Operating instructions

PAC-SC Compact

231513

as of Version 3.1

Edition 19/94

IEF Werner GmbH, Wendelhofstr. 6, D-78120 Furtwangen
Tel.: 07723/9250, FAX: 07723/925100

We are always grateful for any suggestions for improvements and information on errors.

Contents

1 Technical data in brief

Contents

1.1 System concept

The PAC-SC Compact is a positioning and sequence controller.

The unit is suitable for installation in control cabinets or for attachment to mounting plates.

The power output stage for a 2-phase stepping motor and the control electronics for the positioning and sequence controller are integrated in the housing.

Program compilation, commissioning and diagnosis take place via the existing serial interface using the program package PROPAC (which can run on a PC under MS-DOS) from a PC.
The PAC-SC Compact is part of the PA-Control range and therefore has the same instruction set (referred to 1 axis) as the other units in this range. This brief information contains an overview of the instruction set.

Performance features:

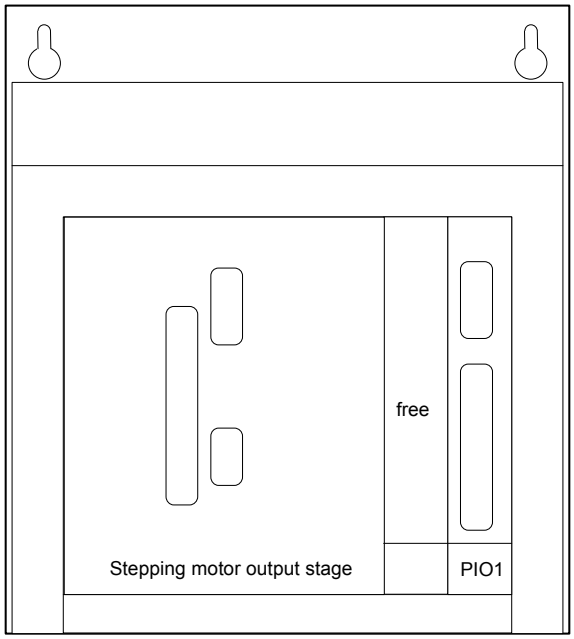
- a 2-phase stepping motor output stage with rotation monitoring
- Traverse frequency ("Traversing speed") up to 50 kHz
- 12 optodecoupled inputs, including 2 limit switch inputs (extendible by 16)
- 8 optodecoupled outputs (extendible by 16)
- a serial interface (RS232)

1.2 Technical data

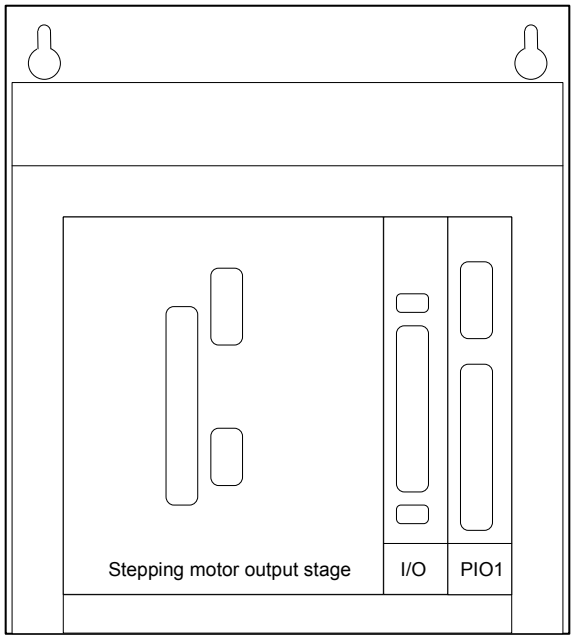
1.2.1 General

Ambient temperature	0 ƒC to 40 ƒC
Number of axis (F_{max} = 50 Hz)	1
2 Limit switch inputs	opto-decoupled 24 VDC typical input current 5 mA Low level 0 - 3 V DC High level 10 - 30 V DC
Stepping motor output stage	2 Phases 12A / 140 VDC Steps per rotation: 200, 400, 500, 800, 1000
Rotation monitoring	Yes
10 inputs (capable of extension to 16) opto-decoupled	24 V DC typical input current 5 mA Low level 0 - 3 V DC High level 10 - 30 V DC
16 outputs (capable of extension to 16) opto-decoupled	positive-switching 24 V DC/0.5 A (ohmic load) max. 2 A per card
1 serial Interface	RS232
Application memory	40 kB (not open-ended)
Supply voltage:	CPU: 24 VDC +/- 15%, 10 VA Output stage: 30-90 VAC, max. 1000 VA Break: 24 VDC

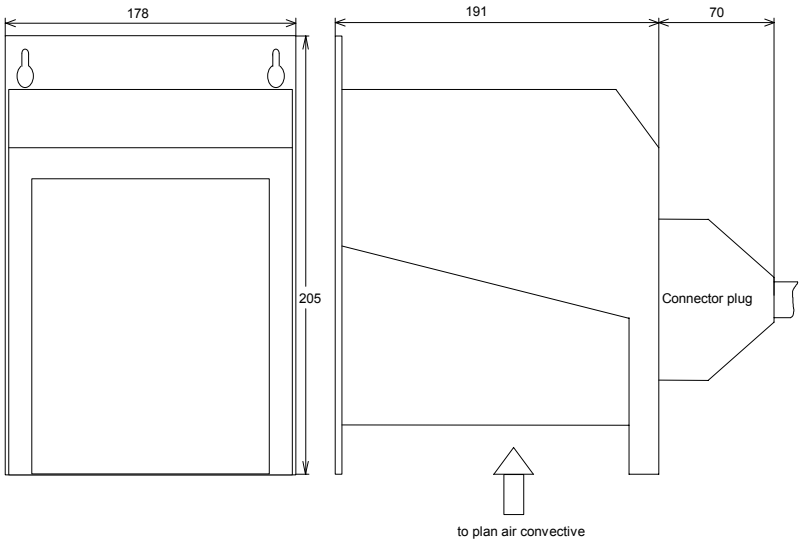
1.2.2 Standard equipment

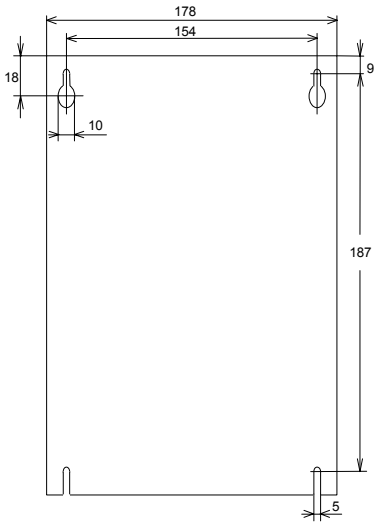


1.2.3 Example of equipment



1.3 Outer dimensions





2 Operator interface

Contents

Important:

In the case of units with a 7-segment status display and also the PA-CONTROL-SC, the front panel can be simulated on the PC with the aid of the program development system PROPAC. Communication with the controller is via a serial interface. This gives the same operating possibilities as with units which have an alphanumeric keyboard and an LCD display.

2.1 The Keyboard

The PA-CONTROL has a full-complement, alphanumeric keyboard (0-9, A-Z). Certain special keys whose functions are described below are also provided.

Special keys:	Name :	Function :
ESC	ESC key	Quit sub-menu, Abort in input field
ALT	ALT key	No function assigned as yet
CTRL	CTRL key	No function assigned as yet
SHIFT	SHIFT key	Calling dual assignment of aphanumeric keys, additional key for cursor keys
↓	Arrow down key	Paging in the menu to the next line (with Autorepeat)
↑	Arrow up key	Paging in the menu to the previous line (with Autorepeat)
→		Move cursor one character to the right in the input feld (with Autorepeat)
←		Move cursor one character to the left in the input feld (with Autorepeat)
	Space bar	Insert blanks in the editor (when programming)
ENTER	ENTER key	Terminate Enter ,Terminate line in the editor
INS	INS key	No funktion assigned as yet
DEL	DEL key	Deletes the character at wich the cursor is positioned
START	START key	Starts Automatic mode
STOP	STOP key	Stops Automatic mode
Automatik | Programm	KEY-operated switch	Access interlock for front panel (the key can be withdrawn in the position shown and then only „Automatic“ is possible).

2.2 General operation

Note:

Differences in menu structure may arise due to the control equipment or the different types of devices.

The basic menu is shown on the display after the PA-CONTROL is switched on.

PA-CONTROL Vxx.xx
1 = Automatic

The operator interface of the PA-CONTROL is subdivided into a menu structure. The main menu and the various sub-menus are structured and can be operated on the basis of the same principle.

Menu principle:

Menu title
1 = First sub-menu option

2 = Second sub-menu option
3 = Third sub-menu option
...
...
9 = Ninth sub-menu option

The number of menu sub-options in the individual menus has been matched to the requirements and limited to 9.
Since only two lines are available on the display, only the title and one of the menu sub-options are displayed on the current menu.

Moving through the menu interface:

Key	Action / Effect
Arrow down	Next menu sub-option is displayed
Arrow up	Previous menu sub-option is displayed
SHIFT + Arrow up	First menu sub-option is displayed
SHIFT + Arrow down	Last menu sub-option is displayed
ENTER	Activating the displayed menu sub-option
ESC	Quitting sub-menu and returning to the previous menu (the first menu line is displayed in the case of the main menu)

One other method of activating a menu option is to press the key with the number prefixed to the menu option.

2.2.1 Input field

If the operator is requested to enter a numerical value (parameter value, register value, ...), an input field is shown on the display.
The input field is delimited by brackets [_.....]. The current value and the cursor (blinking field on the display) are located between these two brackets. The operator can update the entry by pressing the corresponding keys.

Key assignment in the input field:

Key	Response / Effect
Arrow left	Moves cursor one character to the left
Arrow right	Moves cursor one character to the right
SHIFT + DEL	Deletes character to the left of the cursor
DEL	Deletes character at which the cursor is positioned
ENTER	Terminating an entry. The entered value is checked and, if it lies within the permitted limits, is accepted. If the actual value lies above or below the permitted limits, an error message is displayed and the system awaits entry of a new value.
ESC	Aborting the entry. The previous value is retained

2.2.2 Entering the program name

If the control awaits entry of a program name, you will see the following on the display.

```
Please enter new program name!
[ _      ]
```

The cursor (blinking field on the display) is positioned at the first position of the input field. The operator must now press the keys assigned to the letters of the required program name. Complete the entry by pressing key "ENTER" after you have entered all letters of the program name. The program name entered must comply with certain conditions (see below) and is then checked for this. The following message is displayed if the conditions are not fulfilled:

```
ERROR: Illegal program name!
Program: 5ANTON
```

You can acknowledge the error message by pressing a key and then correct the error in the input field.

The program name must comply with the following conditions:

- the first character must be a letter
- blanks are not permitted within the name
- the length is limited to 8 characters
- only "_" and "-" are practical for use as special characters
- upper-case letters

Moving the cursor in the input field:

Key	Response / Effect
Arrow left	Moves cursor one character to the left
Arrow right	Moves cursor one character to the right
SHIFT + DEL	Deletes character to the left of the cursor
DEL	Deletes character at which the cursor is located
ENTER	Terminates an entry

2.2.3 Selecting a name from the program list

If you wish to access an existing program (e.g. to modify the program or to select it and define it as a start program etc.), you are offered an alphabetically sorted list of the existing programs allowing for the possible program types.

```
Program list, please select!      [      ]
Program: 1 EXAMPLE.PNC
```

There are two selection options:

1. You can page in the list with keys "Arrow up" and "Arrow down" until the required program is displayed. You can then complete selection by pressing key "ENTER" and the program is then accepted for the action.

2. A blank input field ([ ]) is displayed at the end of the first display line. You can enter the name of the required program in this input field using the keyboard. The characters entered on the keyboard are accepted only if a program name with the same character string exists. The display (second display line) is corrected in accordance with the input field after each character entered.

```
Program list, please select!      [E     ]
Program: 5 EXAMPLE.PNC
```

You terminate selection by pressing key "ENTER", and the displayed program is then accepted from the second display line for the action.

You can delete the contents of the input field by pressing keys "Arrow down" or "Arrow up".

2.3 Menu structure PA-CONTROL

main-menu	sub-menu
1 = Automatic	- Start - Stop
2 = Manuel	1 = Approach reference point 2 = Axes manual via front panel
3 = Edit program	1 = Display directory 2 = Edit new program 3 = Edit existing program 4 = Copy program 5 = Rename program 6 = Delete program 7 = Print program 8 = Show memory 9 = Transfer program *
4 = Program test and diagnosis	1 = Program test 2 = Inputs 3 = Outputs 4 = Flags (Marker) 5 = Real number register (R) 6 = Integer number register (N) 7 = Closed loop test 8 = Limit switch / Standby 9 = Counter **
5 = Sequence definition	1 = Start program 2 = Program after Stop 3 = Program Start after Stop 4 = Program afterFault 5 = Labelling 6 = Program protection
6 = Parameter	1 = Edit system parameter 2 = Edit axis parameter 3 = Edit ASI-BUS
7 = Basic settings	1 = Bootload system parameters 2 = Bootload axis parameters 3 = Clear program memory 4 = Re-initialize PA-CONTROL
8 = System diagnosis	1 = Display PA-CONTROL-type 2 = Keyboard test 3 = Stop key test 4 = Start key test 5 = Key switch test 6 = Test serial interface 1 7 = Test serial interface 2 8 = Interbus-S monitor ***
9 = Communication with Modem	1 = Communication with Modem 2 = Edit Modem settings 3 = Reset Modem settings

* only with equipment PAB (option)
** only with equipment Counter board (option, only by PAC, PAC-Compact, PAC-Servo)
*** only with equipment Interbus-S board (option)

2.4 Automatic

In Automatic mode, the program defined as the start program is run. Other programs can be called as subroutines. A running program can be interrupted with "STOP". The program is started or an interrupted program can be continued with "START". The required input is assigned via the parameter level for functions external START and STOP (see Chapter Parameters).

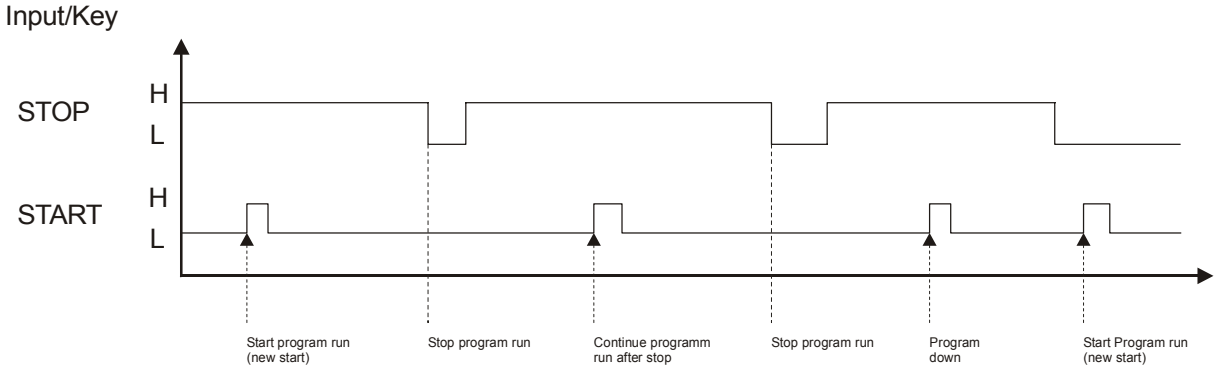


BILD049E

2.5 Manual

Overview, Manual menu

You are offered the following selection after you change to this menu option.

Manual **1 = Approach reference point**

2 = Axes manually via front panel

2.5.1 Reference travel

You are offered the following selection after you change to this menu option.

Reference travel **1 = X**

2 = Y
3 = Z
4 = U

The number of axes available for selection is dependent upon the equipment complement of the PA-CONTROL (1, 2, 4 or 8). You will see the following display after you select the axis.

Press Start

You can now start reference travel by pressing the "Start key", or return to the Manual menu by pressing key "ESC". The following display is shown during reference travel.

Axis running

Reference travel can be interrupted at any time with the "Stop key" and then aborted with key "ESC" or continued with the "Start key".

2.5.2 Axes manually via front panel

The absolute positions of the axes are displayed on the display after you change to this menu option.

X : 0.000 Y : 0.000

You select manual traverse with the corresponding axis by pressing the key with the axis name (X, Y, ...). The following display is then shown (example for the X axis).

X : 0.000 V = 500.000 **1 = 300.000 2 = 500.000 3 = 1000.000**

The following assignment applies to the above display:

1st line:

X: 0.000	:	Current absolute position of the axis
V = 500.000	:	Currently selected traversing speed

2nd line:

1 = 300.000	:	Traverse speed; this is taken from parameter Creep speed
2 = 5000.000	:	Traverse speed; this is taken from parameter Manual speed
3 = 1000.000	:	Traverse speed; this is taken from parameter Reference speed

The following key assignment applies in this state:

Key	Response / Effect
Arrow left	Traverse in negative direction (press briefly: Single-step, press for longer period: Continuous)
Arrow right:	Traverse in positive direction (press briefly: Single-step, press for longer period: Continuous)
1	Set the current traverse speed to the value after 1 = ...
2	Set the current traverse speed to the value after 2 = ...
3	Set the current traverse speed to the value after 3 = ...
ESC	End, return to preceding menu

2.6 Programming

Overview, Programming menu

You are offered the following selection after you change to this menu option

Programming **1 = Directory**

2 = Edit new program
3 = Edit existing program
4 = Copy program
5 = Rename program
6 = Delete program
7 = Print program
8 = Show memory
9 = Transfer program*

Please note:

The user memory of the PA-CONTROL is limited due to the size of the RAM chip on the CPU3 also on the PIO-x. User memory still free can be displayed in menu option 8.

After you activate a menu sub-option of the Programming menu, the PA-CONTROL checks whether the user memory still free suffices for this action. If this is not the case, the activated function is not executed and it is aborted with the following message.

Insufficient memory

You can return to the menu by pressing any key.

If, for example, you nevertheless wish to create new programs, you must increase the size of the free user memory.

Options:

- Deleting programs no longer required
- Replace RAM module on CPU3 with a bigger one (contact IEF Werner beforehand)
- When equipped with PIO-x no extension is possible.
- Deleting less important comments from the programs

* only with aktiv PAB (option)

2.6.1 Display program directory

In this menu option, you can page through the current program directory. You will see the following display:

No.	Name	Type	Stat	Characters	Lines
1	EXAMPLE	PNC	0	13	3

The following assignment applies:

No. : Consecutive numbering of the programs

Name : Program name

Type : Programme type (PN = NC - Programme, PTX = Text programme, PAB = PAB - Programme, when PAB is available)

Status : Programme status (0 = write/read access possible, 1 = write-protected, 2 = transverse sum error, 3 = PAB Programme available in HOST, but not in PAB)

Characters : Number of Ascii characters in the program

Lines : Number of program lines

By simultaneously pressing keys "CTRL" and "DEL", you can activate the delete operation for the displayed program.
Before the program is deleted, you must confirm the delete operation again.

