ;to be set via encoder
                        ;input.
label    L1             ;Pulses can now be
jmp      -1             ;supplied through the
                        ;encoder input.
```

gearz — *Gear ratio numerator*

Valid in SEQUENCE program

```
gearz  <x>  <fknvw>
```

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

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

The encoder units depend on the setting of the parameter “Encoder evaluation DG1 or DG2”.

NOTE
The gear ratio is accepted with the “gearz” command.

```
gearn    x1     4       ;Multiply the position
gearz    x1     3       ;value supplied to the
                        ;encoder input with
                        ;a gear ratio of 3/4
mode     x1     1       ;Position following mode
                        ;to be set via encoder
                        ;input.
label    L1             ;Pulses can now be
jmp      -1             ;supplied through the
                        ;encoder input.
```

getanalog (Series 300 only) — *Read analog voltage*

Valid in SEQUENCE program

```
getanalog  <a>  <knvw>
```

The “getanalog” command can be used for reading a voltage (mV) from an analog module (a2) into the CR.
The second operand is used for specifying the channel number of the analog input.

```
getanalog   a2    1     ;Voltage from analog module
                        ;(channel no. 1) to be
                        ;read into the CR.
```

getport — *Read inputs or flags and convert into a number*

Valid in SEQUENCE program

```
getport  <im>  <fknvw>
```

The “getport” command is used for interpreting a sequence of input signals or flags as a binary value and converting it into an integer number. The result is stored in the CR.
The <i> or <m> operand defines the input or flag with which the bit sequence starts. The second operand specifies the number of input signals or flags representing the binary-code value.

```
getport     i3    4     ;The input signals i3,
                        ;i4, i5 and i6 represent
                        ;binary values which
                        ;are converted to a decimal
                        ;value and stored in
                        ;the CR.
getport     m32   8     ;The flags m32 to m39
                        ;represent binary values
                        ;which are converted to a
                        ;decimal value and stored
                        ;in the CR.
```

goff Position offset for position following mode

Valid in SEQUENCE program

```
goff  <x>  <fknvw>
```

The “goff” command is used in position following mode (electronic gear) for superimposing a position value for the axes x1 to x4 (position offset). The position offset is interpreted as an absolute position and set to zero upon changing to position following mode.

```
gearn    x1    4      ;Set gear ratio
gearz    x1    3      ;to 3/4.
mode     x1    1      ;Axis x1 is set for
                      ;position following mode
                      ;via encoder.
goff     x1    1000   ;Add a position offset of
                      ;1000 to the current
                      ;position value of axis x1
                      ;and execute the
                      ;positioning operation.
goff     x1    2000   ;The axis moves by
                      ;another 1000 steps.
label    L1           ;Pulses can now be
jmp      -1           ;supplied through the
                      ;encoder input.
```

gt Relation (greater than)

Valid in SEQUENCE and PLC program

```
gt  <fknvw>
```

The “gt” command (greater than) can be used for a relational operation (>) between an operand and the CR. After the operation, the CR is either 0 (FALSE) or 1 (TRUE).

CR = 0 (FALSE), if the result is not true or
CR = 1 (TRUE), if the result is true.

```
ld       v10          ;Load variable v10 into the
                      ;CR.
gt       1500         ;CR > 1500 ?
                      ;If yes, the
                      ;CR = 1,
                      ;otherwise it is 0.
```

handshake — *Synchronization with a master controller*

Valid in SEQUENCE program

```
handshake  <i>  <q>
```

The “handshake” command can be used for synchronization with a master controller. The application program stops and waits until the synchronization input is energized (= 1). When the input is energized, the controller resumes execution of the application program and the synchronization output is reset. If the synchronization input is 0, the synchronization output is set to 1.

```
handshake   i10   q8    ;The program is
                        ;stopped until i10 = 1.
                        ;During program stop, the
                        ;output q8 = 1, otherwise
                        ;it is 0.
```

jmp Program jump

Valid in SEQUENCE and PLC program

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

The “jmp” command can be used for performing conditional and unconditional jump operations. The additional letters “c” and “n” are used for conditional jump operations. “k” defines a relative jump operation by k lines.
Jump instructions with “c” are only executed if the value in the CR is not equal to 0.
Jump instructions with “n” are only executed if the value in the CR is equal to 0.

Programming example 1:

```
label    L1       ;Assign label L1.
ld       v10      ;Load variable v10 into the CR.
add      10       ;Add the value 10 to the CR.
st       v100     ;Store the result in variable
                  ;v100.

...
jmp      L1       ;Jump to label L1.
```

Programming example 2:

```
ld       i12      ;Load input i12 into the CR.
jmpc     L10      ;If i12 = 1, jump to
                  ;label L10.
...
label    L10      ;Assign label L10.
```

Programming example 3:

```
ld       i10      ;Load input i10 into the CR.
jmpn     -1       ;If i10 = 0, return by one line.
```

label Program label

Valid in SEQUENCE and PLC program

```
label  <L>
```

The “label” command is used for assigning a label for jump instructions and subprogram calls.

```
label    L1       ;Assign label L1.
ld       v10      ;Load variable v10 into the CR.
add      10       ;Add the value 10 to the CR.
st       v100     ;Store the result in variable
                  ;v100.
jmp      L1       ;Jump to label L1.
```

ld Load operand into the CR

Valid in SEQUENCE and PLC program

```
ld   <fknvwiqmxtp>
ldn  <iqm>
```

The “ld” command can be used for loading an operand into the CR.
The “n” letter can by used for subjecting the operand to a Boolean negation.

NOTE
The “ld x..” command can be used for reading the current axis position, the “ld p..” command for reading the current encoder position.

Programming example 1:

```
ld       i1       ;Load input i1 into the CR.
st       m25      ;Store contents of CR (input i1)
                  ;as flag m25.
```

Programming example 2:

```
ldn      i1       ;Load negated i1 into the CR.
```

Programming example 3:

```
ld       x1       ;Read current axis position
st       v10      ;and store it in variable V10.
```

Programming example 4:

```
ld       p2       ;Read current encoder position
st       v10      ;and store it in variable V10.
```

ld_LBit (Series 300 only)

Read bit from data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
ld_LBit   <knvw>    <knvw>
```

The "ld_LBit" command is used for reading a bit from the data interface of the Lauer operating panel into the CR.
The first parameter, <knvw>, specifies the number of the word from which to read the bit. The data interface comprises 256 words, i.e. the range of values of the first parameter is from 0 to 255.
The second parameter, <knvw>, specifies the number of the bit within the specified word (16 bits). The range of values of the second parameter is from 0 to 15.

Programming examples:

```
ld_LBit    100, 0     ;Read bit 0 in word 100
ld_LBit    100, 1     ;Read bit 1 in word 100
ld_LBit    100, 15    ;Read bit 15 in word 100
ld_LBit    255, 0     ;Read bit 0 in word 255
ld_LBit    255, 1     ;Read bit 1 in word 255
```

ld_LDint (Series 300 only)

Read double word from data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
ld_LDint  <knvw>
```

The "ld_LDint" command is used for reading a double word from the data interface of the Lauer operating panel into the CR.
The <knvw> parameter specifies the number of the double word to be read from the data interface of the Lauer operating panel. The range of values of the parameter is from 0 to 254.
The words 254 and 255 form the last double word in the data interface range.

Programming examples:

```
ld_LDint   0          ;Read double word 0
                      ;(word 0 and word 1)
ld_LDint   1          ;Read double word 1
                      ;(word 1 and word 2)
ld_LDint   254        ;Read double word 254
                      ;(word 254 and word 255)
```

ld_LInt (Series 300 only)

Read word from data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
ld_LInt   <knvw>
```

The “ld_LInt” command is used for reading a word from the data interface of the Lauer operating panel into the CR.
The <knvw> parameter specifies the number of the word to be read from the data interface of the Lauer operating panel. The data interface comprises 256 words, i.e. the range of values of the parameter is from 0 to 255.

Programming examples:

```
ld_LInt    100        ;Read word 100
ld_LInt    101        ;Read word 101
ld_LInt    255        ;Read word 255
```

ld_LKey (Series 300 only)

Read key status (keyboard bit) of Lauer operating panel

Valid in SEQUENCE and PLC program

```
ld_LKey   <knvw>
```

The “ld_LKey” command is used for reading the status of a key on the Lauer operating panel (keyboard bit) directly into the CR.
When the corresponding key is pressed, the keyboard bit has the value 1, otherwise the value 0.
The <knvw> parameter specifies the number of the keyboard bit. Valid values for this parameter are from 0 to 31 only.
The bit number assigned to the operating panel key is defined by programming the operating panel (see Lauer documentation).

Programming example 1:
To check the key with number 10

```
ld_LKey  10       ;Load keyboard bit 10 into the CR
eq       1        ;If key is pressed ...
jmpc     label    ;Jump to "label"
...
```

Programming example 2:
To check the key with number 12
(variable v10 is used for storing the key number)

```
ld         12      ;Load value 12 and
st         v10     ;store it in variable v10
ld_LKey    v10     ;Load keyboard bit 12 into CR
eq         1       ;If key is pressed ...
jmpc       label   ;Jump to "label"
...
```

linmove (Series 300 only)

Linear interpolation to relative target position without waiting

Valid in SEQUENCE program

```
linmove
```

The “linmove” command is used in point-to-point mode for starting a defined linear interpolation process (see “setipos” command) to a relative target position. Program execution continues during the axis movement. The preset position cannot be changed during positioning.

```
setipos    x1    1000    ;Predefine positions in
setipos    x2    2000    ;user-defined units for the
setipos    x3    3000    ;axes x1 to x3 which are
                         ;involved in the
                         ;interpolation.
linmove                  ;Start the linear
                         ;interpolation to the
                         ;relative target position.
ld         m1001         ;Wait until the axes
or         m1002         ;involved in interpolation
or         m1003         ;have come to a standstill.
jmpc       -3
```

linmovef (Series 300 only)

Linear interpolation to relative target position with waiting

Valid in SEQUENCE program

```
linmovef
```

The “linmovef” command is used in point-to-point mode for starting a linear interpolation process (see “setipos” command) to a relative target position, and the SEQUENCE program is stopped until the target position is reached.

```
setipos    x1    1000    ;Predefine positions in
setipos    x2    2000    ;user-defined units for the
                         ;axes x1 to x2 which are
                         ;involved in the
                         ;interpolation.
linmovef                 ;Start the linear
                         ;interpolation to the
                         ;relative target position
                         ;and wait until the target
                         ;position is reached.
```

linpos (Series 300 only)

Linear interpolation to absolute target position without waiting

Valid in SEQUENCE program

```
linpos
```

The “linpos” command is used in point-to-point mode for starting a defined linear interpolation process (see “setipos” command) to an absolute target position. Program execution continues during the axis movement. The preset position cannot be changed during positioning.

NOTE
Before starting absolute positioning operations, a reference movement or dimension setting process should be carried out (see chapter 6.3.2.2 on point-to-point mode).

```
setipos    x1    5000   ;Predefine positions in
setipos    x2    4000   ;user-defined units for the
setipos    x3    3000   ;axes x1 to x3 which are
                        ;involved in the
                        ;interpolation.
linpos                  ;Start the linear
:                       ;interpolation to the
                        ;absolute target position.
ld         m1001        ;Wait until the axes
or         m1002        ;involved in interpolation
or         m1003        ;have come to a standstill.
jmpc       -3
```

linposf (Series 300 only)

Linear interpolation to absolute target position with waiting

Valid in SEQUENCE program

```
linposf
```

The “linposf” command is used in point-to-point mode for starting a predefined linear interpolation process (see “setipos” command) to an absolute target position, and the application program is stopped until the target position is reached.

NOTE
Before starting absolute positioning operations, a reference movement or dimension setting process should be carried out (see chapter 6.3.2.2 on point-to-point mode).

```
setipos    x1    5000   ;Predefine positions in
setipos    x2    4000   ;user-defined units for the
                        ;axes x1 to x2 which are
                        ;involved in the
                        ;interpolation.
linposf                 ;Start the linear
                        ;interpolation to the
                        ;absolute target position
                        ;and wait until the target
                        ;position is reached.
```

lt Relation (less than)

Valid in SEQUENCE and PLC program

```
lt  <fknvw>
```

The “lt” command (less than) can be used for a relational operation (<) between an operand and the CR. After the operation, the CR is either 0 (FALSE) or 1 (TRUE).

CR = 0 (FALSE), if the result is not true or
CR = 1 (TRUE), if the result is true.

```
ld      v11     ;Load variable v11 into the CR.
lt      1800    ;If CR < 1800, the
                ;CR = 1, otherwise it is 0.
```

mode Set axis operating mode

Valid in SEQUENCE program

```
mode  <x>  <k>
```

The “mode” command is used for setting the axis operating mode. This command is only valid when the axis is at a standstill.

0 Point-to-point mode
1 Position following mode via encoder input (electronic gear)
The pulses supplied at the encoder input are converted into a motor movement.
2 Position following mode via variable
The value of the variable is converted into a motor movement.

NOTE
A gear ratio for an electronic gear must be set before activating position following mode.

With position following mode via variables, position following variables are predefined for the axes x1 to x4.

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

NOTE
Point-to-point mode is always active after controller power-on.

The following examples illustrate axis operating mode programming.

Programming example 1:

```
ld      0           ;Load the value 0 into the
                    ;CR.
st      x1          ;Contents of the CR
                    ;stored as position value
                    ;for axis x1.
mode    x1    0     ;Axis x1 is set to
                    ;point-to-point mode.
```

Programming example 2:

```
gearn   x1    3     ;Gear ratio
gearz   x1    4     ;set to 3/4.
mode    x1    1     ;Axis x1 is in position
                    ;following mode via
                    ;encoder.
```

Programming example 3:

```
gearn   x1    3     ;Gear ratio
gearz   x1    4     ;set to 3/4.
mode    x1    2     ;Axis x1 is in position
                    ;following mode via
                    ;variable.
```

move Relative positioning without waiting

Valid in SEQUENCE program

```
move  <x>  <fknvw>
```

The “move” command is used in point-to-point mode for presetting a relative target position and starting a positioning operation. Program execution continues during the axis movement. The preset position can be changed during positioning.

NOTE
During an active relative positioning operation without waiting the “move” command can be used for specifying a new target position.
The position value passed with the “move” command is added to the previously specified target position in order to calculate the new target position.

During positioning of axes x1 to x4, the movement states (positioning, stopped) of the corresponding motor are stored in flags.