Program: EXAMPLE PNC
Confirm delete program? 1 = yes

The program is deleted if you press key "1". Pressing any other keys aborts the delete operation and the program remains unchanged in the user memory.

2.6.2 Create new program

If you activate this menu option, you are prompted to enter a new program name (see description, entering program name).

You must then select the program type.

Program type list, please select!
1 = PNC

2 = PTX
3 = PAB

PNC program : Executable NC program with positioning and I/O instructions

PTX program : Text program for labeling inputs, outputs, ...
(Refer to the Sections "Program test and diagnosis" and "Sequence definitions" for further details.)

PAB program : Executable program for parallel processing (option)

After you make the selection, you will then be in the PA-CONTROL program editor in the 1st program line.
Please refer to Chapter Instructions of the PA-CONTROL family - Programming information for a description of the program editor.

Note:

The once defined program typ can not later changed in the PAC

2.6.3 Modify existing program:

When you activate this menu option, you are offered a list of the existing programs, sorted alphabetically. You can page through the list and access the required program for programming by pressing key "ENTER".

```
Program list; please select!      [      ]
Program: 1 EXAMPLE.PNC
```

After you make the selection, you will be in the PA-CONTROL program editor. The first two lines of the selected program are displayed.
See Chapter Instructions of the PA-CONTROL family - Programming information for a description of the program editor.

2.6.4 Copy program

With this menu option, you can copy the contents of an existing program to a new program.
When you activate this menu option, you are offered a list of the existing programs, sorted alphabetically. You can page through the list and select the required program.

```
Program list, please select!      [      ]
Program: 1 EXAMPLE.PNC
```

After you make the selection, you are prompted to enter a new program name. If the entry is acceptable, the program is copied and is included in the directory as a further program.

Important:

The program type cannot be changed during the copy operation.

2.6.5 Rename program

In this menu option, you can change the program name of an existing program.
When you activate this menu option, you are offered a list of the existing programs, sorted alphabetically. You can page through the list and select the required program.

```
Program list, please select!        [      ]
Program: 1 EXAMPLE.PNC
```

After you make the selection, you are prompted to enter a new program name. After you make the entry, the program is then listed unchanged with a new name in the directory.

2.6.6 Delete program

In this menu option, you can delete an existing program (remove it from the program memory).
When you activate this menu option, you are offered a list of the existing programs, sorted alphabetically. You can page through the list and select the required program.

```
Program list, please select!        [      ]
Program: 1 EXAMPLE.PNC
```

After you make the selection, you are prompted once again to confirm the selection and your wish to delete this program.

```
Program: EXAMPLE PNC
Do you really want to delete program? 1 = yes
```

The program is deleted if you press key "1". Pressing any other keys aborts this selection and the program is then not deleted.

2.6.7 Print program

In this menu item the operator can create a programme printout on a serial printer connected to the serial interface 1or 2 (the PA-CONTROL -SC and the PA-CONTROL Single only have 1 interface each).

The program is transferred (interface number, baud rate, format) under the conditions set in the system parameters (see Chapter Parameters).

The pages of the program printout are consecutively numbered and each page is provided with a header, consisting of the program name, program type, comment field and page number.

When you activate this menu option, you are offered a list of the existing programs for selection, sorted alphabetically.

```
Program list, please select!        [      ]
Program: 1 EXAMPLE.PNC
```

After you make the selection, you are prompted once again to confirm the selection and your wish to print this program.

```
Program: EXAMPLE PNC
Print? 1 = yes
```

You can confirm the selection by pressing key "1". Pressing any other keys aborts this selection and returns you to the menu.

You will then see the following display after you confirm the selection.

```
Enter comment on printout
[                                          ]
```

The second display line has an input field with a length of 38 characters. The operator can enter a text at this point, and this text is then printed out in the program printout in each header as the comment field. You can complete the entry and start the print operation by pressing key "ENTER".

Important:

You can abort the print operation by pressing key "ESC".

2.6.8 Display program memory occupancy

In this menu point, the operator can show how much of the main memory is already occupied by the existing programs or is still free.
There are two different displays for this menu point for the equipment variants with and without PAB (optional).

On units without PAB (optional):

The following display, for example, appears on units without PAB after this menu point is activated:

Memory occupied:	**85**
Memory free:	**21915**

On units with PAB (optional):

The following display, for example, appears on units with PAB after this menu point is activated:

Memory occupied.	**2325**	**1269**
Memory free	**17315**	**22395**

The current figures for the memory of the host CPU are displayed on the left side (in this example: 2325, 17315) in this display. The figures on the right side (in this example: 1269, 22395) show how much memory space on the PAB is still free or is occupied.

Please note:

The value for "memory space free" differs depending on the development level and features!

2.6.9 Storing/deleting programs in a PAB

Please note:

Only active in conjunction with a PAB (optional).

The following display appears after transfer into this menu subpoint:

No.:	Name	Type	Characters	in PAB
1:	A-Test	PAB	703	yes

The name, type and number of characters of the program are shown. It is also shown whether the program is currently under PAB (in PAB yes) or is not present (in PAB no).
The user can scroll through the program directory with the "Arrow up" and "Arrow down" keys. If one of the programs on the PAB is to be deleted or stored, this can be done with the help of the keys "DEL" (delete) and "INS" (store) when the appropriate program is shown on the display.

This menu point can be exited with the aid of the ESC key.

Please note:

In this menu point, the programs are only deleted or stored on the PAB.
The memory of the host CPU is not affected by this.

2.7 Program test and diagnosis

Overview, Progra test and diagnosis menu

You are offered the following selection after you change to this menu option:

Program test and diagnosis **1 = Program test**

2 = Inputs
3 = Outputs
4 = Flags (Marker)
5 = Real number register (R)
6 = Integer number register (N)
7 = Save current states
8 = Limit switches / Standby
9 = Counter*

2.7.1 Program test

The user is offered the following selection after transfer into this menu point:

Program test **1 = Automatic with serial diagnosis**

2 = Automatic (positioning only)
3 = Automatic, individual step

Automatic with serial diagnosis:

The serial diagnosis is set via this menu item. The diagnosis can be activated and/or deactivated using the ENTER key. The current setting is shown on the display. The diagnosis remains active in automatic operation until it is switched off in the same manner.
If the diagnosis has been activated a serial diagnosis can be performed at all times with PROPAC. Please refer to the operating instructions of the programme development system PROPAC for a more detailed description.

Automatic (positioning only):

Not implemented in this version!

Automatic, individual step:

Not implemented in this version!

* Only active in conjunction with a counter board (optional).

2.7.2 Inputs, outputs, flags:

This menu sub-option is used to manually check and change the states of the inputs, outputs or flags. You will see the following display after you activate these sub-options:

e.g.: for inputs

I 5:	**Label text**
I 1 ->	**0000<u>0</u>0000000000**

The following assignment applies:

1st line: The number after "INPUT" is the input at which the cursor field is located. The text from the program assigned in menu option "Sequence definitions -> Text assignment -> 1 = Inputs" is located at the "Label text" position. No text is displayed if no assignment has been made.

2nd line: "I 1 ->" means that, as of input 1, the states of the next 16 inputs are displayed in this line. These are followed by the logic states of the inputs (0 or 1).

0: Input not energized, output or flag reset

1: Input energized, output or flag set.

Moving and actions in the display:

Key	Response / Effect
Arrow right	Move cursor field one position to the right (to the next element)
Arrow left	Move cursor field one position to the left (to the previous element)
SHIFT + Arrow right	Move cursor to the last cursor field of the row (last element)
SHIFT + Arrow left	Move cursor to the first cursor field of the row (first element)
Arrow down	Display next group (e.g. input 17-32)
Arrow up	Display previous group (e.g. input 1-16)
SHIFT + Arrow down	Display last group (e.g. input 497-512)
SHIFT + Arrow up	Display first group (e.g. input 1-16)
ENTER	Select element directly, enter the required element in the input field and display with ENTER
1	Set output or flag at which the cursor field is located
0	Reset the output or flag at which the cursor field is located
ESC	Quit sub-option

The following display is shown for inputs 17-32:

```
I 17: Label text
I 17 ->      0000000000000000
```

The following display is shown for outputs 1-16:

```
O 1:         Label text
O 1 ->       0000000000000000
```

The following display is shown for flags 1-16:

```
M 1: Label text
M 1 ->       0000000000000000
```

Note:

The output and flag statuses are retained when the menu is exited!
The number of inputs and outputs can vary according to the control equipment!

2.7.3 Real number register (R), integer register (N)

These menu sub-options serve to manually check and change the contents of the real number and integer registers. You will see the following display after you activate these sub-options:

e.g. for real number register:

```
R 1: Label text
R 1  = 0.000
```

The following assignment applies:

1st line:	The number after "REAL NUMBER REGISTER" is the number of the register displayed. The text from the program assigned in menu option "Sequence definitions -> Text assignment -> 4 = Real number register" is located at the "Label text" position
2nd line:	"R 1 = 0.000" is the contents of the displayed register (in this case, register 1 with value 0).

Moving and actions in the display:

Key	Response / Effect
Arrow down	Display next register
Arrow up	Display previous register
SHIFT+Arrow down	Last register
SHIFT+Arrow up	Display first register
ESC	Quit sub-option
Numeric key (1..9-+)	Activating the input field and the character is accepted at the first position of the displayed value
Arrow right	Activation of the input field
ENTER	Terminate entry
ENTER	Select element directly, enter the required element in the input field and display with ENTER
ESC	Abort entry and previous value is retained

See description "Operator interface, - Input field" for the key assignment in the input field.

The entered value is transferred to the register only if the number lies within the defined value limits (+/- 8.000.000).

Example for integer register:

```
N 1: Label text
N 1 = 0
```

2.7.4 Positioning control test

Note:

This submenu point is implemented only in control units with positioning control (PA-CONTROL Servo).

After choicing this submenu point there has to be an axis selection first. The operator must point out for wich axes the test should be realize. An axis selection is available (only the first line are shown):

```
Positioning control test axis selection
1 = X
```

If one axis is choosen the following test selection will be displayed:

```
Test selection
1 = positional display
```

2 = positioning control inputs
3 = release output
4 = DA-converter test

Positional display: The position of the selected axis will be displayed. The display occurs in dependence on the gear factor and also with the choosen system of units. The release output is switched off.

Positioning control inputs:

```
-X   +X    ref.   standby
0    0     0      0
```

In this submenu point the limit switch (-X, +X), the additional reference position (ref.) and standby of the output stage (standby) are displayed.

Release output:

```
X    release output, change, end = ESC
actual state 0
```

DA-converter test:

By selection of this point another choice of submenu points is displayed.

DA-converter test **1 = DA-zero**

2 = Da-variable
3 = DA-sow tooth
4 = DA-rectangle

DA-zero:

X: zero volt output **end with ESC**

DA-variable:

X: variable voltage output **end with ESC, actual state [0]**

input range: +/- 10 V, in this case „1“ is the same as 1V

DA-sow-tooth:

X: sow-tooth +/- 10V output **end with ESC**

DA-rectangle:

x: rectangle +/- 1V output **end with ESC**

„X“ at the beginning of the first line means that the X-axis was choosen at the beginning of the menu point positioning control test (other axis are possible).

2.7.5 Limit switches / Standby

This menu sub-option serves to check the limit switches of the individual axes and standby status of the motor power sections. The displays differ in this sub-option, dependent upon configuration of the PA-CONTROL, since only the existing motor axes are allowed for. You will see the following display after you activate the sub-option:

e.g.: PA-CONTROL Compact

```
B X+- Y+-
1   10  01
```

The following assignment applies:

The elements Standby (B) and the limit switches of the axes (X+-, Y+-, ...) are displayed in the first line. The logical state of the corresponding inputs is displayed in the second line.

0	:	Power section not ready, limit switch (NC contact) actuated
1	:	Power section ready, limit switch not actuated

The above display thus has the following significance:

- Power sections X- and Y- axis ready
- Limit switch X+ not actuated (input energized)
- Limit switch X- actuated (input not energized)
- Limit switch Y+ actuated (input not energized)
- Limit switch Y- not actuated (input energized)

Example PA-CONTROL (8 axes)

```
B X+- Y+- Z+- U+-   B V+- W+- Q+- P+-
1   10  01   11  11         1   11   11  11  11
```

On the PA-CONTROL with PLS5, the Standby signals of the individual power output stages (max. 4) are gated in the LV UNIT and are shown here as a general signal (AND).

Note:

The stand-by signal and limit switch of devices equipped with a positioning control (PA-CONTROL Servo) must be checked in the menu „Position control test“!

2.7.6 Display Counter

Please note:

This menu subpoint is active only in conjunction with a counter board (optional).

The user is offered the following selection after transfer into this menu point:

```
Display counter
1 = Counter (1 - 8 / 13 - 20)
```

2 = 2 Phase counter (9 - 12 / 21 - 24)

The following display appears when the first menu subpoint is selected:

```
Display counter: 1
I 1 = 0                                CNT 1 = 0
```

The number of the counter to which the following values apply is displayed in the first line (here: Counter No. 1). The status of the relevant input (here: I 1 = 0) and the value of the counter (here: CNT 1 = 0) are displayed in the second line.
The user has the possibility of displaying the values of the individual counters with the help of the "Arrow up" and "Arrow down" keys.

The following display appears when the second menu subpoint is selected:

```
Display counter: 9
A 9 = 0      B 9 = 0              CNT 9 = 0
```

The number of the counter is shown in the first line here too. The statuses of the relevant channels A and B and the value of the counter are displayed in the second line.
The user again has the possibility of choosing between the individual counters with "Arrow up" and "Arrow down" keys.

2.8 Overview, Sequence definitions menu

You are offered the following selection after you change to this menu option:

Sequence definitions
1 = Start program

2 = Program after stop
3 = Program start after stop
4 = Program after fault
5 = Labelling
6 = Program protection

Definitions are made for the automatic sequence of the PA-CONTROL in this menu.

Note:

Start programme, programme after stop, programme start and stop and programme when faults arise must be different programmes.

Start program: The PA-control starts the automatic sequence with this program when started. This program is, in principle, the main program.

Program after stop: This assignment permits special actions (e.g. closing valves, —) to be implemented when "STOP" is detected in an automatic run. This program is executed after the axes have stopped.

Attention: **In the programme after Stop the command block has the following restrictions:**
1. This programme should not call up any further programmes as sub-routines.
2. Positioning commands are not permissible in this programme.
3. If the G210 command is active, G21 should not used.
4. The commands „Wait for logical state of...“ inputs, outputs and flags should not be used.
5. A time monitoring that is still active is reset by the operating system without further indication and is not reactivated with a „Start“.

Program start after stop: If the PA-CONTROL was in Automatic mode and if Stop has been pressed and it is then intended to continue with Start (Automatic has not been quit), "Program start after stop" is executed before continuing with the interrupted program. This option can be used to reset actions which have been triggered on Stop.

Attention: **This program may call no other programs as subroutines. Positioning instructions are not permitted**

Program after fault: This assignment permits special actions (e.g. closing valves, ...) to be implemented when a fault is detected in the program run (error message, e.g.: value too high, power section not ready, —).

Attention: **This program may call no other programs as subroutines. Positioning instructions are not permitted**

Labels: Texts can be assigned to the inputs, outputs, flags and registers in order to simplify diagnosis during commissioning and troubleshooting. These texts are then displayed in menu option "Program test and diagnosis" in the corresponding sub-menu at the "Label text" position. In order to do this, you must create a text program, whereby the first program line is assigned to the 1st element and the second program line is assigned to the 2nd element etc. The program lines should not be longer than 20 characters in such cases since maximum 20 characters can be displayed in the diagnosis menu.

User title: The operator has the option of replacing the title in the main menu "PA-CONTROL Vxx.xx" by his own title, matched to his application. For this purpose, the operator must create a text program (type: PTX) and assign this text program under sub-menu option "User title". The first program line of this text program is then displayed as the title in the main menu.

Program protection: PA-CONTROL programs can be protected against inadvertent deletion, overwriting and modification by providing them with a program protection attribute. PA-CONTROL programs with write protection cannot be deleted. They can be displayed but not modified in the PA-CONTROL editor. If you intend to delete or modify a protected program, you must cancel the program protection attribute beforehand.

2.8.1 Start program, program after stop, program start after stop

When you activate these sub-options, you are offered a list of possible programs of type "PNC", sorted alphabetically. You can page through the list and select the required program.

Program list, please select!
Program: 1 EXAMPLE.PNC

After you make the selection, the selected program is accepted.

This assignment must be made with extreme care since it influences the entire machine sequence.

2.8.2 Text assignments

When you activate this sub-option, you are offered a further menu list.

Label Program name
1 = Inputs :

2 = Outputs :
3 = Flags :
4 = Real number register :
5 = Integer number register :
6 = User title :

Here as well, you are offered a list of the possible programs, sorted alphabetically, when you activate these sub-options. You can page through the list and select the required program.

Program list, please select!
Program: 1 EXAMPLE.PTX

2.8.3 Deleting assignments:

If you wish to delete an assignment (start program, ..., text program), you must display the assignment to be deleted in the second line. Press key "DEL" to tell the PA-CONTROL you wish to delete an assignment. You must confirm the action before the assignment is really deleted.

Delete program assignment ?	**1 = yes**
1 = Start program.....:	**EXAMPLE**

However, if you wish to only modify an assignment, you can simply overwrite the existing assignment by reselecting.

2.9 Parameters

Overview, Parameters menu

You are offered the following selection after you change this menu option:

Parameters **1 = System parameters**

2 = Axis parameters
3 = Edit ASi-Bus

A distinction is made in the PA-CONTROL between two parameter types, system parameters and axis parameters.

The significance of the individual parameters and how to use them practically are explained in Chapter Parameters.

You are offered a list of the parameters, as follows, when you enter menu sub-option "System parameters".

You must first select the required axis when you enter Axis parameters.

Parameters (definitions) **Min. value <= : Actual value <= Max. value**

You can now page through the list of the parameters and edit the required parameter by activating the input field by pressing a digit key or the cursor key "Arrow left". The value keyed in (modified) must lie between the min. value and the max. value. Otherwise, an error message is displayed when you terminate the entry. The entry must then be repeated or aborted by "ESC".

The submenu process ASI-BUS can only be called up when one or more AS-i cards are connected. For further information on projecting the AS-i see chapter on “Options“.

2.10 Defaults

Overview, Defaults menu

You are offered the following selection when you change this menu option:

Defaults **1 = Bootload system parameter**

2 = Bootload axis parameter
3 = Clear memory
4 = Reinitialize PA-CONTROL

Using these functions, you can set the PA-CONTROL to a defined initial state.

Current machine parameters are overwritten or programs are deleted when you select these functions.
These functions should be selected only by authorized and trained personnel.
Note: Back up your data beforehand on PROPAC or hard copy

The values to which the parameters are set are specified in Chapter Parameters.

Loading default and reinitialization have been carried out by IEF Werner prior to delivery. Reinitialization should be carried out if the CPU board is exchanged, unless otherwise directed by the servicing personnel (e.g. servicing department has made the settings).

When you select one of these options, you must confirm the operation once again by pressing digit
key "1".

e.g.: Load default system parameters

You will see the following on the display:

Confirm re-initalize PAC **1 = Yes / Key = No_**

You can now start loading default system parameters by pressing digit key "1" or you can abort the action by pressing another key.

2.11 System diagnosis

Overview, System diagnosis menu

You are offered the following selection after you change to this menu option:

System diagnosis
1 = Display PAC type

2 = Keyboard test
3 = Stop key test
4 = Start key test
5 = Key switch test
6 = Test serial interface 1
7 = Test serial interface 2
8 = Interbus-S monitor

Note:

If the Interbus-S system (network) is in operation, you will see sub-menu option "8 = Interbus-S monitor" in the "System diagnosis" menu.
The menu item „7 = test serial interface 2“ has no function when equipped with PIO-x (PA-CONTROL-SC and PA-CONTROL Single)

2.11.1 Display PA-CONTROL type

In this sub-option, you can recall the configuration of your PA-CONTROL. The information is displayed in encoded form. You can quit the display and return to the menu by pressing key ESC. You will see 13 pairs of digits on the display, separated by a hyphen.

PAC type:
00-00-00-00-00-00-00-00-00-00-00-00-00

The following allocations apply:

Pair of Digits	Pos.	Value	Explanation
1.	**1.**		
		1	- with LC display and keyboard
		2	- without LC display and keyboard, with 7 segment display
	2.		**Design**
		1	- 19" control unit
		2	- 19"/2 control unit
		3	- Compact 19“ - 2 axis
		4	- Compact 19“ - 1 axis
		5	- Single
		6	- SC-Compact
2.			**Positioning Process**
		0	- STEP (none)
		1	- PLS4 (from V3.3 upwards no longer in production)
		2	- PLS5
		3	- PLS6
		4	- PIO-1
		5	- Positioning control
3.			**Maximum number of axes (1 - 8)**
4.	**1.**		**Type of output stage**
		0	- without output stage
		1	- 2 phase stepping motor without rotation control
		2	- 2 phase stepping motor with rotation control
		3	- 5 phase stepping motor without rotation control
		4	- 5 phase stepping motor with rotation control
		5	- 3 phase stepping motor without rotation control
	2.		**Supply voltage of the output stage**
		0	- none
		1	- 35V
		2	- 60V
		3	- 90V
		4	- 140V

Pair of Digits	Pos.	Value	Explanation
5. / 6.			**Software status PA-CONTROL CPU**
7. / 8.			**Software status positioning module**
9. / 10.			**Software status of the PAB (Option)**
11.	**1.**		
		0	- no external bus coupling
		1	- InterBus-S (IBS)
		2	- one actuator sensor interface (AS-i)
		3	- two actuator sensor interfaces (AS-i)
		4	- three actuator sensor interfaces (AS-i)
		5	- IBS and one AS-i
		6	- IBS and two AS-i
		7	- IBS and three AS-i
	2.		
		0	- none
		1	- CNT-1
		3	- CNT-1 and CNT-2
12.	**1.**		
		0	- none
		1	- REL 9R Relay card
	2.		**Number of I/O cards (1-8)**
13.			**Size of installed RAM**

Example:

PAC - Type:
13-03-02-13-03-60-03-20-03-61-20-01-128

1. Pair of digits: - Compact 19" slide-in with 2 axes with LCD and keyboard
2. Pair of digits: - PLS6
3. Pair of digits: - 2 axes
4. Pair of digits: - 2 phase stepping motor with rotation control
- 90V output stage supply voltage
5./6. Pair of digits: - Master software 3.60
7./8. Pair of digits: - Positioning module software 3.20
9./10. Pair of digits:- PAB Software status 3.61
11. Pair of digits: - One As-i master, no CNT card
12. Pair of digits: - no relay card, one I/O card
13. Pair of digits: - 128kB RAM store

2.11.2 Tests

Keyboard test

You will see the following display when you select this sub-option:

```
Keyboard test, ESC = End !_
```

When pressed, the keys are displayed or their functions are executed, dependent upon the described function (see Chapter Operator interface - Keyboard).
Exceptions: Arrow keys, INS, DEL, ENTER, START, STOP, ALT, CTRL

Stop key test

You will see the following display when you select this sub-option:

```
Stop key test, ESC = End !
Stop key pressed: no_
```

When you press the Stop key, the display changes from "no" to "yes". The function relates to the Stop key on the front panel. The possible external Stop input can be checked via Diagnosis, Inputs.

Start key test

You will see the following display when you select this sub-option:

```
Start key test, ESC = End !
Start key pressed: no_
```

When you press the Start key, the display changes from "no" to "yes". The function relates to the Start key on the front panel. The possible external Start input can be checked via Diagnosis, Inputs.

Key-operated switch test

You will see the following display when you select this sub-option:

Key switch test, ESC = End ! **Key switch set to : Program**

The display changes from "Program" to "Automatic" when you turn the key-operated switch.

Test serial interface 1 (2)

You will see the following on the display when you select this sub-option:

Serial interface 1 (2) test, ESC = End

All keys pressed are displayed in accordance with their function and, at the same time, the corresponding signals are transferred via the selected serial interface.
Characters received via the selected serial interface are displayed immediately.

The characters of the keys now pressed (practical selection: 0-9, A-Z) are

1. shown on the display
2. transferred via the selected serial interface

At the same time, characters received from the selected interface are displayed. A jumper can be fitted between transmitter and receiver (pin 2-3) on the Sub-D connector for checking the transmit and receive modules. The character of the keys pressed is then displayed twice.

From the software version 3.20 upwards RS232/ RS485 are optional depending on the position of the jumper J11 and J12 (see chapter on “Technical Appendix -CPU 3- Card“ and chapter “Commands, Text and Value Outputs via Current Data Channel“).

Interbus-S Monitor

The menu item Interbus-S Monitor can only be selected when communication has been made with the Interbus-S master.
For further information see chapter “Options, Interbus-S Connection“.

2.12 7-segment status display

Fehler! Keine gültige Verknüpfung.

On units equipped with the 7-segment status display, the number displayed indicates the status information of the PA-CONTROL.

Number displayed	Status of the PA-CONTROL
0	PA-CONTROL is in initial state (main menu). It is possible to communicate with the PA-CONTROL with the aid of PROPAC and a PC via the serial interface (exchanging programs, editing parameters and diagnosis etc.).
1	The PA-CONTROL is in Automatic mode and a program is currently being executed.
2	The PA-CONTROL is in Automatic mode but "Stop" has been detected. The program run can be continued with Start or can be aborted (see Chapter Operator interface - Automatic mode).
3 - F	The PA-CONTROL was in Automatic mode and an error has occurred (significance of the error numbers, see Chapter Technical annex - Error messages).

2.13 System initialization

Important basic settings are transferred from the EPROM to the CRAM during system initialization. If this is not carried out at least once, there may be a possibility of undefined system functions.
The system parameters (see Chapter Parameters) are set to the relevant default value!

Please note:

External Stop input No. = 0
External Start input No. = 0
etc.

System initialization of the PA-CONTROL is absolutely essential in the following cases::

- when commissioning the CPU3 for the PA-CONTROL for the first time (carried out at IEF Werner)
- after exchanging the RAM chip on the CPU3
- after exchanging the battery on the CPU3
- after changing the EPROM (new software) on the CPU3

A distinction must be made between the units with LCD and keyboard and the units with 7-segment status display as regards the procedure for system initialization of the PA-CONTROL.

2.13.1 System initialization for units with LCD and keyboard

- Switch off the unit
- Set the key-operated switch to position "Program" (horizontal)
- Keep the "SHIFT" and "9" key pressed and switch the unit on
- You will see the following display after approx. 2 s

Confirm bootload system parameters **1 = yes / Key = no**

When you press digit key "1", loading of the default system parameters is started and the PA-CONTROL then responds with the main menu. Pressing any other key aborts the procedure which, in this case, may lead to a system crash.

-Select sub-option "4" = Reinitialize PA-CONTROL in menu option "7 = Defaults".

Note:

This procedure resets all flags, sets all registers to 0, deletes all programs and loads the parameters with the default values.

2.13.2 System initialization for units without LCD and keyboard

On these units, system initialization can be carried out only with the "PROPAC" program package and PC via the serial interface.

- Connect the PA-CONTROL and PC with the serial interface cable. Only interface 1 (socket connector) can be used on the PA-CONTROL for this purpose. The interface dependent upon the configuration must be used on the PC.
- Start "PROPAC" on the PC
- Select menu option "4 = Simulation of the PAC front panel" in PROPAC
- You will see message "Attempting to establish contact with PA-CONTROL" on the PC monitor
- Now switch on the PA-CONTROL
- Window "Simulation of the front panel PA-CONTROL" with the following message is displayed on the PC monitor after approx. 1 s

Load default system parameters ?
yes = SHIFT 9, no = Key

By pressing keys SHIFT and 9, you can start the default system parameter loading operation after you have confirmed the action again (see below).

Do you really want to load default system parameters ?
1 = yes / Key = no

You can start loading the default system parameters by pressing digit key "1", and the PA-CONTROL then responds with the main menu. Pressing any other key aborts the procedure which, in this case, may lead to a system crash.

-Select sub-option "4 = Reinitialize PA-CONTROL" in menu option "7 = Defaults".

Important:

During this operation, all flags are reset, all registers are set to 0, all programs are deleted and the parameters are loaded with the default values.

3 Instructions of the PA-Control family

Contents

3.1 Overview instruction set

3.1.1 Positioning instructions

The positioning instructions are possible with all available axes !

3.1.2 I/O processing

3.1.3 Dwell and monitoring times

* Only use with devices with pulse generation PLS5, 2 from 8 axes!
Not implemented with the PA-CONTROL SC / PA-CONTROL Single / PA-CONTROL Servo!
** Only available in the PA-CONTROL Single and/or PA-CONTROL SC!
*** Only available in the PA-CONTROL Servo!
**** Not available in the PA-CONTROL Servo

3.1.4 Instructions for program organization

3.1.5 Special functions

3.1.6 Value and text output via serial interface

3.1.7 Read in value from serial interface

If transfer errors or invalid values are recognized, a jump occurs to the point defined by a marker.

3.1.8 Register mapping to outputs or flags

3.1.9 Load register contents from inputs

3.1.10 Load Register Contents from Flag

3.1.11 Arithmetic operations

Note:

There are 512 real number registers (R1 - R512) and 512 integer number registers (N1 - N512) in the PNC programmes.
The registers R1 - R128 and N1 - N128 are in the PAB!

In the examples the register numbers are identified by the letters n, m and i.

Direct addressing with real number register:
(The same operations are also available for the integer registers (Nn, Ni, Nm))
Please note: N5:=3 is possible, N5:=5.7 is not permitted

Rn:=245.73	Load register with 245.73	3-122
Rn:=-1.0	Load register with -1.0	3-122
Rn:=X	Load register with absolute position of the X axis	3-122
Rn:=Rm	Load register Rn with contents of Rm	3-122
Rn:=Rn+56	Increment register content by 56	3-124
Rn:=Rn-56	Decrement register content by 56	3-125
Rn:=Rm*2	Multiply register Rm by 2 and assign results to Rn	3-126
Rn:=Rm/2	Divide register Rm by 2 and assign results to Rn	3-127
Rn:=Rn+Rm	Increment register content by register content of Rm	3-124
Rn:=Rn-Rm	Decrement register content by register content of Rm	3-125
Rn:=Rm*Ri	Multiply register Rm by Ri and assign result to Rn	3-126
Rn:=Rm/Ri	Divide register Rm by Ri and assign result to Rn	3-127
Rn:=Rai	Load register with axis parameter	3-123

Indirect addressing:

This is identified by "!" between R and the register number. The register of n serves as the destination or source address.

Example:

R3:=5
R!3:=245 Corresponds to function R5:=245

Indirect addressing is permitted for all register operations

Conversion:

R1:=N45	Load real number register 1 with the value from integer register 45	3-122
N23:=R56	Load integer register 23 with the value from real number register 56. The real number is rounded	3-122

3.1.12 Compare operations

A logical state (0/1) is always assigned to the addressed flag!

M1:=R2=5	M1 is set if content of R2 is equal to 5; otherwise it is reset	3-129
M5:=R2>5	M5 is set if content of R2 is greater than 5; otherwise it is reset	3-129
M1:=R2<5	M1 is set if content of R2 is less than 5; otherwise it is reset	3-129
M8:=N2=R4	M8 is set if content of R2 is equal to N4; otherwise it is reset	3-129
M1:=R7>N5	M1 is set if content of R7 is greater than N5; otherwise it is reset	3-129
M1:=R5<N8	M1 is set if content of R5 is less than N8; otherwise it is reset	3-129
M1:=N2=5	M1 is set if content of N2 is equal to 5; otherwise it is reset	3-129
M5:=N2>5	M5 is set if content of N2 is greater than 5; otherwise it is reset	3-129
M1:=N2<5	M1 is set if content of N2 is less than 5; otherwise it is reset	3-129
M8:=N2=N4	M8 is set if content of N2 is equal to N4; otherwise it is reset	3-129
M1:=N7>N5	M1 is set if content of N7 is greater than N5; otherwise it is reset	3-129
M1:=N5<N8	M1 is set if content of N5 is less than N8; otherwise it is reset	3-129

Indirect addressing can be used here as well for the registers. In the case of compare operations, the first operand is always a register and the second operand may be a register or a constant.

3.1.13 Instructions for logic operations

Please refer to the Operating instructions for the parallel sequential control (PAB) for a detailed description with examples further to the instructions listed below.
Two registers are used for implementing the logic operations in the PAB: The "bit accumulator" and the "bit accumulator stack". The result of a logic operation between two elements is stored in the two registers for further processing.

LD I3.1	Opens a new path for logic operations, store state of the "bit accumulator" in the "bit accumulator stack" and load the bit accumulator with logical 1 if input 3 is energized, otherwise, with logical 0.	3-134
AND I4.0	Performs an AND operation on the "bit accumulator" and the listed element and stores the result in the "bit accumulator". If the state of the element corresponds to the listed notation (in this case: input 4 not energized), the bit accumulator is AND-ed with a logical 1 (it remains unchanged). Otherwise, the bit accumulator is AND-ed with a logical 0 (loaded with a logical 0).	3-134
OR M12.0	Performs an OR operation on the "bit accumulator" and the listed element and stores the result in the "bit accumulator". If the state of the element corresponds to the listed notation (in this case: flag 12 not set), the bit accumulator is OR-ed with a logical 1 (it is loaded with a logical 1). Otherwise, the bit accumulator is OR-ed with a logical 0 (it remains unchanged).	3-135
OUT 05	Influences the list of elements corresponding to the contents of the "bit accumulator". "Bit accumulator" equal to "1" element is set. "Bit accumulator" equal to "0" element is reset.	3-134
AND-LD	Performs an AND operation on the "bit accumulator" and the "bit accumulator stack" and stores the result in the "bit accumulator".	3-136

OR-LD Performs an OR operation on the "bit accumulator" and the "bit accumulator stack" and stores the result in the "bit accumulator". 3-137

3.1.14 Communication with the parallel sequential control

Please note:

These commands can only be used in conjunction with the PAB (optional).

RUN Name Start program Name from the PAB or continue executing program "Name" (if "SLEEP" was active) 3-139

SLEEP Name Stop execution of program "Name" on the PAB and set it to mode "SLEEP" 3-141

CANCEL Name Terminate execution of program "Name" on the PAB 3-142

CASE.RUN.Ni Starting PAB programmes depending on a integer number register 3-143
(TYPE_001) Ni=1: Start PAB programme " TYPE_001.PAB"
(TYPE_002) Ni=2: Start PAB programme " TYPE_002.PAB"
(TYPE_003) Ni=3: Start PAB programme " TYPE_003.PAB"
ELSE NEUWAHL when Ni<1 or Ni>3 (only 3 jump marks defined here) then jump to mark "NEUWAHL"

CASE.SLEEP.Ni Stopping PAB programmes depending on a integer number register 3-145
(TYPE_001) Ni=1: Stop the PAB programme " TYPE_001.PAB"
(TYPE_002) Ni=2: Stop the PAB programme " TYPE_002.PAB"
(TYPE_003) Ni=3: Stop the PAB programme " TYPE_003.PAB"
ELSE NEUWAHL when Ni<1 or Ni>3 (only 3 jump marks defined here) then jump to mark "NEUWAHL"

CASE.CANCEL.Ni Terminates the PAB programmes depending on a integer number register 3-147

(TYPE_001) Ni=1: Terminates the PAB programme " TYPE_001.PAB"

(TYPE_002) Ni=2: Terminates the PAB programme " TYPE_002.PAB"

(TYPE_003) Ni=3: Terminates the PAB programme " TYPE_003.PAB"

ELSE NEUWAHL when Ni<1 or Ni>3 (only 3 jump marks defined here) then jump to mark "NEUWAHL"

3.1.15 Commands for controlling the counters:

Please note:

These commands can only be used in conjunction with the counter board (optional).

CNT1:=10	Set counter 1 to the value 10
CNT2:=N10	Set counter 2 to the contents of the integer register 10
N1:=CNT2	Read counter 2 and store the read value in the integer register 2
CNT4.1.3456	Wait until the contents of counter 4 is greater than 3456
CNT3.1.N67	Wait until the contents of counter 3 is greater than the contents of the integer register 67
CNT10.0.3456	Wait until the contents of counter 4 is less than 3456
CNT11.0.N67	Wait until the contents of counter 3 is less than the contents of the integer register 67

3.2 Programming information

A program is identified by a program name and has an extension. This extension serves to indicate the type of program.
A program name consists of maximum 8 alphanumeric characters (0-9, A-Z). The 1st character must be a letter. Only the hyphen "-" and the underscored character "_" are permitted as special characters within the name. The program type consists of three letters.

The following program types are possible in the PA-CONTROL:

- PNC: Programs with instructions for positioning, inputs/outputs, program branching. These programs implement a program run. A syntax check is conducted in these programs when writing the program.

- PTX: Text programs for labelling the inputs/outputs, flags and registers and for the user title. No syntax check is conducted in the PTX programs.

- PAB Program with in-, output and program branching commands. This rogram is intended for parallel process Control (option).

A PNC program consists of individual instructions which are executed one after the other and must be terminated with the END instruction.
A PNC program may call a further PNC program with the SUB instruction as a subroutine (a nesting depth of up to 15 is permitted).
Conditional and unconditional jumps to labels may be executed within a PNC program. The label names are known only within a PNC program. It is thus possible (practical?) to use the same label names in different PCN programs.
Jump instructions from one PCN program to another PCN program are not possible.