Flag	Function	
	0	1
m1001	Axis x1 stopped	Axis x1 positioning
m1002	Axis x2 stopped	Axis x2 positioning
m1003	Axis x3 stopped	Axis x3 positioning
m1004	Axis x4 stopped	Axis x4 positioning

```
move    x1      500     ;500 user-defined units are
                        ;specified as the relative
...                     ;target position for
                        ;axis x1, and a new absolute
                        ;target position is
                        ;calculated.
...
ld      m1001           ;Wait until axis x1 has
jmpc    -1              ;stopped
```

movef Relative positioning with waiting

Valid in SEQUENCE program

```
movef  <x>  <fknvw>
```

The “movef” command is used in point-to-point mode for defining a relative target position and starting a positioning operation, and the SEQUENCE program is stopped until the target position is reached. The signal states of the flags m1001 to m1004 are the same as for the “move” command.

```
movef   x1      1000    ;1000 user-defined units
                        ;are specified as the
                        ;relative target position
                        ;for axis x1, and the
                        ;program waits until the
                        ;calculated new absolute
                        ;target position is reached.
```

mul Multiplication

Valid in SEQUENCE and PLC program

```
mul  <fknvw>
```

The “mul” command can be used for a multiplication of an operand and the CR.

```
ld      100             ;Load the value 100 into the
                        ;CR.
mul     20              ;Multiply the contents of
                        ;the CR (100)
                        ;with the value 20.
st      v3              ;Store the result in the
                        ;variable v3.
```

nop Dummy command

Valid in SEQUENCE and PLC program

```
nop
```

The “nop” command is a dummy command without a function (execution time approx. 500 µs – 1000 µs).

or OR operation

Valid in SEQUENCE and PLC program

```
or   <iqm>
orn  <iqm>
```

The “or” command can be used for an OR operation with a Boolean operand and the CR.
The letter “n” can be used for negating the operand before the operation.

Programming example 1:

```
ld       i1       ;Load input i1 into the CR.
or       i2       ;OR operation between
                  ;inputs i1 and i2.
                  ;CR = i1 or i2
st       m25      ;Store the contents of the CR
                  ;in the flag m25.
```

Programming example 2:

```
ld       i1       ;Load input i1 into the CR.
and      i2       ;AND operation with inputs
                  ;i1 and i2.
                  ;CR = i1 and i2.
orn      i3       ;Negated OR operation
                  ;between CR and input i3.
                  ;CR = CR orn i3
st       m25      ;Store the contents of the CR
                  ;in the flag m25.
```

pos Absolute positioning without waiting

Valid in SEQUENCE program

```
pos  <x>  <fknvw>
```

The “pos” command is used in point-to-point mode for presetting an absolute target position and starting a positioning operation. Program execution continues during the axis movement.
The signal states of the flags m1001 to m1004 are the same as for the “move” command.

NOTE
During an active absolute positioning operation without waiting, the “pos” command can be used for specifying a new absolute target position.

NOTE
Before starting absolute positioning operations, a reference movement or dimension setting process should be carried out (see chapter 6.3.2.2 on point-to-point mode).

```
pos     x1      1000    ;1000 user-defined units
                        ;are specified as the
...                     ;absolute target position
                        ;for axis x1.
ld      m1001           ;Wait until axis x1 has
jmpc    -1              ;stopped
```

posf Absolute positioning with waiting

Valid in SEQUENCE program

```
posf  <x>  <fknvw>
```

The “posf” command is used in point-to-point mode for defining an absolute target position, and the SEQUENCE program is stopped until the target position is reached.
The signal states of the flags m1001 to m1004 are the same as for the “move” command.

NOTE
Before starting absolute positioning operations, a reference movement or dimension setting process should be carried out (see chapter 6.3.2.2 on point-to-point mode).

```
posf    x1      2000    ;2000 user-defined units
                        ;are specified as the
                        ;absolute target position
                        ;for axis x1, and the
                        ;program waits until the
                        ;new target position is
                        ;reached.
```

r — Set output or flag to 0

Valid in SEQUENCE and PLC program

```
r  <qm>
```

The “r” command can be used for resetting an output or flag to 0, based on the current result. The command is only executed if the CR = 1.

```
ld          i1          ;Load input i1 into the CR.
and         i2          ;AND operation with
                        ;inputs i1 and i2.
                        ;CR = i1 and i2.
r           q5          ;Output q5 is set to 0
                        ;if CR = 1.
```

rec_char — Receive character from the serial interface and wait

Valid in SEQUENCE program

```
rec_char  <c>  <fnvw>
```

The “rec_char” command reads a character from the interface <c> and stores its ASCII decimal value into the variable <n>, <v> or <w>.

```
rec_char    c1      v20     ;Reads a character from
                            ;interface c1 and
                            ;stores the decimal
                            ;value in the
                            ;variable v20.
```

rec_char_n — Receive a character from the serial interface without waiting

Valid in SEQUENCE program

```
rec_char_n  <c>  <fnvw>
```

The “rec_char_n” command reads a character from the interface <c> and stores its ASCII decimal value into the variable <n>, <v> or <w>.

NOTE
If no character is available, the command returns -1 in the CR.

```
rec_char_n  c1      v20     ;Reads a character
                            ;from interface c1
                            ;and stores the decimal
                            ;value in the
                            ;variable v20.
eq          -1              ;If no character
jmpc        L10             ;is available, jump to
                            ;label L10.
```

rec_dez — *Receive numbers with decimal places*

Valid in SEQUENCE program

```
rec_dez   <c>  <fnvw>
```

The rec_dez command can be used for reading in a numerical value with decimal places via the serial interface. The number received is stored in a variable (f, v, w, n) as a 32-bit number. The allowed number of decimal places is specified in the “Decimal point” control parameter.
You may only enter the ASCII characters <0 – 9> <CR> and, as the first character, a <->. The <DEL> code (“←” key on FT2000) or <STRG>+<Y> deletes all characters entered in the display.

Control parameters:

```
Decimal point           2
```

```
label      L10
rec_dez    c1     v10    ;Receive number with
                         ;2 decimal places and store
                         ;it in the variable v10.
eq         -1             ;If no character available,
jmpc       L10            ;go to label L10.
```

Input example: Value in v10

```
200260     <CR>   20026000
200260     <CR>   2002600
-2002.6    <CR>   200260
-2002.60   <CR>   200260
<CR>               Value remains unchanged
<Strg_Y>           Value remains unchanged
```

rec_var — *Output, edit and receive number via the serial interface*

Valid in SEQUENCE program

```
rec_var  <c>  <fnvw>
```

The “rec_var” command outputs the decimal number stored in <n>, <v> or <w> via the serial interface <c> on a VT52 terminal (FT2000). This number can then be edited on the terminal. The control code <CR> finishes number input and stores the new number in the corresponding variable.

```
ld          1024        ;Outputs the decimal
st          v10         ;number 1024
rec_var     c1    v10   ;on a terminal at the
                        ;interface c1.
                        ;The number can be edited.
                        ;<CR> finishes the input
                        ;and stores the number
                        ;in the variable v10.
```

rec_var_n — *Receive number via serial interface and store it in a variable*

Valid in SEQUENCE program

```
rec_var_n  <c>  <fnvw>
```

The “rec_var_n” command reads a string from the interface <c> and converts it into a number. The value is stored in the variable <f>, <n>, <v> or <w>. This command is used for entering numbers quickly without outputting them previously via the interface.
Number input is finished with the control code <CR>.