A label name consists of maximum 8 alphanumeric characters (0-9, A-Z). The 1st character must be a letter. Only the hyphen "-" and the underscored character "_" are permitted as special characters within the name.

The label name is terminated by a blank, semicolon or end-of-line character. When defining the label, the label must be located in the first column of the line, headed by the "$" character. The label may be defined only once within a program. Of course, a jump from various positions to this one label is permitted.

An editor is provided for writing PA-CONTROL programs (see Chapter Operator interface for calling the editor).

The editor has a line-oriented organization, whereby the line number is always prefixed in front of the program line at the left on the display. A program may have a length of up to 999 lines.

The editor is always in Insert mode, i.e. new characters are inserted (appended) at the cursor position. A program line may have a length of up to 36 characters.

When you quit the program line, a syntax check is conducted on the instructions. You must confirm error messages of the syntax check (press a key) and the error must then be rectified. Otherwise, you cannot quit the program line and the program cannot be stored.

Key assignment in the program editor:

Key	**Response / Effect**
Arrow left	Moves one character to the left (to the start of the program)
Arrow right	Moves one character to the right (max as far as the end of the program)
Arrow down	Moves to next program line down (max. last line)
Arrow up	Moves to previous program line (max. first line)
SHIFT + Arrow left	Moves to first column of the current program line
SHIFT + Arrow right	Moves to the end of the current program line
DEL	Deletes character at which the cursor is located (beyond line boundary)
SHIFT + DEL	Deletes character to the left of the cursor (beyond line boundary)
ENTER	Inserts a new program line at the cursor position
ESC	Quits the program editor and stores the program on request

3.3 Programming elements

Various elements which can be addressed during programming are available in the PA-CONTROL.

Inputs	:	I
Outputs	:	O
Flags	:	M
Timers	:	T
Real number registers	:	R
Integer registers	:	N
Axes	:	X, Y,
Labels	:	$Label
Comment	:	;

Inputs:	An input can be scanned for its logical state
Outputs:	Logical state 0 or 1 can be assigned to an output. In the case of conditional jumps, a logical state may be scanned.
Flags:	Logical state 0 or 1 may be assigned to a flag and a logical state can be scanned.
Timers:	Timers are used to generate defined waiting times.
Real number registers:	Real number registers (1-512) are special memory locations for real numbers (+/- 8.000.000,000). The register contents can be used for positioning and for arithmetic and compare operations. Up to 3 positions after the decimal point are permitted.
Integer registers:	Integer registers (1-512) are special memory locations for integers (+/- 8.000.000). The register contents can be used for timers and for counting, arithmetic and compare operations. No digits after the decimal point are permitted.
Axes:	Axes are stepping motor axes and serve the purpose of positioning. They are addressed with names X, Y, Z, U, V, W, Q, P (only inasmuch as these names are present in the control).
Labels:	Labels are orientation points and serve as destinations for conditional and unconditional jumps. Example of a label: $ROTBLAU
Comment:	A comment is initiated by a semicolon (;) and serves the purpose of program documentation (see examples in the section on explanation of the instructions) and also serves the purpose of operator prompting in Automatic mode.

3.4 Explanations

Instruction abbreviations

Instruction form:

mmmmm nnnnn

The instructions are subdivided into operand and operator or, in other words, abbreviation and a number.
The spaces between the operand and operator have been included in the "Instruction form" section for the sake of clarity and are not permitted for input in the PA-CONTROL!
There may be several instructions in one program line. Instructions of one line must be separated from each other by at least one blank.

Example:

Program : BEISPIEL
1
2

The program name for the examples has been shown in italics. The numbers in italics at the start of the line are the line numbers which are generated by the editor itself in the PA-CONTROL and which serve only the purpose of orientation in the program editor.
The examples can be keyed in directly. As in the PA-CONTROL, comments are separated with a semicolon and can be included in the program provided they do not exceed the maximum line length of 36 characters. All programs are always terminated with the END instruction.

Important: The examples have been written without a knowledge of the actual mechanical system and must always be checked for practical applicability.

Application:

A few words on the practical application of the respective instruction are contained in this section.

Description:

The description is explained once more in general terms in this section.

Important:

Here, we would like to point out special features and provide help in the form of cross references.

3.5 Wait for logical state of input

li.j

Instruction form:

nn.m

Example:

```
Program : BEISPIEL
1    I8.0                    ;Control waits until input 8 is logical 0
2    I4.1                    ;Control waits until input 4 is logical 1
3    END
```

Application:

Scanning proximity switches, pushbuttons and switches etc.

Description:

The control checks whether the input nn specified in the operator possesses the state m with the instruction I (input). If the scan response is not true, the control waits until the condition in the operator m is satisfied.

3.6 Wait for logical state of output

Oi.j

Instruction form:

Onn.m

Example:

```
Program : BEISPIEL
1   O8:=1      ;Control sets output 8 (e.g. triggering for parallel sequential control)
2   O7.1       ;Control waits until output 7 is logical 1 (has been set by parallel sequential control)
3   END
```

Application:

Scanning outputs

Description:

In the case of instruction O (output), the control checks whether the output nn specified in the operator has the state m. If the scan is not true, the control waits until the condition in the operator m is satisfied.

Note:

This instruction is practical only on a PA-CONTROL equipped with a parallel sequential control (PAB). This is because only the PAB can set the output whilst the PA-CONTROL waits for the output.

3.7 Wait for logical state of flag

Mi.j

Instruction form:

Mnn.m

Example:

```
Program : BEISPIEL
1   O8:=1      ;Control sets output 8 (e.g. triggering for parallel sequen-
               tial control)
2   M7.1       ;Control waits until flag 7 is logical 1 (has been set by
               parallel sequentia control)
3   END
```

Application:

Scanning flags

Description:

In the case of instruction M (flag), the control checks whether the flag nn specified in the operator has the state m. If the scan is not true, the control waits until the condition in the operator m is satisfied.

Note:

This instruction is practical only on a PA-CONTROL equipped with a parallel sequential control (PAB). This is because only the PAB can set the flag whilst the PA-CONTROL waits for the flag.

3.8 Set/reset output

Oi:=j

Instruction form:

On:=m

Example:

```
Program : BEISPIEL
1    O8:=1                    ;Output 8 is set to 1
2    O7:=0                    ;Output 7 is set to 0
3    END
```

Application:

Switching relays, signal forwarding to other electrical devices.

Description:

The available outputs can be influenced with instruction O (output). If the output nn is set to state m = 1, it switches through, i.e. the transistor is forward-biased, and remains switched on until it is reset with m = 0 and the transistor is rendered reverse-biased.

3.9 Set/reset flag

Mi:=j

Instruction form:

Mn:=m

Example:

```
Program : BEISPIEL
1    I8.1                    ;Control waits until input 8 is logical 1
2    M12:=1                  ;Flag 12 is set to logical 1
3    M239:=0                 ;Flag 239 is set to logical 0
4    END
```

Application:

Storing states, processing situations

Description:

The available flags can be influenced with instruction M. If the flag nnn is set to logical state m = 1, it remains in this state until it is reset with m = 0 (see also Instruction description, Compare operations).

Note:

The status of the flag is still retained even when the control is switched off.

The flags 240-256 are used by the system and are therefore not suitable for storing general information.

M250: Is set at ’1’ after recognizing the operating voltage (Power On flag).

M251: Restores the position of the key switch Automatic = ’1’, Programme = ’0’

M254: The STORE command sets the flag at ’1’. If the command has been successfully carried out the flag remains at ’1’. If an error occurs during the STORE command the flag is reset at ‘0’ (See chapter 3.21 and 3.22)

M255: The CASE.xxx commands set the flag at ’1’, when the jump mark is defined, otherwise with ELSE, the flag is reset at ’0’. (See chapter 3.19, 3.20 and 3.21)

M256: is influenced by the G23-, G120- and G123 functions (see chapter 3.27, 3.36 and 3.37).

3.10 Dwell time

T

Tnnnn
TNn

Example:

```
Program : BEISPIEL
1    T1                    ;Dwell time 10ms
2    T300                  ;Dwell time 3s
3
4    TN2                   ; Dwell time corresponding to content of N2
5    END
```

Application:

Wait times, keeping outputs set for defined times.

Description:

Instruction T results in a control dwell time for the time specified in the operator.

Possible values:

Minimum time n	=	1	=	10 ms
Maximum time n	=	8.000.000	=	80.000,00 s possible.

Important:

Any sign of the register content is suppressed. The register content remains unchanged.Instruction form:

3.11 Unconditional jump

JMP

Instruction form:

JMP Label

Example:

```
Program : BEISPIEL
1    $ENDLOS
2    I1.1                    ;Wait until input 1 is energized
3    O1:=1
4    T100
5    O1:=0
6    JMP ENDLOS              ;Jump to label "ENDLOS"
7    END
```

Application:

For program loops

Description:

An "Unconditional jump" is executed with function JMP.

3.12 Subroutine call

SUB

Instruction form:

SUB	Name

Example:

```
Program : BEISPIEL
1    I2.1
2    SUB LAMBLI          ;Subroutine "LAMBLI" is called and executed. Further
                         processing of the next instruction in the calling program
                         (in this case I3.1) occurs after recognition of the END in-
                         struction in the subroutine.
3    I3.1
4    I4.1
5    SUB LAMBLI          ;Subroutine "LAMBLI" is called and executed. Further
                         processing of the next instruction in the calling program
                         (in this case I5.1) occurs when the END instruction is re-
                         cognized in the subroutine}
6    I5.1
7    END

Program : LAMBLI
1    O1:=1               ;Output 1 is set for 1 s, then reset
2    T100
3    O1:=0
4    END
```

Description:

Subroutine call SUB allows the user to create a well-structured program. Functions which are required more than once can thus be placed in a subroutine. The main program then remains clear and subroutines can be tested individually.

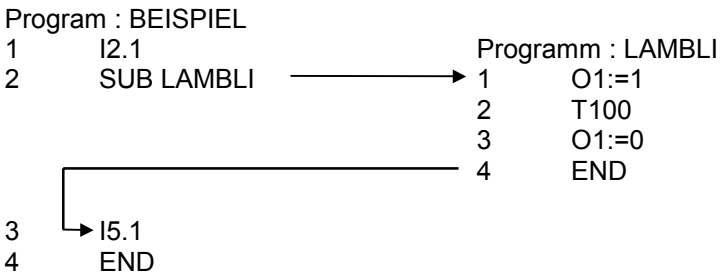
Important:

A subroutine may not call itself. Subroutines must be terminated with the END instruction. A nesting depth of up to 15 is permitted for subroutine calls.

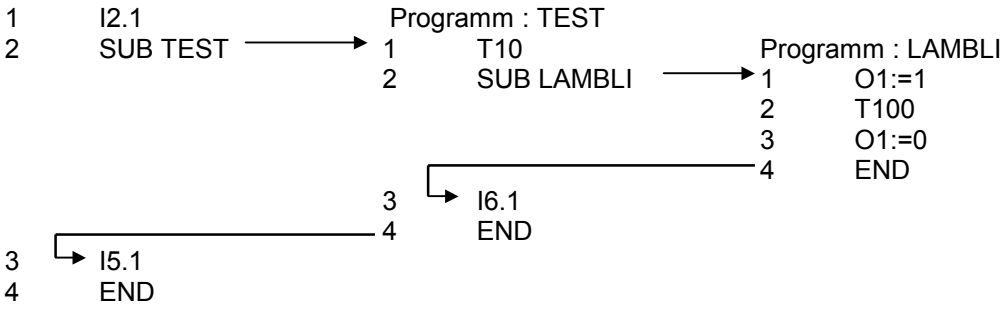
Subroutine call, continued

Examples of nestings:

1.



2.



3.13 Display register n

GRn.m
GNn.m

Instruction form:

GRn.m
GNn.m
GR!n.m
GN!n.m

Example:

```
Program : BEISPIEL
1    G24R4.0          ;Load register 4 via a keyboard
2    GR4.200          ;Display contents of register 4 for 2 s (200*10 ms) on the
                      display
3    N35:=234         ;Load register 35 with 234
4    GN35.100         ;Display content of register N35 for 1 s (100*10 ms) on
                      the display
5    END
```

Description:

The content of register n is displayed in the 1st display line for m*10 ms. The program stops during this display time.

The display is as follows, for instance:

R4 = current value

3.14 Linear interpolation with 2 out of 8 axes

G01

Instruction form:

G01 Ai An
G01 ARi ARn

Example:

```
Program: BEISPIEL
1   G25.X G25.Y G90
2   X10                 ; Traverse X axis to position 10
3   Y20                 ; Traverse Y axis to position 20
4   FB1000              ; Set tool path feedrate to 1000 Hz
5   G01 X40 Y30         ; Linear interpolation with the X and Y axes from the cur-
                          rent  position to position X40 and Y30
6   END
```

Application:

Simultaneous traversing of 2 axes involving moving on a straight line between the start and destination positions.

Description:

With linear interpolation the target position is reached with the path speed on a straight line. Movements can be carried out axis-parallel and under a given angle. The path speed is set before the first **G01** with the command **FBn**, can be redefined however at the beginning of a new interval (beginning with **G01**) with **FBn**. The path acceleration is set using the command **G100.B.n**.

Note:

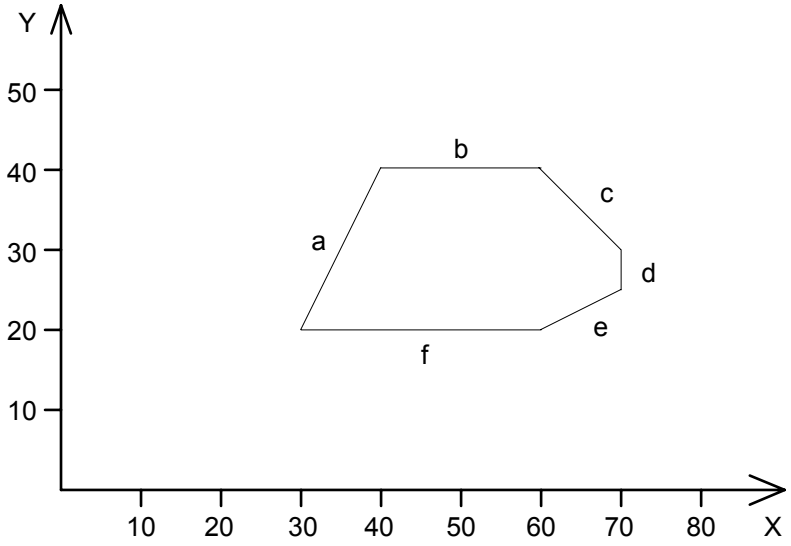
Only possible in conjunction with PLS5-X.

If interpolation commands appear immediately one after the other in the programme (without other commands in-between), these interpolation commands are then executed together at constant path speed. The linear and circular interpolation commands can be mixed.
From Version 3.2 upwards it is possible to alter the speed at the beginning of a new interval beginning with G01, (the new V path is switched to without stopping). The transfer to the new V path is carried out with the preset path acceleration. This cannot be altered within a contour. When using this command an activation of the G2xx commands is no longer possible.

Continuation

Complex example involving execution of several linear interpolation operations one after the other following a contour at constant tool path feedrate. Distances "a" to "f" are moved in one continuous string owing to the fact that the G01 instructions follow consecutively one after the other (with no other instruction in between).

```
Program: Beispiel
1    O1: = 0 I5.1          ; Raise tool
2    G25.X G25.Y G90       ; Reference travel operations
3    X30 Y20               ; Approach workpiece edge
4    O1: = 1               ; Lower tool
5    I6.1                  ; Wait until tool has been lowered
6    FB500                 ; Set tool path feedrate to 500 Hz
7    G100.B.30             ; Set tool path acceleration to 300 Hz/ms
8    G01 X40 Y40           ; Move along side edge "a"
9    G01 X60 Y40           ; Move along side edge "b"
10   G01 X70 Y30           ; Move along side edge "c"
11   G01 X70 Y25           ; Move along side edge "d"
12   G01 X60 Y20           ; Move along side edge "e"
13   G01 X30 Y20           ; Move along side edge "f"
14   O1: = 0               ; Raise tool
15   X1 Y1                 ; Approach initial position
16   END
```



3.15 Circular interpolation

G02/G03

Instruction form:

G02 An Am CAi CAj
G02 ARn ARm CARi CARj
G03 An Am CAi CAj
G03 ARn ARm CARi CARj

Example:

```
Program: BEISPIEL
1   G25.X G25.Y G90
2   X10                  ; Traverse X axis to position 10
3   Y20                  ; Traverse Y axis to position 10
4   FB1000               ; Set tool path feedrate to 1000 Hz
5   G100.B.10            ; Set tool path acceleration to 10 Hz/ms
6   G02 X20 Y30 CX10 CY0 ; Circular interpolation with X and Y axes from the current
                           position to the end position X20 and Y30, whereby the
                           center of the circle lies offset by 10 in the positive coordi-
                           nate for the X axis and lies on the start coordinate (offset
                           by 0) for the Y axis, when viewed from the start position.
7   END
```

Application:

Simultaneous traversing of 2 axes with movement along a circular arc (also a full circle) between start and destination positions.

Description:

With circular interpolation the path speed is used to move from the current position (KA) on the arc to the specified target position (KE). The centre point of the arc (KM) is specified, signed relative to the initial point by the vectors CI and CJ (see diagram) independent of the system of measuring (**G90**/**G91**). **G02** describes a circle clockwise und **G03** anticlockwise.
The path speed can be set before interpolation with the command **FBn**. The path acceleration is set using the command **G100.B.n**.
From version 3.2 upwards it is also possible to redefine the path speed at the beginning of new intervals (within a contour), beginning with **G01**, **G02** or **G03.** The new V path is switched to without stopping

Note:

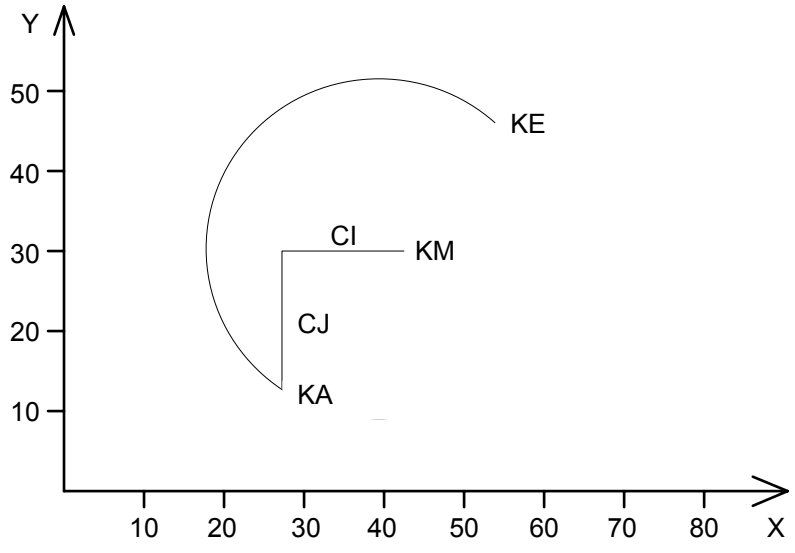
Only possible in conjunction with PLS5-X.