NOTE
When the “rec_var_n” command is issued, a question mark is output as a prompt via the interface.
Positive and negative numbers are valid for entry.

```
rec_var_n   c1    v10   ;Reads a decimal
                        ;number from the
                        ;interface c1 into
                        ;the variable v10.
```

ref Reference movement without waiting

Valid in SEQUENCE program

```
ref  <x>
```

The “ref” command is used for starting a reference movement to a limit or reference switch. The program then continues in parallel to the reference movement. Selection of the limit or reference switch is effected by changing the parameter “Type of reference movement” (see chapter 6.1.2, “Control parameter setting”).
Reference movements are only possible in point-to-point mode.
When the reference movement is started with the “ref” command, the motor moves to the limit or reference switch at the set speed. Subsequently it approaches the reference point in the opposite direction. The speed for this movement is set with the parameter “Clearing speed limit switch”.

```
ref      x1       ;Based on the parameter setting
...               ;"Type of reference movement", a
                  ;reference movement is executed.
ld       m1001    ;Wait until axis x1 has stopped
jmpc     -1
```

reff Reference movement with waiting

Valid in SEQUENCE program

```
reff  <x>
```

The “reff” command is identical with the “ref” command, except for one difference. The SEQUENCE program is stopped until the reference movement is completed.
If an error should occur during the reference movement, further program execution is blocked by the reference movement.

```
reff     x1       ;A reference movement is executed
                  ;and the program waits until
                  ;a reference point is reached.
```

restart Restart SEQUENCE program

Valid in SEQUENCE program

```
restart
```

The “restart” command is used for resetting and reinitializing the controller. The SEQUENCE program and the PLC program restart from line 1. The contents of variables is retained.

```
restart          ;Reset controller.
```

ret Return from subprogram

Valid in SEQUENCE program

```
ret
```

The “ret” command is used for returning from a subprogram; for a programming example, see “cal” command.

s Set output or flag to 1

Valid in SEQUENCE and PLC program

```
s  <qm>
```

The “s” command can be used for resetting an output or flag to 1, based on the current result. The command is only executed if the CR = 1.

```
ld        i1          ;Load input i1 into the CR.
and       i2          ;AND operation with
                      ;inputs i1 and i2.
                      ;CR = i1 and i2.
s         q5          ;Output q5 is set to 1
                      ;if CR = 1.
```

screen Screen control (VT52 terminal)

Valid in SEQUENCE program

```
screen  <c>  <fknvw>
```

The “screen” command is a universal command for screen control on a VT52 terminal or FT2000 terminal (see also chapter 5.4.3, “Miscellaneous test functions”). The operand <fknvw> defines the action.
The control codes generated with this command cannot be interpreted by an ASCII terminal.

<fknvw>	Effect
0	Clear screen
1	Go up by 1 line
2	Go down by 1 line
3	1 character to the right
4	1 character to the left
5	Cursor position: line 1, column 1 (HOME)
6	Delete the line on the left of the cursor
7	Delete the line on the right of the cursor
8	Delete entire line

```
screen    c1    0     ;Clear the screen
screen    c1    8     ;Delete the line
screen    c1    5     ;Line 1, column 1
                      ;(home position)
```

setanalog (Series 300 only) — Output analog voltage

Valid in SEQUENCE program

```
setanalog  <a>  <knvw>
```

The “setanalog” command outputs an analog voltage (mV) on the analog output of the analog module (a2). The voltage is specified in the CR in millivolts (mV).
The second operand <knvw> specifies the channel number of the analog output.

```
ld          5000        ;Output 5 V voltage on
setanalog   a2    1     ;analog output 1 of the
                        ;analog module.
```

setcurrent — Set the motor current

Valid in SEQUENCE program

```
setcurrent  <x>  <fknvw>
```

The “setcurrent” command can be used for setting the motor current for various movement states of the drive (for default settings, see controller manual). The electrical current value is specified as a percentage (0-100%) and refers to the maximum current set on the front panel. The electrical current value is passed to the command in the CR.
You can specify the current for standstill, for the acceleration/deceleration phase and for constant movement. The set electrical current values are effective for the respective movement states.
The second parameter, <fknvw>, defines the movement state to which the electrical current setting applies.

0 = Standstill
1 = Acceleration/deceleration
2 = Constant movement

```
ld           50       ;The current for the
setcurrent   x1   0    ;standstill state is reduced
                        ;to 50% of maximum current.
```

setipos (Series 300 only) — Define linear interpolation

Valid in SEQUENCE program

```
setipos  <x>  <knvw>
```

The “setipos” command can be used in point-to-point mode for defining a linear interpolation process involving two or three axes. For each axis involved in the interpolation process, a target position must be specified in user-defined units.
The number of subsequent “setipos” commands defines a 2-axis or 3-axis interpolation.
The interpolation process is started with a different command (e.g. “linmove” or “linpos”).

The preset position must not be changed during positioning.

Programming example 1:

```
setipos     x1    1000  ;Predefine positions in
setipos     x2    2000  ;user-defined units for
setipos     x3    3000  ;the axes x1 to x3 which
                        ;are involved in the
                        ;interpolation.
linmovef                ;Start the linear
                        ;interpolation to the
                        ;relative target position.
```

Programming example 2:

```
setipos     x1    1000  ;Predefine positions in
setipos     x2    2000  ;user-defined units for
setipos     x3    3000  ;the axes x1 to x3 which
                        ;are involved in the
                        ;interpolation.
linposf                 ;Start the linear
                        ;interpolation to the
                        ;absolute target position.
```

setsiglist (Series 300 only)

Activate position list

Valid in SEQUENCE program

```
setsiglist  <x>  <knvw>
```

The “setsiglist” command activates a position signal list. The output q0 is toggled depending on the position signal list, i.e. it is set to 0 or 1 and vice versa. The position variables of the corresponding axis are used in the position signal list.
The position lists start with w100 (w200, w300, w400), depending on the selected axis (x1 = w100 ..., x2 = w200 ..., x3 = w300 ..., x4 = w400 ...).
The second operand of the command specifies the number of positions to be processed.
If the second operand contains a negative value, the list is processed in reverse order.
When the first postion of the list is passed, the output q0 is set to 1 and the list is processed.
Whenever the axis passes the next position from this list, the signal level on the output q0 changes. If the output is active, it is deactivated, and vice versa.

```
setsiglist    x1    14    ;The position list for
                          ;axis x1 is activated.
movef         x1    1000  ;Start positioning
```

settrigger (Series 300 only)

Activate trigger function

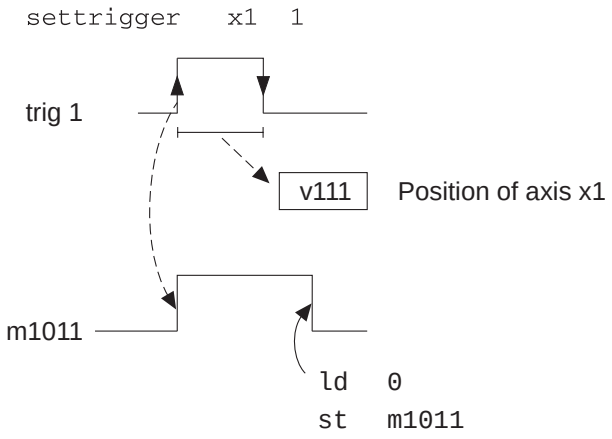
Valid in SEQUENCE program

```
settrigger  <x>  <k>
```

The “settrigger” command activates the trigger input of axis <x>.
The <k> constant is used for selecting whether the trigger input is to respond to a rising (k=1) or a falling (k=0) edge.
The occurrence of a trigger event is registered in the flags m1011, m1012, m1013 and m1014. The position of the axis is stored in the variables v111 to v114 while the trigger signal is active. When the trigger signal deactivates, the position of the axis is kept in the respective variable. The flags must be reset by the user.

Axis	Trigger flag	Trigger position
x1	m1011	v111
x2	m1012	v112
x3	m1013	v113
x4	m1014	v114

```
settrigger    x1    0     ;Trigger input to be
                          ;activated for axis x1
                          ;(rising edge)
```



snd_char — *Output character via the serial interface*

Valid in SEQUENCE program

```
snd_char  <c>  <fknvw>
```

The “snd_char” command outputs a single character via the serial interface <c>. The decimal value contained in the constant <k> or in the variable <n>, <v> or <w> is output as an ASCII character.