If interpolation commands appear immediately one after the other in the programme (without other commands in-between), these interpolation commands are then executed together at constant path speed. The linear and circular interpolation commands can be mixed.
From Version 3.2 upwards it is possible to alter the speed at the beginning of a new interval beginning with G01, (the new V path is switched to without stopping). The transfer to the new V path is carried out with the preset path acceleration. This cannot be altered within a contour. When using this command an activation of the G2xx commands is no longer possible.

Example with circular interpolation.

Program: Beispiel

```
1    G25.X G25.Y G90          ; Reference travel operations
2    X27 Y15                  ; Approach start point of circular arc
3    FB1000                   ; Set tool path feedrate to 1000 Hz
4    G100.B.30                ; Set tool path acceleration to 30 Hz/ms
5    G02 X50 Y47 CX13 CY15    ; Circular arc counterclockwise to the destination position
                              X50 and Y47, whereby the center point of the circle for X
                              is 13 (I) units away from the start position of the circle and
                              the center point of the circle for Y is 15 (J) units away
                              from the start point of the circle (in relative terms).
6    END
```



3.16 Switch display on/off

G11

Instruction form:

G11.n

Example:

```
Program : BEISPIEL
1       G11.0                ;Display switched off
2       G11.1                ;Display switched on
3       END
```

Application:

For sequences where speed optimization is necessary and display of the program steps is not required.
The user can assume the form of the display with the commands of the G5xx group.

Description:

Display of instructions during the program run is controlled with function G11. The instruction sequence is executed more quickly if the display is switched off with G11.0. The display can be switched back on again with G11.1. It is not possible to make an absolute statement on the time savings which are possible by using the G11 function since these savings depend on the number of characters in the display to be built up.

Important:

The instruction last executed is displayed irrespective of the program status.
The following instructions are never displayed: JMP, M1:=, etc.!
This permits a previous program line to be displayed when programming loops.

3.17 Conditional jump

G21

Instruction form:

G21 Ii.j	Label
G21 Ii.j	Label
G21 Oi.j	Label
G21 Oi.j	Label
G21 Mi.j	Label
G21 Mi.j	Label

Example:

```
Program : BEISPIEL
1   G21 I1.1 EIN_5          ;If input 1 is energized, program execution is continued at
                            label "EIN_5"; otherwise, it is continued with the next in-
                            struction
2   O1:=1 T100 O1:=0        ;Output 1 set for 1 second
3   $EIN_5
4   T10
5   G21 O5.1 MERK_31        ;If output 5 is set, the program is continued at label
                            "MERK_31" otherwise, it is continued with the next in-
                            struction
6   O1:=1 T100 O1:=0        ;Output 1 set for 1 second
7   $MERK_31
8   T10
9   G21 M31.0 ENDE          ;If flag 31 is logical 0, the program is continued at label
                            "ENDE"; otherwise, it is continued with the next instruc-
                            tion
10  O1:=1 T100 O1:=0        ;Output 1 set for 1 second
11  $ENDE
12  T10
13  END
```

Application:

Program branch depending on the logical state of an input, flag or output.

Description:

A "conditional jump" is executed with function G21. The condition specified in the operand may be the logical state of an input, output or flag. A jump to the label specified in the operator is performed if the condition is satisfied, and the program is continued at this point. If the condition is not satisfied, the program is continued in the next line.

Note:

There may be no other instruction in this line after the specification of the jump destination. A comment is permitted.

3.18 Conditional subroutine call

G22

Instruction form:

G22 Ii.j	Name
G22 Oi.j	Name
G22 Mi.j	Name

Example:

```
Program : BEISPIEL
1    G25.X G90                  ;Reference travel X axis, absolute dimension system}
2    G22 I1.1 AUSG_2            ;Falls ;If input 1 is energized, subroutine "AUSG_2" is
                                executed and execution is then continued with the next
                                instruction of the calling program
3    G22 M21.1 AUSG_2           ;If flag 21 is set, subroutine "AUSG_2"  is executed and
                                execution is then continued with the next instruction of the
                                calling program
4    G22 O7.1 AUSG_2            ;If output 7 is set, subroutine "AUSG_2" is executed and
                                execution is then continued with the next instruction of the
                                calling program
5    END


Program : AUSG_2
1    O2:=1 T100 O2:=0 END       ;Set output 2 for 1 s
```

Application:

A subroutine is executed depending on the logical state of an input, flag or output.

Description:

A "conditional subroutine call" is executed with function G22. The condition specified in the operand may be the logical state of an input, output or flag. The subroutine call is executed corresponding to the operator if the condition is satisfied. The program is continued with the next instruction of the calling program as soon as the subroutine has been executed or if the condition is not satisfied.

Note:

There may be no other instruction in this line after the specification of the subroutine name. A comment is permitted.

3.19 Case jump

CASE.JMP

Instruction form:

```
CASE.JMP.Nn
(Label1)
(Label2)
...
(Label_i)
ELSE Label_e
```

Example:

```
Program : BEISPIEL
1    $GRAU
2    G24.N8.0                ;Read in value via keyboard
3    CASE.JMP.N8             ;Check content of N8 and branch if:
4    (ROT)                   ;N8 = 1: Jump to label "ROT"
5    (BLAU)                  ;N8 = 2: Jump to label "BLAU"
6    (GELB)                  ;N8 = 3: Jump to label "GELB"
7    (ROSA)                  ;N8 = 4: Jump to label "ROSA"
8    ELSE GRAU               ;Jump to label "GRAU" if N8<1 or N8>4 (value of N8 not
                             within the scheduled range which is 1 to 4 in this case).
9    $ROT
10   O1:=1 T100 O1:=0        ;Set output 1 for 1 s
11   JMP ENDE
10   $BLAU
12   O2:=1 T100 O2:=0        ;Set output 2 for 1 s
13   JMP ENDE
14   $GELB
15   O3:=1 T100 O3:=0        ;Set output 3 for 1 s
16   JMP ENDE
17   $ROSA
18   O4:=1 T100 O4:=0        ;Set output 4 for 1 s
19   $ENDE
20   END


Program: AUSG_2
1    O2:=1 T100 02:=0 END    ; Set output 2 for 1 s
```

Application:

Program branch if there are more than two options, e.g.: Different production types, variations of traversing frequencies etc.

Description:

The CASE.JMP instruction is a program branch as a function of the content of an integer register. Case jumps can be set up simply with this instruction.
The PA checks the content of the integer register and jumps, corresponding to the value, to the labels. If the content of the integer register is less than 1 or greater than the number of elements in the label table (in the example: (ROT), (BLAU), (GELB), (ROSA), a jump is executed to the label after ELSE; otherwise, a jump is executed to the corresponding label.

3.20 Case subroutine

CASE.SUB

Instruction form:

```
CASE.SUB.Nn
(Name1)
(Name2)
...
ELSE Label_e
```

Example:

```
Program : BEISPIEL
1    $FEHLER
2    G24.N8.0              ;Read in value via keyboard
3    CASE.SUB.N8           ;Check contents of N8 and call subroutine if:
4    (KLEIN)               ;N8=1 :  Run subroutine "KLEIN"
5    (GROSS)               ;N8=2 :  Run subroutine "MITTEL"
6    (MITTEL)              ;N8=3 :  Run subroutine "GROSS"
7    ELSE FEHLER           ;Jump to label "FEHLER" if value of N8 lies outside sche-
                           duled limits (in this case if N8<1 or N8>3)
8    O12:=0
9    I1.1
10   END


Program :  KLEIN
1    O1:=1 T100 O1:=0      ;Set output 1 for 1 s
2    END


Program : GROSS
1    O2:=1 T100 O2:=0      ;Set output 2 for 1 s
2    END


Program : MITTEL
1    O3:=1 T100 O3:=0      ;Set output 3 for 1 s
2    END
```

Application:

Program branch if there are more than two options, such as: different production types, variations of traversing frequencies, ...

Description:

The CASE.SUB instruction is a program branch as a function of an integer register. Case programs can be set up simply with this instruction.
The PA checks the content of the integer register and performs a subroutine call depending on the value. If the content of the integer register is less than 1 or greater than the number of elements of the name table (in this example: KLEIN, MITTEL, GROSS), a jump is made to the label defined after ELSE. Otherwise, the corresponding subroutine is called and run, and the system then continues with the program line after the ELSE branch (in the example: line 8 with O12:=0).

The following assignment applies:

Value in register	Name in name table
1	1st name
2	2nd name
...	...
n	nth name

Note:

The subroutines must exist. Otherwise, an error message "Subroutine not found" is issued.
Subroutines may not call themselves.

3.21 Storing Values in PNC Programmes

CASE.STORE

Command form:

CASE.STORE.Ni
(Name1)
(Name2)
...
ELSE Marke_e

Example:

```
Programme:BEISPIEL
1    $FEHLER
2    G24.N8.0                  ;Enter value via keyboard
3    CASE.STORE.N8             ;Check the contents of N8 and call up subroutine at:
4    (TYPE_001)                ;N8=1 : subroutine " TYPE_001" check for allocations and
                                update
5    (TYPE_002)                ;N8=2 : subroutine " TYPE_002" check for allocations and
                                update
7    ELSE FEHLER               ;Jump to "ERROR" mark, if the value of N8 is beyond the
                                intended limits (here when N8<1 or N8>2)
8    END


Programme: TYPE_001
1    R10:=0_ _ _ _ _ _ _ _ _ _ ;X-Collect position for product type 1
2    R11:=0_ _ _ _ _ _ _ _ _ _ ;X-Intermediate position for product type 1
3    N10:=0_ _ _ _ _ _ _ _ _ _ ;Number of products to be machined
4    M101:=0                   ;Product flag for product 1
5    END

Programme TYPE_002
1    R10:=0_ _ _ _ _ _ _ _ _ _ ;X-Collect position for product type 2
2    R11:=0_ _ _ _ _ _ _ _ _ _ ;X-Intermediate positin for product type 2
3    N10:=0_ _ _ _ _ _ _ _ _ _ ;Number of products to be machined
4    M102:=0                   ;Product flag for product 2
5    END
```

Note:

In the example blanks are represented by an underline " _ ".

Application:

Programme branching for storing current register values in different PNC files. For different product types traversing positions (R- register) or status flags for example can be stored and called up again as required.

Description:

The **CASE.STORE** command is a programme branching that is dependant on the contents of the integer number register **Ni**. The PA control checks the contents of the integer number register and performs a subroutine call-up in accordance with the value. If the contents of the integer number register are smaller than 1 or greater than the number of elements of the name table (here an example: TYPE_001, TYPE_002), a jump is made to the mark that is defined behind ELSE. Otherwise the respective subroutine is called up and processed and continued with the programme line after the ELSE branch.
With the command **CASE.STORE** all the current values of the N registers, R registers and flags filed in a PNC programme.
When carrying out the **CASE.STORE** command the called-up PNC programme is checked for allocations. If a register and/or flag allocation is found the current value is entered in the allocation. There must be sufficient blanks available for this purpose.

- Maximum number of space holders per register : 9 digits
- Maximum number of space holders per flag :1 digit

Before carrying out STORE commands

```
1   R10:=0_ _ _ _ _ _ _ _ _ _   ;X collect position
2   R11:=0_ _ _ _ _ _ _ _ _ _   ;X intermediate position
3   N10:=0_ _ _ _ _ _ _ _ _ _   ;Number of  products to be processed
4   M101:=0                     ;Product flag for product 1
5   END
```

After carrying out the STORE command

```
1   R10:=145.345_ _ _ _         ;X collect position
2   R11:=34565.789_ _           ;X intermediate position
3   N10:=645_ _ _ _ _ _ _ _     ;Number of products to be processed
4   M101:=1                     ;Product flag for product 2
5   END
```

Note:

The called-up PNC programme must be available. A programme line should only contain one allocation.
There must be sufficient zeros or blanks for overwriting after the allocation to ";" (comment) or "CR" (end of line).
Calculation allocations are not allowed in overwrite mode.
The STORE command sets the flag 254 at '1'. If the command has been carried out successfully the flag remains at '1'. If insufficient digits have been reserved for the allocations the flag is reset at '0'.

3.22 Storing Values in a PNC Programme

STORE

Command form:

STORE Name

Example:

```
Programme:BEISPIEL
1    G24.R10.0          ;Enter real number register via keyboard
                        e.g. 4678.123
2    G24.N1.0           ;Enter integer number register via keyboard
                        e.g. 967
3    M25:=N1>11         :M25 is used when contents of N1 are greater than 11.
4    STORE TYPE_001     ;Overwrite allocations in the PNC programme TYPE_001
                        with current values of the registers and flags
5    END
```

Before carrying out the STORE command

```
Programme: TYPE_001
1    R10:=0_ _ _ _ _ _ _ _ _ _ ;X collect position for product type 1
2    N1:=0_ _ _ _ _ _ _ _ _ _  ;Number of products to be processed
3    M25:=0                    ;Product flag for product type 1
4    END
```

After carrying out the STORE command

```
Programme: TYPE_001
1    R10:=4678.123_ _          ;X collect position for product type 1
2    N1:=967_ _ _ _ _ _ _      ;Number of products to be processed
3    M25:=1                    ;Product flag for product  1
4    END
```

Note:

In the example blanks are represented by an underline " _ ".

Application:

Storing current register values in different PNC files. For different product types traversing positions (R register) or status flags for example can be stored and called up again as required.

Description:

With the command **STORE** the current values of N registers, R registers and flags are filed in a PNC programme.
When carrying out the **STORE** command the called-up PNC programme is checked for allocations. If a register and/or flag allocation is found, the current value is entered in the allocation. Sufficient blanks must be available for this purpose.

- Maximum number of space holders required for register: : 9 digits
- Maximum number of space holders required for flag : 1 digit

Note:

The called-up PNC programme must be available. Only one allocation is permitted per programme line.
Sufficient zeros or blanks must be available for overwriting after the allocation to ";" (comment) or "CR" (end of line).
Calculation allocations are not permitted in the programme being overwritten.
The STORE command sets the flag 254 at '1'. If the command has successfully been carried out the flag remains at '1'. If insufficient digits have been reserved for the allocations the flag is reset at '0'.

3.23 Loop with conditional jump

DEC

Instruction form:

DEC.Nn Marke

Example:

```
Program : BEISPIEL
1    G24.N2.0                  ;Read in number of loops
2    $ANFANG
3    O1:=1
4    T100
5    O1:=0
6    T100
7    DEC.N2 ANFANG             ;All retries done? No: Jump to ANFANG
8    END
```

Application:

For setting up program loops, the number of runs of which is stored in an integer register, e.g. machine n parts, traverse n times, ...

Description:

The DEC instruction serves to set up program loops. It is comparable with the Pascal instruction Repeat ...Until or the C instructions Do..While or While..DO.
The DEC instruction checks the content of the integer register. If the content of the integer register is greater than 0, the content is decremented by 1 and a jump to the label is executed; otherwise, the system continues with the next instruction (next line).

Note:

The DEC instruction can be used only for integer registers.
There may be no other instructions in this program line after the DEC instruction. A comment is permitted.

Examples of loops with the DEC instruction:

1. Loop program part is run through at least once (Repeat...Until, Do...While), i.e. the program part is always run, and a check is then conducted (DEC.) in order to establish whether the program part has to be run again.

```
Program : BEISPIEL
1    G24.N3.0
2    $MACHWAS
3    O1:=1
4    T100
5    O1:=0
6    T100
7    DEC.N3 MACHWAS
8    END
```

2. The loop program part is not run depending on value (While...), i.e. the register is checked first (DEC.N3) and then the program part is run, depending on the content of the register.

```
Program : BEISPIEL
1    G24.N3.0
2    $SCHLEIFE
3    DEC.N3 MACHWAS
4    JMP FERTIG
5    $MACHWAS
6    O1:=1
7    T100
8    O1:=0
9    T100
10   JMP SCHLEIFE
11   $FERTIG
12   END
```

3.24 Break Automatic Cycle

Break

Command form:

Break

Example:

```
Programme:BEISPIEL.PNC
1    $SCHLEIF
2    I10.1                    ;Start
3    SUB HOLEN                ;Subroutine for collect part
4    SUB BEARBEIT             ;Subroutine for process part
5    SUB ABLEGEN              ;Subroutine for deposit part
6    G21 I11.1 SCHLEIFE       ;Continue producing
7    END

Programme:HOLEN.PNC
1    G421.1.100 Fehler        ;Error, when part collected process is not completed in 1
                              s
2    O1:=1                    ;Open gripper
3    I1.0 I2.1                ;Gripper open
4    O2:=1                    ;Lower gripper
5    I3.0 I4.1                ;Gripper down
6    O1:=0                    ;Close gripper
7    I2.0 I1.1                ;Gripper closed
8    O2:=0                    ;Lift gripper
9    I4.0 I3.1                ;Gripper up
10   G401.1
11   END
12
13   §FEHLER
14   I12.1                    ; ACKNOWLEDGE ERROR WHEN PART COLLECTED
15   BREAK
```

Description:

The **BREAK** command discontinues the automatic cycle of the PAC. The PAC returns to the main menu.

Application:

The **BREAK** command is for discontinuing the automatic cycle of the PAC when for example a machine error has occurred in a subroutine. Using the **BREAK** command avoids returning to the programme structure to return to the main programme from more than one subroutine level and from there to branch to the end of the programme.

3.25 Axis positioning

X

Instruction form:

Xnnnnn
XRn

The positioning instructions are possible with all existing axes !

Example:

```
Program : BEISPIEL
1    G25.X          ;Perform reference travel with X axis
2    G90            ;Absolute dimension system
3    X200           ;Axis X 200 in positive direction
4    X198           ;Axis X by 2 in negative direction to absolute position 198
5    R3=678
6    XR3            ; Axis X 678 in positive direction
7    G91            ;Incremental dimension system
8    R3=-500
9    X-10           ;Axis X in negative direction
10   XR3            ;Axis X in negative direction
11   END
```

Application:

Positioning

Description:

The positioning instruction traverses the axes in positive or negative direction, viewed from the datum. The value entered in the operator is the traverse system in the incremental dimension system (G91).

Note:

The end point of the movement is dependent upon the programmed dimension system, either in absolute dimensions (G90) or in incremental dimensions (G91).

3.26 Traversing speed

F

Instruction form:

```
FA      nnnnn
FA      Rn
FB      nnnn                    ; speed for the interpolation
FB      Rn
```

Example:

```
Program : BEISPIEL
1    G25.X                      ;Perform reference travel with X axis
2    G91                        ;Set absolute dimension system
3    FX200                      ;Traversing speed X axis 200 Div./s
4    X1000
5    R4:=500
6    FXR4                       ;Traversing speed X axis 500 Div./s
7    X10
8    END
```

Application:

Different speeds for traversing and moving axes.

Description:

With the **F** command the control is instructed to move at the speed specified in the operator for the rest of the programme. All axes that are addressed with the operand F can be moved at different speeds.
If the traversing speed is determined by the contents of a register, the signs and the digits after the comma of the register contents are not taken into account.
The axis parameter determined in the parameter list can also be loaded via a register (see Chapter 0).
FB is for setting the path speed for the interpolation (FBmin = 50Hz, FBmax = 5000HZ).

Note::

If speed F has not be programmed during the programme run then all traversing commands are carried out at the speed defined in the parameter field.
A speed setting that is higher than the one set in the parameter field leads to the fault signal „Value too great“
A speed that is lower than the minimum one set in the parameters leads to the fault signal „Value to small“
A value smaller than 10Hz leads to the fault signal "Value too small ".

Special feature of the PAC-Servo:

With this type of control the speed can be altered during an active positioning process.

Example: Speed profile

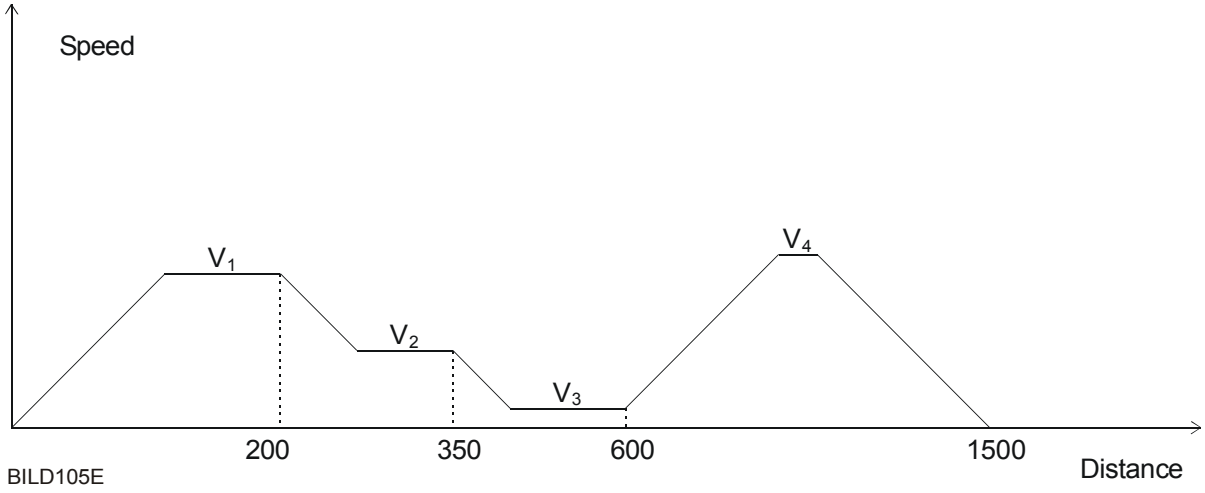


BILD105E

```
Programme :BEISPIEL
1   G90                ;Positioning performed in absolute system of measuring
2   G25.X              ;Return to datum
3   G100.X.100         ;Determine acceleration of X axis
4   G210               ;Processed commands beyond the line limit
5   FX1000             ;Traversing speed X axis v1: 1000AE/s
6   X1500              ;Positioning of the X axis
7   G230.1.X200        ;Wait until the current absolute position of the X axis is
                        greater than 200
8   FX500              ;Traversing speed X axis v2: 500AE/s
9   G230.1.X350
10  FX150              ;Traversing speed X axis v3: 150AE/s
11  G230.1.X600
12  FX1200             ;Traversing speed X axis v4 : 1200AE/s
13  G213
14  END
```

Note:

The selected speeds must be reachable in the target interval.
The acceleration <u>cannot</u> be altered during this process It must be defined beforehand.

3.27 Travel until condition is satified

G23

Instruction form:

G23 n.m

Example:

```
Program : BEISPIEL
1    G25.X G91          ;Perform reference travel with X axis, switch over to absolute dimension system
2    G23 I3.1           ;The next positioning operation is executed (in this case X100) provided input 3 is set; otherwise, the instruction after this is executed
3    X100               ;Traverse axis X 100 in positive direction
4    SUB LOCHBOHR
5    END
```

Application:

A workpiece is traversed under a recognition device until it is recognized. The recognition device then sends a signal back to the PA so that the program can be continued.

Description:

With the function G23, the next traversing instruction is executed as long as the condition n.m of the operand is valid. If the condition is not satisfied, the traversing instruction is aborted and the next instruction is executed.

Note:

The current absolute position is taken into account for subsequent positioning if the movement commands are aborted.
The command G23 resets the marker 256. If the positioning command is carried out after G23 and if the event (change of the input state) occurs during positioning, the marker 256 is set.
This allows the user to interrogate the marker 256 to determine whether the positioning process has been completed by the state change or end of travel.

3.28 Load register via keyboard

G24.Rn.0
G24.Nn.0

Instruction form:

G24.Rn.0
G24.Nn.0

Example:

```
Program : BEISPIEL
1     G25.X G91                ;Reference travel, incremental dimension system}
2     G24.R1.0                 ;Read in a value via keyboard to register R1, in this case
                                as a traversing distance
3     G24.N1.0                 ;Read in a value via keyboard to register N1, in this case
                                as a number of travel operations
4    $FAHR_1
5     XR1                      ;Travel with X axis, distance in R1
6     DEC.N1 FAHR_1            ;All travel operations completed? If not, then jump to label
                                "FAHR_1"
7     END
```

Applikation:

During execution of a PA program, it is possible to influence the program sequence by entering values via the keyboard. The entered values may be a timer, a repeat factor or for positioning.

Description:

The program waits for an operator entry via the PA keyboard after instruction G24.Rn.0 is called. The operator is requested to make an input on the display by output of the program line and current register content.

Display:

BEISPEL	**0**	**R1 [current value]**
G24.R1.0	**;Program comment**	

The displayed current register content can be overwritten with the numeric keys of the keyboard. The old value is retained if only key ESC is pressed.

Note:

The maximum number of digits/characters is 7.

No limit value check is performed for the entered value. This must be done by the user in conjunction with the compare operations.

A program abort occurs if the Stop key is pressed during input. This is indicated by "*" and message "Program abort during value input".

3.29 Load register via inputs

G24.Rn.S
G24.Nn.S

Instruction form:

G24.Rn.S. NrE.NrA.AzV.AzN
G24.Nn.S. NrE.NrA.AzV.AzN

Example:

```
Program : Beispiel
                                  ;Value is entered in register N5
                                  ;Reading is performed as of input 3
                                  ;The value to be read in has two digits in front of the de-
                                  cimal point
1   G24.N5.1.3.2.2.0              ;The value to be read in has no digits after the decimal
                                  point
                                  ;Output 2 is the first output for multiplex mode.
                                  ;No sign evaluation

2   $SCHLEIFE
3   O1:=1 T5  O1:=0 T5
4   DEC.N5 SCHLEIFE
5   END
```

Description:

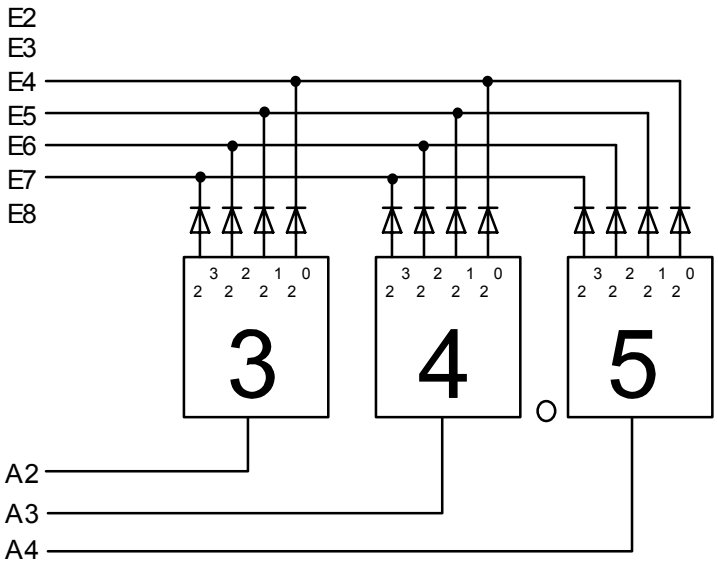
Instruction G24.Nn.S.NrE.NrA.AzV.AzN (G24.Rn) results in reading in of a value into a register with the number n (n=1 - 128). The real number with a maximum of 8 digits is supplied via inputs of the PA. Reading takes place in multiplex mode in order to limit the number of required inputs. 4 inputs and one output for each decade to be read are always occupied.
S may have the numeric value 1 or 2. A 1 identifies the number as being without a sign, while a 2 indicates that there is a sign in the highest-order decade. For this reason, the maximum number is 7 for a value with a sign in the highest-order decade. The number is negative if the highest-order bit of the decade is set to 1.
The number of outputs for multiplex mode correspond to the total of digits before and after the decimal point.

Note:

A "0" must be entered at the corresponding positions in the instruction if no digits are required before or after the decimal point. The number of digits must not exceed 8.
A time of approx. 2 ms is required for each decade for reading in. This fact must be taken into account in applications where time is critical and the read-in procedure must be performed in advance where appropriate. Register contents are retained even in the event of a mains power failure. The inputs are read in as a BCD value (0-9). The required outputs cannot be used as flags since they assume the value "0" after reading in.

Assignments of inputs and outputs:

Example: G24.R1.1.4.2.2.1



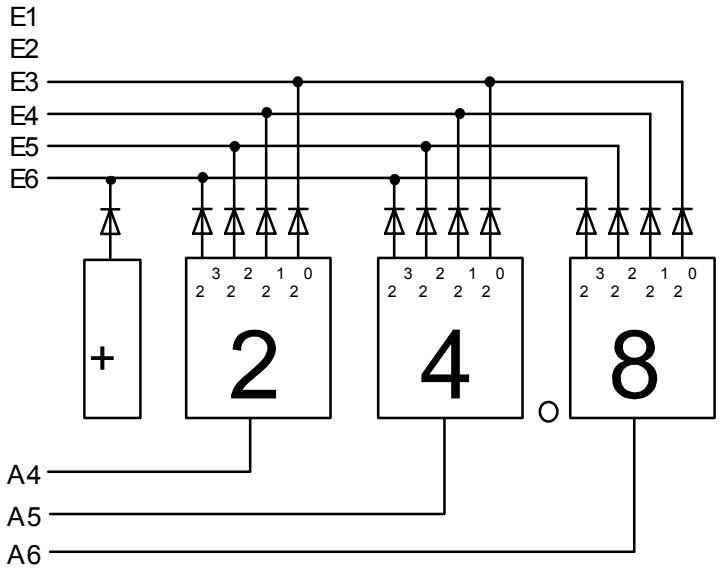
- No sign evaluation
- Input 4 is the first input for reading
- Output 2 is the first output for multiplex mode
- 2 digits before the decimal point
- 1 digit after the decimal point, i.e. 3 digits !

States of the inputs E4 to E7 for the number 34.5 (with the corresponding control in multiplex operation):

	E4	E5	E6	E7
3	1	1	0	0
4	0	0	1	0
5	1	0	1	0

Assignment of inputs and outputs:

Example: G24.R1.2.3.4.2.1



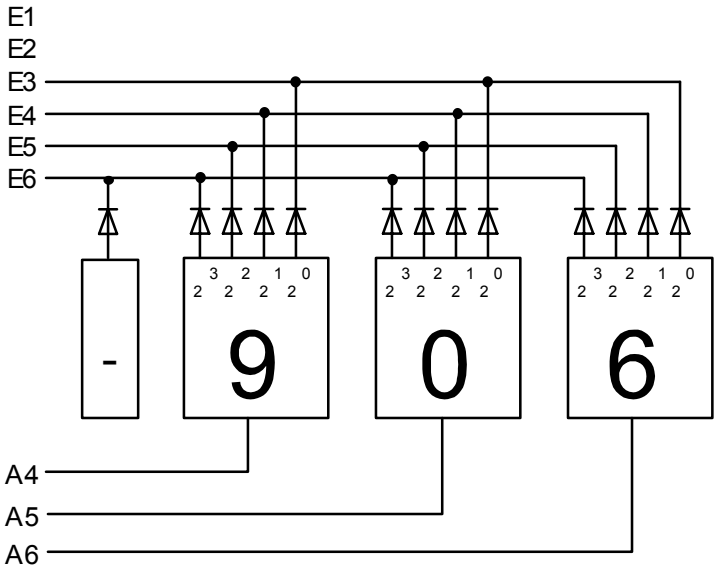
- Sign evaluation
- Input 3 is the first input for reading
- Output 4 is the first output for multiplex mode
- 2 digits before the decimal point
- 1 digit after the decimal point, i.e. 3 digits !

States of the inputs E3 to E6 for the number +24.8 (with the corresponding control in multiplex operation):

	E3	E4	E5	E6
+	-	-	-	0
2	0	1	0	0
4	0	0	1	0
8	0	0	0	1

Assignment of inputs and outputs:

Example: G24.N1.2.3.4.3.0



- Sign evaluation
- Input 3 is the first input for reading
- Output 4 is the first output for multiplex mode
- 3 digits before the decimal point
- 0 digit after the decimal point

States of the inputs E3 to E6 for the number -906 (with the corresponding control in multiplex operation):

	E3	E4	E5	E6
-	-	-	-	1
9	1	0	0	1
0	0	0	0	0
6	0	1	1	0

3.30 Reference travel

G25

Instruction form:

G25.A

Example:

```
Program : BEISPIEL
1    G25.X                    ;Reference travel is performed for the X axis
2    G25.Y                    ;Reference travel is performed for the Y axis
3    END

Programm : BEISPIEL
1    G90
2    G21 I5.0 REFEREN         ;Reference condition for Y axis satisfied?
3    G25.0 G91
4    X100                     ;Satisfy reference condition for Y axis!
5    G90
6    $REFEREN
7    G25.Y
8    G25.X
9    END
```

Applikation:

Reference travel must be performed after every restart, i.e. after a power failure or after the PA has been switched off. This is the only way of guaranteeing that the control executes its program in a defined manner.

Description:

The PA-CONTROL performs reference travel with the corresponding axis after start of instruction G25.A. The position reached after reference travel is considered as the datum for the axis.

Note:

The reference flag must be set by a reference travel (G25.X, G25.Y,...) or by instruction G25.0 before positioning can be performed with an axis. Instruction G25.0 does not change the position values of the axes.
The speed is taken from the parameters (reference speed) for the reference travel.
The parameter setting for the reference speed and the acceleration must be adapted to the overtravel (distance between reference switch and mechanical stop).

3.31 Set position to zero

G26

Instruction form:

G26.A

Example:

```
Program : BEISPIEL
1    G26.X                    ;Set absolute position of the X axis to zero
2    G26.Y                    ;Set absolute position of the Y axis to zero
3    END
```

Applikation:

A sensible application for this PA instruction is for continuous movements, e.g. use of the rotary table DT140 in only one direction.

Desciption:

The absolute position counter of each axis can be set to zero with function G26.A. The variable A of the operand indicates the selected axis.

Note:

The software limit switches defined by way of the range on the parameter level are taken over unchanged. If this instruction is repeated correspondingly, it is thus theoretically possible to perform infinite positioning in one direction, since no counter overflow and no transgression of the range limits is possible.
The remainder values for gear factors with digits after the decimal point not equal to zero are also cleared, i.e. they cannot be taken into account in subsequent positioning.

3.32 Set position to dimension

G29

Instruction form:

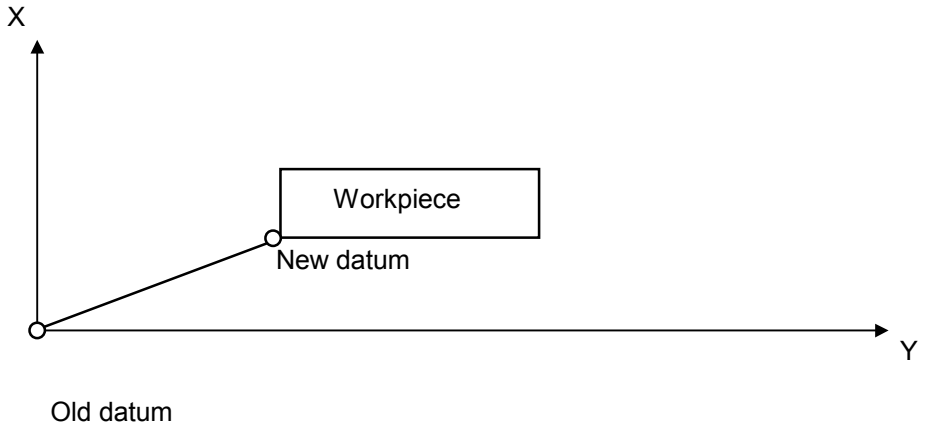
G29.Ai
G29.ARn

Example:

```
Program : BEISPIEL
1   G25.X          ;Reference travel X axis
2   G90            ;Absolute dimension system
3   X700           ;Position to position +700 of X axis
4   G29.X0         ;Set current position of the X axis to zero, datum position
                    of workpiece and axis system is the same
5   END
```

Application:

When using dimensions from drawings, it is possible to define a datum which corresponds to the drawing origin.



Description:

The position counter of each axis can be set to a new value with function G29. The operator must enter the selected axis. The software switches are automatically changed as well.

3.33 Absolute dimension system

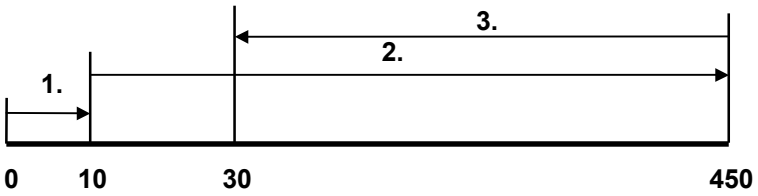
G90

Instruction form:

G90

Example:

```
Program : Beispiel
1    G25.X          ;Reference travel
2    G90            ;Following position instructions are executed in the abso-
                     lute dimension system
3    X10            ;Position 10 is approached with the X axis, travel = 10
                     mm
4    X450           ;Position 450 is approached with the X axis, travel = 440
                     mm
5    X30            ;Position 30 is approached with the X axis,
                     travel = -420 mm
6    END
```



[Skizze]

Application:

For dimensions which are all measured from a datum.

Description:

All subsequent positioning operations are performed in the absolute dimension system after instruction G90 is called until the system is switched over to incremental dimension system with instruction G91.