Programming example 1:

```
snd_char    c1    65    ;Outputs the character 'A'
                        ;(ASCII decimal value =
                        ;65) via the serial
                        ;interface c1.
```

Programming example 2:

```
ld          66          ;Outputs the character 'B'
st          v10         ;(ASCII decimal value =
snd_char    c1    v10   ;66) via the serial
                        ;interface c1.
```

snd_dez — *Output number with decimal places via the serial interface*

Valid in SEQUENCE program

```
snd_dez  <c>  <fnvw>
```

The “snd_dez” command can be used to output a 32-bit integer number with fictitious decimal places via the serial interface. The number of decimal places is specified in the “Decimal point” control parameter. The number must be stored in a variable (f, v, w, n).

Control parameters:

```
"Decimal point"          2
ld            12         ;Output number in v10
st            v10        ;with 2 decimal
snd_dez       c1    v10  ;places 0.12



ld            1000
st            v10
snd_dez       c1    v10  ;10.00
```

snd_str — *Output a string*

Valid in SEQUENCE program

```
snd_str  <c>  <fknvw>
```

The “snd_str” command outputs a string via the serial interface.
The string must be entered using the “Send Texts” editor. The number of the string to be output is determined by the value of the constant <k> or the variable <n>, <v> or <w>.

Send text:

```
14: Reference movement in progress.
15: Reference movement finished.
```

Programming example 1:

```
snd_str     c1    14    ;Send text 14 is
                        ;output via the serial
                        ;interface c1.
```

Programming example 2:

```
ld          15          ;Send text 15 is
st          v10         ;output via the serial
snd_str     c1    v10   ;interface c1.
```

snd_var — *Output number via the serial interface*

Valid in SEQUENCE program

```
snd_var  <c>  <fnvw>
```

The “snd_var” command outputs the decimal value of the variable <n>, <v> or <w> as a string through the serial interface <c>.

```
ld        x1          ;Read actual position of
st        v20         ;axis x1 and store it in
                      ;the variable v20.
snd_var   c1    v20   ;Output actual position
                      ;as a string via the
                      ;serial interface c1.
```

st — *Store CR in operand*

Valid in SEQUENCE and PLC program

```
st  <fnvwqmxt>
stn  <qm>
```

The “st” command can be used for writing the CR to an operand. The letter “n” can be used for negating the operand.

NOTE
The <t> operand (timer) is only valid in the PLC program.

Programming example 1:

```
ld        i1          ;Load input i1 into the CR.
st        m25         ;Contents of the CR (input
                      ;i1) to be saved as
                      ;flag m25.
```

Programming example 2:

```
ld        i1          ;Load input i1 into the CR.
stn       m25         ;Contents of the CR
                      ;(negated value of
                      ;input i1) to be stored as
                      ;flag m25.
```

st_LBit (Series 300 only)

Write bit to data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
st_LBit   <knvw>    <knvw>
```

The “st_LBit” command is used for writing a bit to the data interface of the Lauer operating panel.
The first parameter, <knvw>, specifies the number of the word into which to write the bit. The data interface range comprises 256 words, i.e. the range of values of the first parameter is from 0 to 255.
The second parameter, <knvw>, specifies the number of the bit within the specified word (16 bits). The range of values of the second parameter is from 0 to 15.

NOTE
The lower section of the data interface is pre-assigned; see documentation on Lauer operating panel.

Programming examples:

```
st_LBit    100, 0     ;Write bit 0 to word 100
st_LBit    100, 15    ;Write bit 15 to word 100
st_LBit    255, 0     ;Write bit 0 to word 255
st_LBit    255, 1     ;Write bit 1 to word 255
```

st_LDint (Series 300 only)

Write double word to data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
st_LDint  <knvw>
```

The “st_LDint” command is used for writing the contents of the CR as a double word to the data interface of the Lauer operating panel.
The <knvw> parameter specifies the number of the double word in the data interface of the Lauer operating panel.
The range of values of the parameter is from 0 to 254. The words 254 and 255 form the last double word in the data interface.

NOTE
The lower section of the data interface is pre-assigned; see documentation on Lauer operating panel.

Programming examples:

```
st_LDint   100        ;Write double word 100
                      ;(word 100 and word 101)
st_LDint   254        ;Write double word 254
                      ;(word 254 and word 255)
```

st_LInt (Series 300 only)

Write word to data interface of Lauer operating panel

Valid in SEQUENCE and PLC program

```
st_LInt   <knvw>
```

The “st_LInt” command is used for writing the contents of the CR as a word to the data interface of the Lauer operating panel.
The <knvw> parameter specifies the number of the word in the data interface of the Lauer operating panel. The data interface comprises 256 words, i.e. the range of values of the parameter is from 0 to 255.

NOTE
The lower section of the data interface is pre-assigned; see documentation on Lauer operating panel.

NOTE
The CR always uses values of DINT type, i.e. “st_LInt” may involve some loss of information.

Programming examples:

```
st_LInt    100        ;Write word 100
st_LInt    120        ;Write word 120
st_LInt    255        ;Write word 255
```

stop

Stopping an axis

Valid in SEQUENCE program

```
stop  <x>
```

The “stop” command can be used for stopping an axis movement or a linear interpolation process. When the “stop” command is input, the drive is deactivated; it can be reactivated with the “pos” command.

```
stop      x1        ;Axis x1 is stopped.
ld        m1001     ;Wait until axis x1 has stopped
jmpc      -1
```

stopa (Series 300 only)

Stopping all axes

Valid in SEQUENCE program

```
stopa
```

The “stopa” command can be used for stopping all axis movements or a linear interpolation process. When the “stopa” command is input, the drives are deactivated; they can be reactivated with the “pos” command.

```
stopa               ;All axis movements or a
                    ;linear interpolation are
                    ;stopped.
ld        m1001     ;Wait until all axes have stopped.
or        m1002
or        m1003
or        m1004
jmpc      -4
```

stimer Load and start timer

Valid in PLC program

```
stimer  <t>
```

Timers are special variables, the values of which change with time. The timers t0 to t9 are defined. The “stimer” command can be used for loading a time value into a timer and activating it. The time resolution is 100 ms (time = time value x 100 ms). A running timer can be stopped by specifying the value 0. A new time value can be input at any time.

```
ld       10       ;Load time value, here 1 s.
st       t1       ;Store time value in timer 1.
stimer   t1       ;Activate timer 1.

ld       t1       ;Read time from timer 1.
eq       0        ;If timer 1 = 0,
st       q1       ;set output 1.
```

sub Subtraction

Valid in SEQUENCE and PLC program

```
sub  <fknvw>
```

The “sub” command can be used for a subtraction involving an operand and the CR.

```
ld       100           ;Load the value 100 into
                       ;the CR.
sub      30            ;From the contents of the
                       ;CR (value 100), subtract
                       ;the value 30.
st       v6            ;Store the result in the
                       ;variable v6.
```

vel Set the set speed

Valid in SEQUENCE program

```
vel  <x>  <fknvw>
```

The “vel” command can be used for programming the set speed of an axis. If a set speed is not programmed in point-to-point mode, the axis positions at the “Standard speed” entered in the parameter editor.
The set speed may be changed before or during a positioning operation. The set speed must always be greater than 0.

```
vel      x1     10000  ;A set speed of 10000 Hz
                       ;is set for the axis x1.
```

wait — *Suspend program execution for some time*

Valid in SEQUENCE program

```
wait  <fknvw>
```

The “wait” command is used for specifying a wait time in the SEQUENCE program. The time resolution is 1 ms. Any initiated axis movements are completed.

```
wait      10      ;The SEQUENCE program is
                  ;stopped for 10 ms.
```

wsave — *Save position variables*

Valid in SEQUENCE program

```
wsave
```

The “wsave” command saves all position variables w100 to w499 into the EEPROM.