3.34 Incremental dimension system

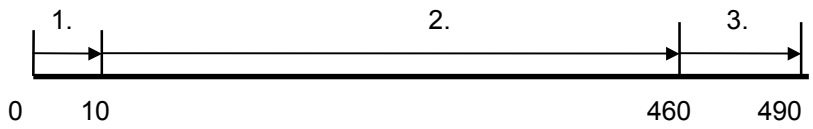
G91

Instruction form:

G91

Example:

```
Programm : Beispiel
1   G25.X          ;Reference travel
2   G91            ;Following position instructions are executed in the in-
                    cremental dimension system
3   X10            ;Position 10 is approached with the X axis, travel = 10
                    mm
4   X450           ;Position 460 is approached with the X axis, travel = 450
                    mm
5   X30            ;Position 490 is approached with the X axis, travel = 30
                    mm
6   END
```



Application:

-For dimensions which are not referred to a datum.
-For dimension intervals which occur several times in succession and can be repeated with a program loop (DEC.Ni).

Description:

All subsequent positioning operations are performed in the incremental dimension system after instruction G91 is called until the system is switched over to absolute dimension system with instruction G90.

3.35 Define acceleration

G100

Instruction form:

G100.A.i
G100.A.Rn
G100.B.i
G100.B.Rn

Example:

```
Program : Beispiel
1    G25.X                ;Reference travel X axis
2    G91                  ;Incremental dimension system
3    FX250
4    G100.X.15            ;Definition of acceleration for the X axis
5    X250                 ;Traversing instruction for the X axis
6    G100.X.20            ;Definition of acceleration for the X axis
7    X250                 ;Traversing instruction for the X axis
8    END
```

Application:

Using instruction G100, it is possible to obtain an acceleration value which differs from the values defined in the parameter values and which is adapted to the particular situation. The definition made here is valid until the next G100 instruction or until a restart. The stored parameter values are valid if instruction G100 is not used.

Description:

A stands for the axis designation and can have a value between 1 and the maximum number of axes in multi-axis systems. **G100.B** serves the employment for the track acceleration of interpolation.

Note:

An error message "Value too high" is issued for instruction G100.A.i if i is greater than the parameter value.

3.36 Residual Distance Positioning

G120

Command form

G120 In.m +Am
G120 In.m +ARi

Example:

```
Programm:BEISPIEL
1   G25.X G90                ;Return to datum with X axis and switch to absolute measurement system
2   G120 I3.1 X200           ;Activate residual distance positioning, as long as the entry 3 is set the X axis will move at the next positioning. If the entry is reset the residual distance will be positioned..
3   X3000                    ;Positioning when the residual distance positioning is active
4   ...
7   ...
8   R15:=25                  ;File residual distance in register 15
9   G120 I5.0 XR15           ;Activate residual distance positioning, as long as the entry 5 has not been set, the X axis will move at the next positioning. If the entry is set, the residual distance of 25 is positioned.
10  X300
11  G22 M256.0 SONDER        ;The entry did not change during the positioning, subroutine is called up
12  ...
13  END

Programme: SONDER
1   ...
2   ...
3   END
```

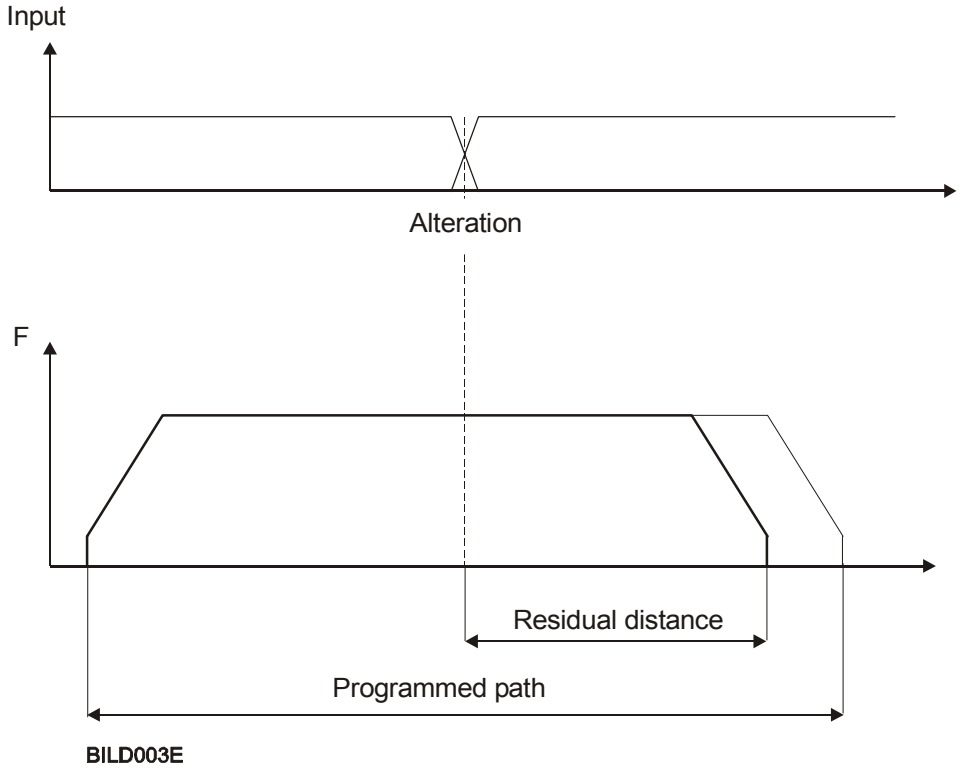
Application:

A part has to aligned when depositing. It is turned (rotation axis) and when an input signal is given (light barrier, sensor,..) it is turned a given number of degrees further (residual distance).

Description:

This command takes into consideration the defined input with the subsequent positioning process. If this input alters it status, the residual distance is then positioned.

Condition: - The positioning has not yet reached the braking stage
- The residual distance must be greater than the braking distance



Note:

Command is only available in the PAC-Single and/or PAC-SC compact!

The commands G23, G123 or G210 should not be simultaneously active.
The command G120 resets the flag 256. If all requirements have been met and if „altering the input status“ occurs during the positioning process, the flag 256 is set.

3.37 Traverse axis provided condition is fulfilled

G123

Instruction form:

G123.A In.m
G123.A Mn.m
G123.A On.m

Example:

```
Program: BEISPIEL
1   G25.X G90          ; Reference travel is performed with the X axis, switch
                       over to absolute dimension system
2   G123.X I3.1        ; The X axis is traversed during the next positioning op-
                       eration provided input 3 is set. Otherwise, the current po-
                       sitioning operation is aborted.
3   X100               ; Traverse axis X 100 in positive direction
4   SUB LOCHBOHR
5   END
```

Application:

A workpiece is traversed beneath a detection device until it is detected. The detection device then returns a signal to the PA so that the program can be continued.

Description:

With function G123, the next traverse instruction of the corresponding axis is executed provided condition n.m of the operand is valid. If the condition is not fulfilled, the traverse instruction is aborted and the next instruction is executed. G123 acts only on the axis defined in the command.

Note:

-If traverse commands are aborted, the current absolute position for the subsequent position operation is taken into consideration.
-The instruction group G210 (process beyond line limit) is permitted together with instruction G123.
-Only one G123 may be active per axis. Logic operations involving several conditions can be implemented via the instructions for logic operations.
-G23 and G123 may not be active simultaneously.
-Operators O and M are practical only in conjunction with the PAB.

3.38 Switch to Measuring Mode

G140

Command form:

G140.A

Example:

```
Programme:BEISPIEL
1   G25.X          ;Return to datum with X axis
2   X100           ;Move to position 100
3   I5.1           ;Wait until set-up mode is active, guard circuit bridged
4   G140.X         ;Switch X axis to measuring mode, axis can be moved
                    manually.
5   I5.0           ;Guard circuit is reactivated, set-up mode is blocked
6   G141.X         ;Switch X axis back to control mode
7   X1000          ;Move X axis to position 1000
8   END
```

Applicaton:

During set-up mode make the axis currentless.

A stands for axis name.

A = 0 : all axes
A = X : X axis
A = Y : Y axis
etc.

Description:

The command **G140** gives the programmer the opportunity to make the servo axes currentless during the set-up mode. The measuring mode however remains active so that the positions of the axes are not lost. After switching on control operation again it is not necessary to repeat a return to datum as the position measuring remains active.

Note:

This command is <u>only</u> available for the PAC-Servo.
When switching a vertical axis to the measuring mode this may slip down. Risk of collision !

3.39 Switch to Control Mode

G141

Command form:

G141.A

Example:

```
Programme: BEISPIEL
1    G25.X          ;Return to datum with X axis
2    X100           ;Move to position 100
3    I5.1           ;Wait until set-up mode is active, guard circuit bridged
4    G140.X         ;Switch X axis to measuring mode, axis can be moved
                    manually.
5    I5.0           ;Guard circuit is reactivated, set-up mode is blocked
6    G141.X         ;Switch X axis to control mode again
7    X1000          ;Move X axis to position 1000
8    END
```

Application:

Make the axis currentless during the set-up mode.

A stands for axis name.

A = 0 : all axes
A = X : X axis
A = Y : Y axis
etc.

Description:

The command **G141** gives the programmer the opportunity of switching a servo axis that has been made currentless with **G140** back to control operation. The position altered in measuring mode is however retained. A repeat return to datum is not necessary after switching on control operation again.

Note:

This command is only available for the PAC-Servo.
When switching on control operation again there is a risk of collision due to a possible change in position!

3.40 Traverse part-distance with Start-Stop

G150

Instruction form:

G150. An
G150. ARi

Example:

```
Program: BEISPIEL
1   G25.X G91          ; Perform reference travel with X axis, switch over to incremental dimension system
2   G150.X200          ; Traverse the last 200 increments at Start-Stop speed during the next positioning operation with the X axis.
3   X1000              ; Position 1000 increments with the X axis. The deceleration process is started so that Start-Stop speed is reached 200 increments before the destination position. The last 200 increments are then positioned at the Start-Stop speed.
4   END
```

Application:

For positioning operations in which the last part-distance is positioned in Start-Stop mode.

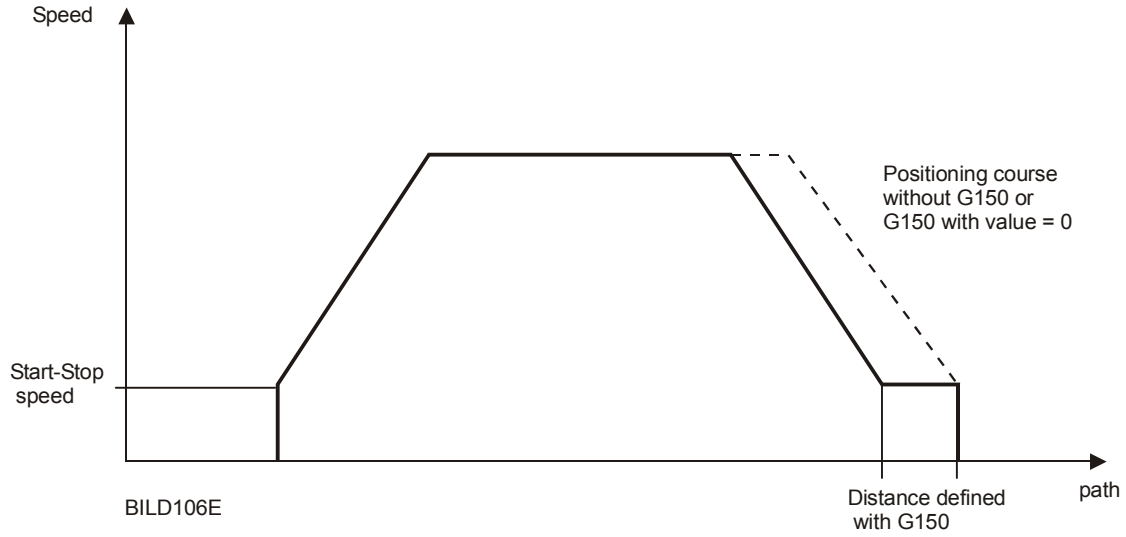
Examples: Joining operation
faster response to G23 or G123 instructions

Description:

Instruction G150 provides the programmer with the option of starting the deceleration operation at an earlier point during the next positioning operation with the corresponding axis and positioning the distance defined with G150 using the Start-Stop speed. The transmission from the deceleration ramp to the Start-Stop speed (distance) is performed continuously.

Note:

This command is not available with the positioning control version.



Note:

-The number after the axis name in instruction G150 is a distance and is interpreted as an unsigned quantity and direction-independently.
-The instruction is allowed for only during the next positioning operation for this axis.
-If the distance to be traversed during the next positioning operation is less than the distance from instruction G150, instruction G150 is ignored and the positioning operation runs as normal.

3.41 Commands of the G2?? - Group

G2??

Introduction

With the commands of this group complex machine functions are possible as parallel cycles can be realised easily.

Without the commands of the G2?? - group the following applies:

The programmes are processed in lines, command on command. With positioning processes the execution of the programme is only continued when the process has been completed.

Example 1:

```
1 G90                ;Absolute system of measuring
2 G25.X              ;Return to datum of X axis
3 X1000              ;Positioning process
4 O1:=1              ;Set output 1
5 I1.1               ;Wait until input 1 at 1
6 O1:=0              ;Reset output 1
7 END
```

In example 1 the cycle in line 3 remains stationary for the period of the positioning process. The output 1 is set upon completion of the positioning process when input 1 is set.

Using the G2?? commands

Example 2:

```
1 G90                ;Absolute system of measuring
2 G25.X              ;Return to datum of X axis
3 G210               ;Processed commands beyond the line limit, do not wait
                     until
                     ;the end of the positioning processes
4 X1000              ;Positioning process
5 O1:=1              ;Set output 1
6 I1.1               ;Wait until input 1 at "1"
7 O1:=0              ;Reset output 1
8 G213               ;Wait until all positioning processes are completed,
                     ;Re-cancel G210 function
9 END
```

Output 1 is set immediately after starting the positioning process. Line 6 is then processed although the positioning process has not yet been completed and input signals can be reacted to for example during the positioning process.
Virtually all commands initiated by the command G210 are permissible in this operating status. „Restarting“ an axis that is already undergoing positioning is not possible but another axis or axes can be started that have completed the previous positioning process. The commands of the G2?? group also offer simple inquiry possibilities as regards the positioning processes.

3.41.1 Process instruction beyond line limit

G210

Instruktion form:

G210

Example:

See the next few instructions

Application:

Program parts in which input and output functions are to be operated during positioning.

Description:

Instruction **G210** starts a so-called "process beyond line limits" in the PA-CONTROL.

This means:

- if several positioning tasks of different axes are to be executed in several lines in direct succession (with no other instructions in between), these positioning instructions are all started simultaneously.

- The positioning tasks are only started and the processing of the following commands is continued (until G213 has been reached).
  This means that further commands (I/O processing) can be carried out whilst the axes are being positioned.

3.41.2 Position-related jump

G211

Instruction form:

G211.A.j Label

Application:

Execution of a jump dependent on whether the selected axis or all axes has/have reached its/their end position.

A stands for the axis number and j stands for the condition.

The following applies:

A = 0 : all axes,
A = 1 : X axis,
A = 2 : Y axis,
A = 3 : Z axis,
usw.

j = 0 : Jump if position of selected axes has not yet been reached
j = 1 : Jump if position of selected axes has been reached

Note:

The instruction evaluates only the end position
No further instruction may be contained in this line after G211.A.j

Example 1:

```
Program : BEISPIEL
1    G25.X G25.Y              ;Reference travel of the X and Y axes
2    G90                      ;Absolute dimension system
3    G210                     ;Process instructions beyond line limit start
4    X10000 Y6000             ;Axes start
5    $SETZE
6    O1:=1 T50 O1:=0          ;Set output 1 for 500 ms
8    G211.0.0 SETZE           ;Jump to label "SETZEN" if all axes have not yet reached
                              their position
9    G213                     ;Wait until all axes are in position (cancel the G210 func-
                              tion)
10   O2:=1
11   END
```

Example 2:

```
Program : BEISPIEL
1    G25.X G25.Y              ;Reference travel of the X and Y axes
2    G90                      ;Absolute dimension system
3    G210                     ;Process instructions beyond line limit start
4    X10000 Y6000             ;Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0          ;Set output 1 for 500 ms
7    T50                      ;Wait 500 ms
8    G211.0.1 FERTIG          ;Jump to label "FERTIG" if all axes have reached their
                              position
9    JMP SCHLEIFE             ;Jump to "SCHLEIFE"
10   $FERTIG
11   G213                     ;Wait until all axes are in position (cancel the G210 func-
                              tion)
12   O2:=1
13   END
```

Example 3:

```
Program : BEISPIEL
1    G25.X G25.Y              ;Reference travel of the X and Y axes
2    G90                      ;Absolute dimension system
3    G210                     ;Process instructions beyond line limit start
4    X10000 Y6000             ;Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0          ;Set output 1 for 500 ms
7    T50                      ;Wait 500 ms
8    G211.Y.1 FERTIG          ;Jump to label "FERTIG" if the Y axis has reached its po-
                              sition
9    JMP SCHLEIFE             ;Jump to "SCHLEIFE"
10   $FERTIG
11   G213                     ;Wait until all axes are in position (cancel the G210 func-
                              tion)
12   O2:=1
13   END
```

Example 4:

```
Program : BEISPIEL
1    G25.X G25.Y              ;Reference travel of the X and Y axes
2    G90                      ;Absolute dimension system
3    G210                     ;Process instructions beyond line limit start
4    X10000 Y6000             ;Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0          ;Set output 1 for 500 ms
7    T50                      ;Wait 500 ms
8    G211.X.0 SCHLEIFE        ;Jump to label "SCHLEIFE" if the X axis has not yet rea-
                              ched its position
9    G213                     ;Wait until all axes are in position (cancel the G210 func-
                              tion)
10   O2:=1
11   END
```

3.41.3 Position-related subroutine call

G212

Instruction form:

G212.A.j Name

Description:

Execution of a subroutine call depending on whether the selected axis or all axes has/have reached its/their end position.

A stands for the axis name and j stands for the condition.