NOTE:
On a WDP3-01X controller, the data saved with “wsave” are reloaded into the position variables (w100 to 499) at power-on.

```
wsave             ;Save position variables
                  ;in EEPROM
```

8.2 Sample projects

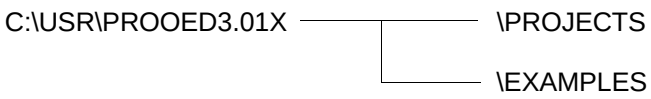
The following section describes, by way of the sample project “MANUAL.PO3”, how to load the supplied sample programs into the controller and execute them.

“MANUAL.PO3” is used for moving the stepping motor manually.

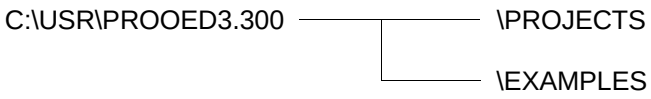
It is not possible to provide generally applicable usage information on the sample programs here. This section merely illustrates the procedure for using the sample programs.

The sample programs are stored in subdirectories of the ProOED3 directory, e.g.:

For WDP3-014/018 controllers with OED3:



For Series 300 controllers with OED3:



Requirements

- The PC must be wired to the controller (see chapter 3).
- The controller and the stepping motor must be wired correctly according to the requirements (see controller manual).
- The PC and the controller must be switched on.
- ProOED3 must have been started.

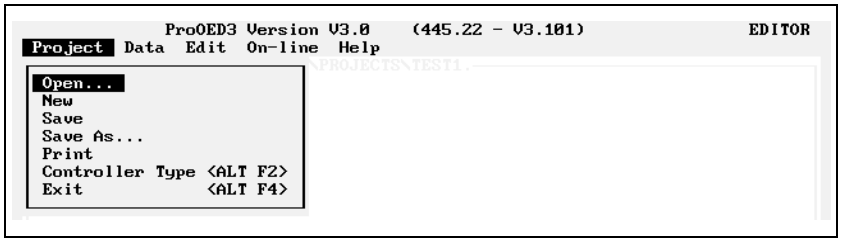
NOTE
For a description of the sample projects, refer to chapter 8.2.3.

8.2.1 Loading the sample project for manual mode

Step 1

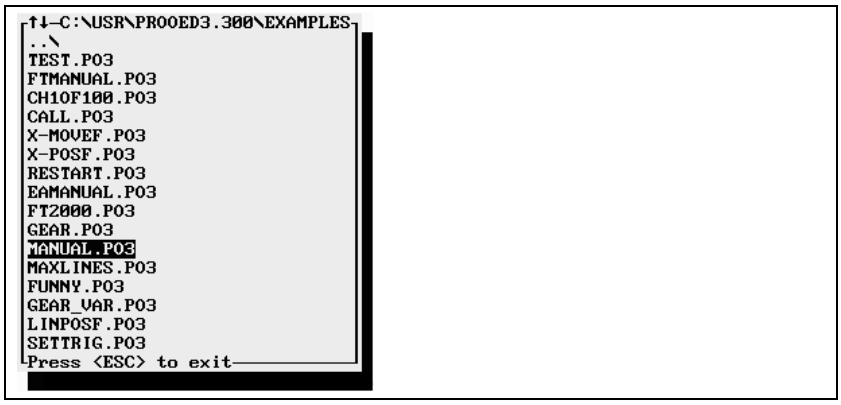
Opening a project

1. Select the menue option “Project/Open”.



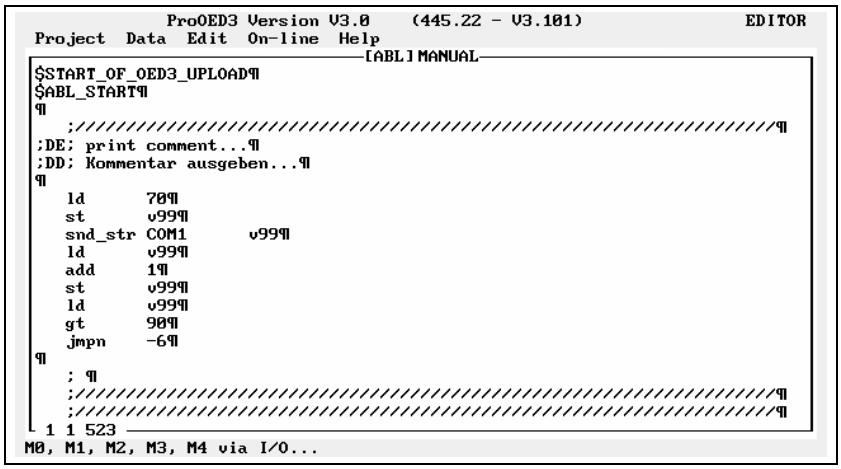
Open project

2. Change to the “...\EXAMPLES” directory.
3. Move the highlight to “MANUAL.PO3” and press the <↵> key.



Select project

A screen which is similar to the one below is displayed.



Project opened

Step 2

Setting the controller type

Select the controller type if required. Otherwise proceed with step 3.

1. Select the menue option “Project/Controller Type”.
2. Use the <↑> and <↓> keys to move the highlight to the controller type connected (e.g. WDP5-318) and press the <↵> key.

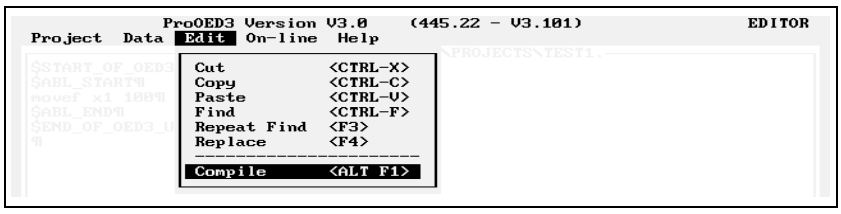
Select controller type

Step 3

Compiling the project

When a project is opened, the SEQUENCE editor is active.

1. Select the menue option “Edit/Compile” or press <Alt>-<F1> to compile the SEQUENCE program. Acknowledge the message which informs you that the program has been compiled successfully by pressing the <↵> key.
2. Select the menue option “Data/PLC Program” or press <Alt>-<2> to change to the PLC editor.
3. Select the menue option “Edit/Compile” or press <Alt>-<F1> to compile the PLC program. Acknowledge the message which informs you that the program has been compiled successfully by pressing the <↵> key.

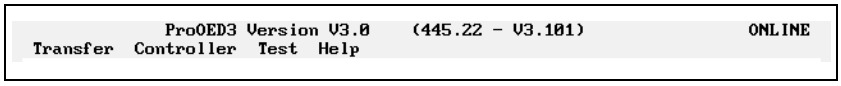


Compile program

Step 4

Loading a program into the controller

1. Press <Alt>-<O> to open the on-line functions.

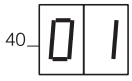


On-line

2. If necessary, set the PC interface using “Transfer/PC interface”.

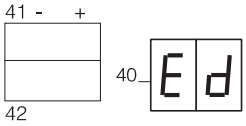
Activating editing mode on the controller

NOTE
The controller must be in editing mode for the PC to be able to communicate with the controller.

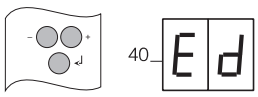


1. Set the controller to STOP status (status display on the controller must show “01”); see controller manual.

2. Select editing mode.



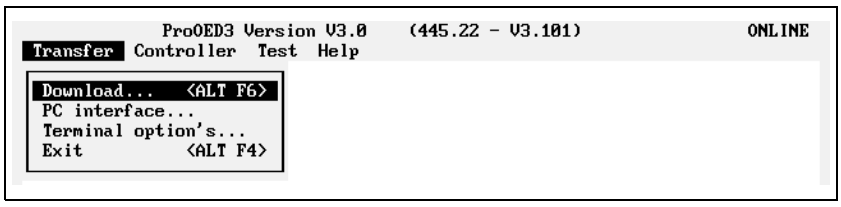
Series 300:
Press and hold key 41 on “+” side.
Press key 42 on “+” side and hold it together with key 41 for three seconds, until “Ed” appears in the controller status display.



WDP3-014/018:
Press and hold the “+” key together with the “↵” key until “Ed” appears in the controller status display.

Loading the project into the controller (download)

1. Select the menue option “Transfer/Download” or press <Alt>-<F6> to open the selection menue for download.

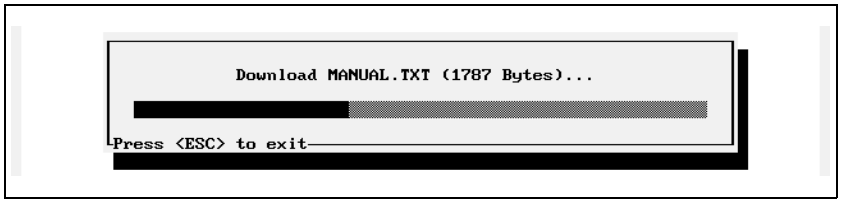


Select download

2. The highlight is on “Complete”.
   => Press the <↵> key.
   “Complete” means that the entire project is loaded into the controller.

Select option

Variables, control parameters, texts, and the control program are transferred to the controller.

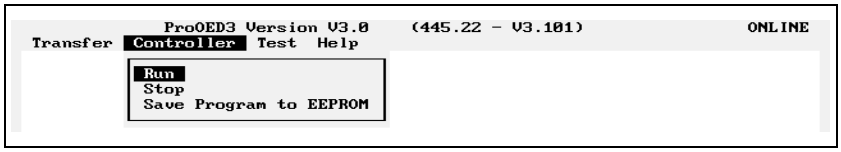


Download

Step 5

Starting the “Manual” control program

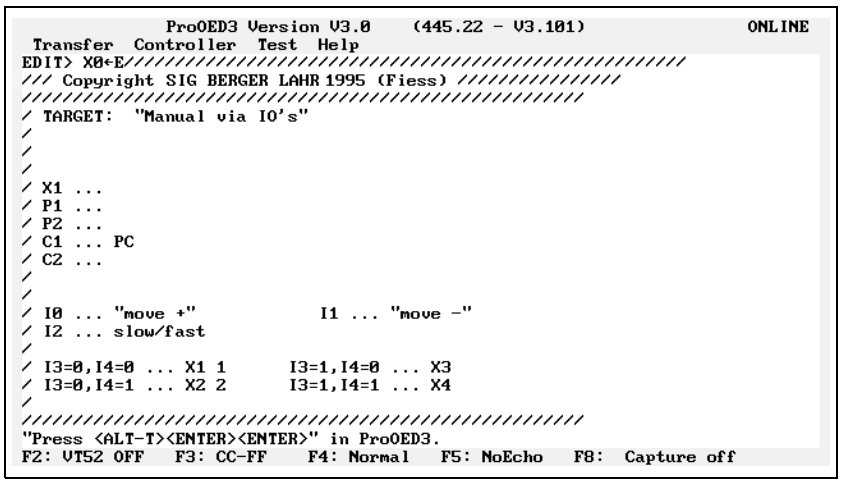
1. Start the controller by selecting “Controller/Run” or by pressing the selector switch 41 or the “+” key on the front panel.



Start program

2. When a sample program is started, information on usage is output on the ProOED3 screen.

Information on usage



According to the information on the screen, moving the stepping motor and selecting the axis for “MANUAL.PO3” is done as follows:

Input	Function
I0	Move the motor to the left
I1	Move the motor to the right
I2	Select slow or fast speed

Select the current axis via I3 and I4:

Axis	Input signal status	
	I3	I4
x1	0	0
x2	0	1
x3	1	0
x4	1	1

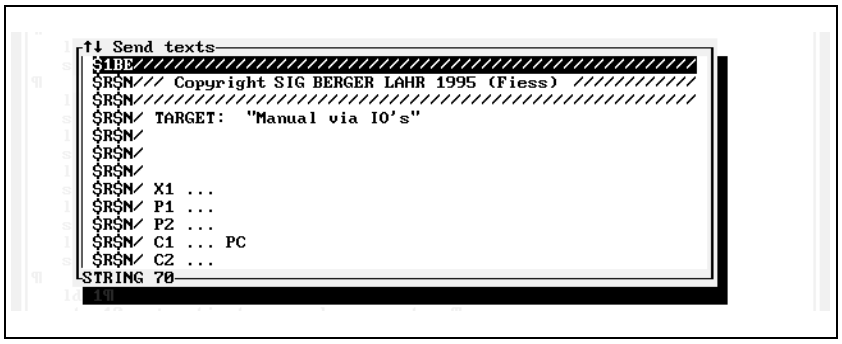
Where to find information on the sample projects before program execution:

Each sample project contains information on usage, inputs used, etc. in the send texts. These send texts are output on the ProOED3 screen after program start.

To view the information before program execution, proceed as follows:

1. Select the menue option “Data/Send Texts”.
2. Move the highlight to string 70 or higher. The information is stored there.

NOTE
The “RN” characters are control codes for carriage return and line feed.



Information in send texts

8.2.2 Copying a sample project to a different directory

Copying projects to different directories is effected on the DOS operating system level.

Example:

To copy the MANUAL project from the EXAMPLE2.XXX directory to the PROJECTS directory and use it there with the name TEST1.

1. Press <Alt>+<F4> to exit ProOED3 and return to the DOS command prompt.
2. Enter

   CD C:\USR\PROOED3.01X\PROJECTS

   or

   CD C:\USR\PROOED3.300\PROJECTS

   and press the <↵> key.

   If the destination directory does not exist yet, create the directory with the command

   MD C:\USR\PROOED3.01X\PROJECTS

   or

   MD C:\USR\PROOED3.300\PROJECTS

3. Enter

   COPY C:\USR\PROOED3.01X\EXAMPLES\MANUAL.*␣TEST.*

   or

   COPY C:\USR\PROOED3.300\EXAMPLES\MANUAL.*␣TEST.*

   and press the <↵> key.

   You can then open and edit the “TEST1” project just as any other project.

8.2.3 Sample project descriptions

Name	Description
CALL	The maximum call depth of subprograms is tested. The inputs I10, I11, LIMP are used to select the specific “cal” command to be executed: I10=1 -> Execute CAL I11=1 -> Execute CALC LIMP=1 -> Execute CALN
CH1OF100	The GETPORT command is used for reading the inputs I0 to I8. The value read is used with the variables w100 to w499 to indirectly determine and approach a position.
EAMANUAL	The inputs I0 to I2 can be used to move the motor via the manual movement flags m0 to m2. In the SEQUENCE program, a dialog with the FT2000 is established. The current motor position is output. In the PLC program, the inputs I0 to I2 are read and written to the manual movement flags (see chapter 6.3.2.7, “Manual movement via the signal inputs” and 6.1.3.3, “Operands”).
FTMANUAL	You can use the function keys to move the motor via the manual movement flags m0 to m2. In the SEQUENCE program, a dialog with the FT2000 is established. The current motor position is output. In the PLC program, the function keys are read and written to the manual movement flags (see chapter 6.1.3.3, “Operands”).
FT2000	Display control commands are issued by means of text output to the FT2000. The following display functions are available: – Clear screen – One line up – One line down – One character to the right – One character to the left – Absolute cursor positioning – Delete character to the right of the cursor – Delete character to the left of the cursor – Delete line
GEAR	Motor movement is effected in position following mode via the encoder. First the gear ratio is set with the “GEARN/GEARZ” commands, then positioning following mode is set. You can then use the inputs to change the gear ratio and execute an absolute positioning operation in position following mode with the “GOFF” command. I0=1 ... “gearz” = “gearz” - 1 I1=1 ... “gearz” = “gearz” + 1 I10=1 ... “goff” = “goff” - 1000 I11=1 ... “goff” = “goff” + 1000
GEAR_VAR	Motor movement is effected in position following mode via the encoder. First the gear ratio is set with the “GEARN/GEARZ” commands, then positioning following mode is set. The motor reference variable is preset by an encoder, the value of which is read in, multiplied with the gear ratio and used as the new setpoint for the motor. The motor moves when the reference encoder value changes You can then use the inputs to change the gear ratio and execute an absolute positioning operation in position following mode with the “GOFF” command. I0=1 ... “gearz” = “gearz” - 1 I1=1 ... “gearz” = “gearz” + 1 I10=1 ... V101 = V101 + 1 I11=1 ... V101 = V101 - 1

Name	Description
MAXLINES	This project has the maximum possible number of lines for the SEQUENCE and the PLC programs. You can test how long a DOWNLOAD to the controller will take with the maximum program size by compiling it.
MANUAL	The inputs I0 to I2 can be used to move the motor via the manual movement flags m0 to m2. In the PLC program, the inputs I0 to I2 are read and written to the manual movement flags (see chapter 6.3.2.7, “Manual movement via the signal inputs” and 6.1.3.3, “Operands”).
RESTART	The “RESTART” command in the SEQUENCE program stops the controller, reinitializes and restarts it. The “RESTART” command has the same effect as stopping and restarting the controller via the front panel.
TEACHIN	The inputs I0 to I2 can be used to move the motor via the manual movement flags m0 to m2. In the PLC program, the inputs I0 to I2 are read and written to the manual movement flags (see chapter 6.3.2.7, “Manual movement via the signal inputs” and 6.1.3.3, “Operands”).
TIMER	Various time variables are programmed in the PLC program with the STIMER command. The timer variable T0 is used as the basic value, T1=2*T0, T2=4*T0, T3=8*T0, etc. The current values of these variables are displayed in the SEQUENCE program.
VT52EMUL	Display control commands are issued by means of text output on the screen in VT52 mode. You can use the “CURSOR” and “SCREEN” commands for this. The following display functions are available: – Clear screen – One line up – One line down – One character to the right – One character to the left – Absolute cursor positioning – Delete character to the right of the cursor – Delete character to the left of the cursor – Delete line
X-LINPOS	The “LINPOSF” command is used for motor positioning. A reference movement is executed with the “REFF” command before each positioning operation. The input I0 is used to execute the command displayed on the screen.
X-MOVEF	The “MOVEF” command is used for motor positioning. A reference movement is executed with the “REFF” command before each positioning operation. The input I0 is used to execute the command displayed on the screen.
X-POSF	The “POSF” command is used for motor positioning. A reference movement is executed with the “REFF” command before each positioning operation. The input I0 is used to execute the command displayed on the screen.

8.3 Glossary

Accumulator
A register in the controller which is used for storing intermediate results. The contents of the accumulator is also referred to as the current result (CR).

Arithmetic operation
The “add”, “sub”, “mul” and “div” commands can be used for arithmetic operations involving an operand and the CR.

Current result (CR)
The current result (CR) is a temporary storage element (accumulator) on the controller which is used for arithmetic and logical operations and for data transfer. In contrast to conventional programmable logic controllers, BERGER LAHR Series 300 controllers only have one accumulator for the CR. The memory size of the accumulator is automatically adjusted to the data type of the operand.

Drive unit
Drive units are processing parameters internal to the controller, which are used for positions, speed and acceleration values.

Electronic gear
In position following mode, a gear ratio can be applied to the reference variable (e.g. encoder) to implement step-up or step-down gearing. This is also referred to as an electronic gear. The following applies:

Drive units = Reference variable x Gear ratio

Encoder
Sensor for motor position detection (actual position detection).

Encoder evaluation
This parameter is used for setting the encoder resolution (500 or 1000 marks) and the internal evaluation factor (single, double or quadruple). The following applies:

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

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. The controller has a certain memory area for flags.

Input/output
The controller is provided with a certain number of inputs and outputs through which sequential operations are controlled. Inputs and outputs can be processed simultaneously with the execution of movements.

Interpolation

Simultaneous coordinated movements of several axes (at least two).

Logical operation

The “and” and “or” commands can be used for logical operations involving two or more Boolean operands and the CR.

Normalizing factor

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

Operand

Many instructions require an operand with which the operation is performed.

Point-to-point mode

In point-to-point mode, a positioning command is used for moving from point A to point B. Positioning can be effected with absolute values (relative to the reference point of the axis) or with incremental values (relative to the current position of the axis).

Position following mode

In position following mode, positions are preset via an encoder input or a variable.

A gear ratio can be applied to a position, which enables you to implement an electronic gear.

Process image (PI)

In the process image, the inputs/outputs are temporarily stored.

Reference movement

The “ref” and “reff” commands can be used for performing reference movements. Reference movements are only possible in point-to-point mode. When executing a reference movement, a reference point is approached which is to be defined as the zero point for the system of dimensions.

Relational operations

The “eq”, “gt” and “lt” commands can be used for a relational operation involving an operand and the CR. After the operation, the CR has always the data type BOOLE and contains any of the following values:

0 or FALSE, if the result is not true or

1 or TRUE, if the result is true.

Setpoint (set position)

In point-to-point mode, setpoints are preset with the “pos(f)” or “move(f)” commands and a positioning operation is initiated.

Setting and resetting

The set (“s”) and reset (“r”) commands can be used for setting a variable of type BOOLE (e.g. output or flag) to 1 or resetting it to 0, depending on the current result.

Transfer operation

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

User-defined unit

User-defined units are processing parameters which can be freely defined by the user. The following applies:
Drive units = User-defined units x Normalizing factor
(drive units are given in steps or increments, user-defined units may be cm, for example).
Positions, speed and acceleration values are always specified in user-defined units.

Variable

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

8.4 Abbreviations

AC	Alternating current
add	Addition command
and	AND operation
ASCII	American Standard Code for Information Interchange
b	Boolean value
B	Byte
c1	Serial interface 1
c2	Serial interface 2
cal	Subprogram call
CR	Current result
DC	Direct current
DG	Encoder
div	Division command
Doc. no.	Documentation number

E	Encoder
eq	Relational command EQUAL TO
f	FRAM variable
gt	Relational command GREATER THAN
Hz	Hertz
i	Input
I/O	Input/output
jmp	Jump to a label
k	Value

L	Label
ld	Load
LIMN	CCW limit switch
LIMP	CW limit switch
lt	Relational command LESS THAN
m	Flag
M	Motor
ms	Milliseconds
mul	Multiplication command
N	Number of encoder marks
or	OR operation

p	Encoder
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 a subprogram
s	Set
st	Store
Stop	Stop signal
sub	Subtraction command

t	Timer
TRIG	Trigger signal
v	Variable
w	Position variable
x	Axis number

9 Index

R

S

10 Corrections and additions

At present there are no corrections or additions.