The following applies:

A = 0	: all axes,
A = X	: X axis,
A = Y	: Y axis,
A = Z	: Z axis,
etc.	
j = 0	: Jump to subroutine if position of selected axes has not yet been reached
j = 1	: Jump to subroutine if position of selected axis has been reached

Note:

The instruction evaluates only the end position
No further instruction may be contained in this line after G212.A.j

Example 1.

```
Program : BEISPIEL
1    G25.X G25.Y          ;Reference travel of the X and Y axes
2    G90                  ;Absolute dimension system
3    G210                 ;Process instructions beyond line limit start
4    X10000 Y6000         ;Axes start
5    T100                 ;1 s waiting time
6    G212.0.0 AUSGSET     ;Jump to subroutine if all axes have not yet reached their
                          position
7    G213                 ;Wait until all axes are in position (cancel the G210 func-
                          tion)
8    O2:=1
9    END

Program : AUSGSET
1    O9:=1 T50 O9:=0
2    END
```

Example 2:

```
Program : BEISPIEL
1    G25.X G25.Y          ;Reference travel of the X and Y axes
2    G90                  ;Absolute dimension system
3    G210                 ;Process instructions beyond line limit start
4    X10000 Y6000         ;Axes start
5    T100                 ;1 s waiting time
6    G212.0.1 AUSGSET     ;Jump to subroutine when all axes have reached their po-
                          sition
7    G213                 ;Wait until all axes are in position (cancel the G210 func-
                          tion)
8    O2:=1
9    END

Program : AUSGSET
1    O9:=1 I7.1 T50 O9:=0
2    END
```

Example 3:

```
Program : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ;Axes start
5    T100                       ;1 s waiting time
6    G212.X.0 AUSGSET           ;Jump to subroutine if the X axis has not yet reached its
                                position
7    G213                       ;Wait until all axes are in position (cancel the G210 func-
                                tion)
8    O2:=1
9    END

Program : AUSGSET
1    O9:=1 I7.1 T50 O9:=0
2        END
```

Example 4:

```
Program : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ;Axes start
5    T100                       ;1 s waiting time
6    G212.Y.1 AUSGSET           ;Jump to subroutine when the Y axis has reached its po-
                                sition
7    G213                       ;Wait until all axes are in position (cancel the G210 func-
                                tion)
8    O2:=1
9    END

Program : AUSGSET
1    O9:=1 I7.1 T50 O9:=0
2    END
```

3.41.4 Wait until all axes in position

G213

Instruction form:

G213

Example:

```
Program : BEISPIEL
1    G210                    ;Process beyond line limit,
2    X1234.56 Y399.10        ;Start positioning for X and Y axes
3    O1:=2 T10 O1:=0
4    G213                    ;Wait until all axes are in position, reset function of in-
                              struction G210
5    END
```

Application:

There are two reasons for using instruction G213:

-Cancellation of instruction G210
-Waiting until all moving axes have reached their destination position

Description:

Processing of instructions is cancelled during positioning with instruction G213 (G213 = Reset G210. In order to achieve this, the system waits, when instruction G213 is reached, until all moving axes have reached their destination position.

3.41.5 Position-related jump (current position)

G221

Instruction form:

G221.j.An.Label
G221.j.ARn.Name

Application:

A conditional jump can be realized as a function of a current axis position by using instruction G221.0.

j stands for jump condition, An stands for the axis and the position and label stands for the destination to which the jump is to take place.

The following applies:

j = 0	Jump if the position (An) is less than the current axis position
j = 1	Jump if the position (An) is greater than the current axis position

Note:

This instruction acts on the absolute position of the specific axis irrespective of selected dimension system (G90/G91)
This line may not contain any further instruction after G221.

Eample 1:

```
Program : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.1.X8000 ROT           ;Jump "ROT" if the X position is greater than 8000
8    O2:=1 T10 O2:=0
9    $ROT
10   G221.0.0 SCHLEIFE          ;Jump to "SCHLEIFE" if all axes are not yet in position
11   G213
12   END
```

Example 2:

```
Program : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.1.Y2000 ROT           ;Jump to "ROT" if the Y position is greater than 2000
8    O2:=1 T10 O2:=0
9    $ROT
10   G211.0.0 SCHLEIFE          ;Jump to "SCHLEIFE" if all axes are not yet in position
11   G213
12   END
```

Example 3:

```
Program : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.0.X8000 ROT           ;Jump to "ROT" if the X position is smaller than 8000
8    O2:=1 T10 O2:=0
9    $ROT
10   G211.0.0 SCHLEIFE          ;Jump to "SCHLEIFE" if all axes are not yet in position
11   G213
12   END
```

3.41.6 Position-related subroutine call (current position)

G222

Instruction form:

G222.j.An Name

Applikation:

A conditional subroutine call can be realized as a function of a current axis position by using instruction G222.

j stands for the jump condition, An stands for the position of the axis and Name stands for the subroutine to be called

The following applies:

j = 0	Jump if the position (An) is less than the current axis position.
j = 1	Jump if the position (An) is greater than the current axis position

Note:

This instruction acts on the absolute position of the specific axis irrespective of selected dimension system (G90/G91)

Example 1:

```
Programm : BEISPIEL
1    G25.X G25.Y                ;Reference travel of the X and Y axes
2    G90                        ;Absolute dimension system
3    G210                       ;Process instructions beyond line limit start
4    X10000 Y6000               ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.1.X8000 ROT           ;Call subroutine "ROT" if the X position is greater than
                                8000
8    G211.0.0 SCHLEIFE          ;Jump "SCHLEIFE" if all axes have not yet reached their
                                position
9    G213
10   END

Programm : ROT
1    O2:=1 T10 O2:=0
2    END
```

Example 2:

```
Program : BEISPIEL                ;Reference travel of the X and Y axes
2    G90                          ;Absolute dimension system
3    G210                         ;Process instructions beyond line limit start
4    X10000 Y6000                 ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.1.Y1000 ROT             ;Call subroutine "ROT" if the Y position is greater than
                                  1000
8    G211.0.0 SCHLEIFE            ;Jump to "SCHLEIFE" if all axes have not yet reached
                                  their position
9    G213
10   END

Program : ROT
1    O2:=1 T10 O2:=0
2    END
```

Example 3:

```
Program : BEISPIEL
1    G25.X G25.Y                  ;Reference travel of the X and Y axes
2    G90                          ;Absolute dimension system
3    G210                         ;Process instructions beyond line limit start
4    X10000 Y6000                 ; Axes start
5    $SCHLEIFE
6    O1:=1 T50 O1:=0 T50
7    G221.0.X4000 ROT             ;Call subroutine "ROT" if the X position is smaller than
                                  4000
8    G211.0.0 SCHLEIFE            ;Jump to "SCHLEIFE" if all axes have not yet reached
                                  their position
9    G213
10   END

Program : ROT
1    O2:=1 T10 O2:=0
2    END
```

3.41.7 Wait until current psition < / > as value

G230

Instruction form:

G230.j.An

Eample:

```
Program : BEISPIEL
1   G25.X                    ;Reference travel of the X axe
2   G90                      ;Absolute dimension system
3   G210                     ;Process instructions beyond line limit start
4   X10000                   ;Start der Achse
5   O1:=1 T10 O1:=0
6   G230.1.X3000 O2:=1       ;Output 2 is set from X3000 to X5000
7   G230.1.X5000 O2:=0
8   O1:=1 T10 O1:=0
9   G213
10  G210                     ;Process instructions beyond line limit start
11  X0
12  O1:=1
13  G230.0.X3000             ;Output 1 is set from X10000 to X3000
14  O1:=0
15  G213
16  END
```

Application:

By using instruction G230, it is possible to suspend further execution of the PA program until a certain position is reached.

j stands for the condition and An stands for the axis and the position.

The following applies:

j = 0	Wait until position is less than current position
j = 1	Wait until position is greater than current position

Note:

This instruction acts on the absolute position of the specific axis irrespective of the selected dimension system (G90/G91)

3.42 Time monitoring instructions

G4??

Introduction

Time monitoring **instructions** of the PA family

The instructions described below are primarily suitable if operations have to be monitored with respect to a defined end and if auxiliary programs have to be started to remedy or record errors in the event of irregularities. These situations can always occur if sequences are started by the controls, e.g. soldering programs, which are executed in parallel and independently of the programs of the PA.

After the corresponding time has elapsed, a conditional jump or subroutine call can take place, depending on the respective instruction. Set time conditions can be reset or aborted. The time condition is checked before each instruction executed by the PA and a reaction initiated where appropriate.

Note:

With „Programme after Stop“ a time monitoring that is still active is reset by the operating system without further signalization and not reactivated with a start (see also Chapter 2.8 Cycle Definitions).

3.42.1 with conditional jump

G421.1.

Instruction form:

G421.1.nnn Marke

Example:

```
Program : BEISPIEL
1    $ANFANG
2    O1:=1                      ;Set output 1
3    G421.1.200 FEHLER_1        ;Jump to label ‹FEHLER_1" if 2 seconds have elapsed
                                and the time condition has not been reset
4    I1.1 O1:=0                 ;Wait for input 1 = 1 and reset output 1
5    G401.1                     ;Reset the time condition
6    END
7
8    $FEHLER_1
9    O2:=1                      ;Set fault signal
10   I2.1                       ;Wait for error acknowledgment signal
11   O1:=0 O2:=0                ;Reset output 1 and fault signal
12   JMP  ANFANG
```

Description:

The monitoring time is started with instruction G421.1.200 Label. The monitoring function is reset if the reset instruction G401.1 is recognized before expiry of the
2-second monitoring time (200*10 ms). If this does not occur, the program is continued at the specified label.

A G421 running in the program is aborted by a further G421 or G422 and, at the same time, the new time monitoring instruction is started. The remaining time of the first monitoring task is not further processed.

3.42.2 with conditional subroutine call

G422.1

Instruction form:

G422.i.nnn Name

Example:

```
Program : BEISPIEL
1    O1:=1                    ;Set output 1
2    G422.1.200 FEHLER        ;Subroutine call if 2 seconds have elapsed and the time
                               condition has not been reset.
3    I1.1 I1.1 O1:=0          ;Wait for input 1 = 1 and reset output 1
4    G401.1
5    END

Program : FEHLER
1    O3:=1                    ;Fault at ...
2    I5.1                     ;Acknowledgment by operator
3    END
```

Description:

The monitoring time is started with instruction G422.1.200 Name. The monitoring function is reset if the reset instruction G401.1 is recognized before expiry of the
2-second monitoring time (200*10 ms). If this does not occur, the jump is made to the specified subroutine.

Important:

In the example, instruction "I1.1" has been written twice intentionally (3rd program line: I1.1 I1.1 01:=0). The reason is that the return from the time monitoring subroutine does not occur as the result of the PA instruction (when the time had elapsed) which was interrupted by time monitoring but in response to the following instruction.

Example:

The PAC is in the 3rd program line at the first I1.1 instruction and waits for the input, time monitoring is activated, subroutine "FEHLER" is called and is run. The return from the subroutine then occurs at the second I1.1 instruction in the 3rd program line.

3.42.3 Reset time condition

G401.1

Instruction form:

G401.1

Example:

Example, see instruction G421 or G422

Description:

The time monitoring function which was started by instructions G421 or G422 is stopped by this instruction. The time monitoring function is then no longer active.

3.43 Text and value output, value transfer

G5??

Introduction:

This command group enables texts, register contents, input, output and marker states to be output via the current data channel during the program sequence (automatic operation).

The following data channels are possible:

- LCD display on the PA-Control front panel
- Serial interface 1 and 2

The current data channel can be selected with the command **"G500."**.
Initialization (baud rate, number of data bits, parity, ...) of the selected serial interface also takes place via the command **"G500."**.

Please note:

The desired data channel (with initialization in the case of a serial interface) should be defined with the command "G500." at the start of every program in which the commands of the G5xx group are used. Otherwise, it cannot be guaranteed that all operations take place via the data channel currently defined in the parameters.

The following applies only to the equipment with CPU3:

From the software version 3.20 upwards the RS485 channel at interface 2 can be used as an alternative to the RS232 channel. For this purpose the jumper on the CPU3 at J12 should be removed and inserted at J11 (see chapter "Technical Appendix“ -> "CPU3“).

The RS485 signals are on PIN8 = B and PIN9 =A. After running through the switch-on functions of the PAC the transmitter operates at high resistance. A programme example for the coupling of several PACs based on the master-slave-principle can be found on the demo disk.

Note:

All PACs used must be equipped at least with the system software 3.2.

3.43.1 Interface initialization

G500

Instruction form.

G500.Nr.Baudrate.Dataformat.Handshake
G500.0

Example:

Programm:BEISPIEL
See examples of the following commands

Description:

G500. switches between the two serial interfaces and the LC display of the PA-Control front panel Only the No.: 0 is necessary after the command G500. when the LC display is selected. All other parameters are omitted.
If a serial interface is selected as a data channel, it is initialized in accordance with the transferred parameters [refer to the list for the meaning of the transfer parameters].

Please note:

The desired data channel should be defined at the start of every program!
Only details without CR-LF are meaningful for output on the LC display of the PA-Control.

Significance of the transfer parameters:

No:

0 : LC-Display is activated
1 : Interface 1 is activated
2 : Interface 2 is activated*

Baud rate:

1 : 110 baud
2 : 300 baud
3 : 1200 baud
4 : 2400 baud
5 : 4800 baud
6 : 9600 baud
7 : 19200 baud
8 : 38400 baud (currently not yet available)

Data format:

1 : 8 data bits, 1 stop bit, no parity
2 : 7 data bits, 1 stop bit, no parity*
3 : 7 data bits, 1 stop bit, even parity
4 : 7 data bits, 1 stop bit, odd parity

Handshake :

0 : Hardware handshake (CTS) inactive
1 : Hardware handshake (CTS) active

* Not possible with the PAC-SC Compact and the PAC-Single

3.43.2 Delete the display

G501

Instruction form:

G501

Example:

```
Program: EXAMPLE
1   G11.0          ;Switch off sequence displays
2   G500.0         ;Switches the G500 commands to the current display
                    medium (LC display or, in the case of "Simulation of the
                    front panel", the serial interface)
3   G501           ;Deletes the display and sets the cursor in the top left
                    corner (1st column, 1st line)
4   G510.Hello     ;Outputs the text "Hello" on the current display medium.
5   END
```

Description:

The command G501 deletes the display of the current display medium and sets the cursor in the top left corner.

Note:

Before the command G501 is used, switchover to the current display medium should be performed with the command "G500.".

3.43.3 Delete up to line end

G502

Instruction form:

G502

Example:

```
Program : EXAMPLE
1   G11.0          ;Switch off sequence displays
2   G500.0         ;Switches the G500 commands to the current display
                    medium (LC display or, in the case of "Simulation of the
                    front panel", the serial interface)
3   G501           ;Deletes the display and sets the cursor in the top left
                    corner (1st column, 1st line).
4   N1:=234
5   $LOOP
6   G503.1.2       ;Position the cursor in the first column of the 2nd line
7   G510.VALUE N1
8   G503.11.2      ;Position the cursor in the 11th column of the 2nd line
9   G502           ;Delete as from the cursor position up to the end of the
                    line
10  G520.N1        ;Output the value of N1 at the display (flush left)
11  T100
12  DEC.N1 LOOP
13  END
```

Description:

The command G502 deletes the display on the display medium from the current cursor position to the end of the line. The position of the cursor remains unchanged.

The command can be used to delete the previously displayed value when register values are output so as to then output the new register value.

Please note:

Before using the command G502, switchover to the current display medium should be performed with the command "G500.".

3.43.4 Position the cursor

G503

Instruction form:

G503.n.m
G503.Nn.m
G503.n.Nm
G503.Nn.Nm

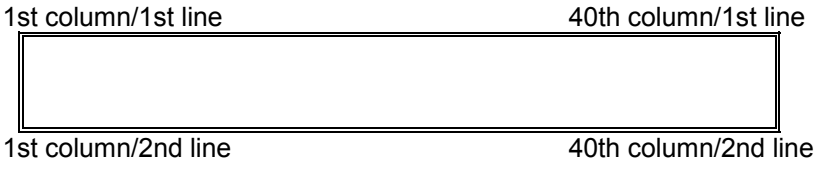
Example

```
Program : EXAMPLE
1   G11.0                ;Switch off sequence displays
2   G500.0               ;Switches the G500 commands to the current display
                          medium (LC display or, in the case of "Simulation of the
                          front panel", the serial interface)
3   N23:=5
4   N35:=2
5   G501                 ;Deletes the display and sets the cursor in the top left
                          corner (1st column, 1st line)
6   G503.4.1             ;Sets the cursor in the 4th column of the 1st line
7   G510.HELLO           ;Outputs the text "HELLO" at the current cursor position
                          on the current display medium
8   G503.N23.N35         ;Sets the cursor in the 5th column of the 2nd line. See
                          values of N23 and N35
9   G510.OPERATOR        ;Outputs the text "OPERATOR" at the current cursor po-
                          sition on the current display medium
10  END
```

Description:

The command G503 serves to position the cursor on the display.

The LCD display is divided up into 40 columns and 2 lines.



Please note:

Before using the command G503, switchover to the current display medium should be carried out with the command "G500.".

3.43.5 Text output

G510.

Instruction form:

G510. text

Example:

Program : BEISPIEL

```
1   G500.2.6.1.0                ;Initializes the selected interface
2   G510.Maschinenfehler !      ;Outputs the text "Machine error!" via the serial interface
3   END
```

Description:

G510. outputs the text via the selected serial interface.

Note:

No CR and no LF are sent at the end of the text.
There may be no other instruction in this line after G510. The text starts after "G510." and ends at the end of the line (terminate entry with ENTER in the program editor)

3.43.6 Text output

G511.

Instruction form:

G511. text

Example:

```
Program : BEISPIEL
1   G500.2.6.1.0                ;Initializes the selected interface
2   G511.Maschinenfehler !      ;Outputs the text "Machine error!" via the serial interface
                                 and terminates the transmission with CR and LF
3   END
```

Description:

G511. outputs the text via the selected serial interface and sends a CR and LF after the text.

Note:

The transmission is terminated with CR and LF.
There may be no other instruction in this line after G511. The text starts after "G511" and ends at the end of the line (terminate entry with ENTER in the program editor)

3.43.7 Control character output

G512.

Instruction form:

G512. n

Example:

```
Program : BEISPIEL
1    G500.2.6.1.0      ;Initializes the selected interface
2    G512.7            ;Outputs command "BEL" via serial interface.
3    G511.36           ;Outputs character "$" via serial interface.
4    G511.4            ;Outputs 04Hex (EOT) via serial interface.
5    END
```

Description:

G512. interprets n as the nth character (ordinal) of the ASCII table and outputs the character via the serial interface.
All numbers between 0 and 255 are permitted.

Application:

Output of control characters to printer. Please refer to your printer's ASCII table for the significance of the individual characters.

Note:

No CR and no LF are sent after the character.

3.43.8 Text Output

G515.

Command form:

G515.i.Name
G515.Ni.Name

Example:

Programme : MELDUNG.PTX
1 Machine ready
2 Fault at the part feed conveyor
3 Part magazine empty
4 Text not defined

Programme :BEISPIEL :PNC

```
1   G500.0                  ;Initialization, LC-Display is current data channel
2   G515.2.MELDUNG          ;outputs the second line of the programme
                            „MELDUNG.PTX“ on the current data channel
3   N5:=3
4   G515.N5.MELDUNG         ;outputs the third line of the programme
                            „MELDUNG.PTX“ on the current data channel
5   END
```

Description:

G515 outputs a text (line) that is filed in a programme file (*.PTX) on the current data channel. With this function different control sequences can be output to the printer and similar devices that otherwise require several G512 commands each.

Note:

The file must be a *.PTX type file and available when the programme is running. If the respective text line cannot be found (number is greater than the number of lines in the file) the last programme line is specified.

3.43.9 Value output

G520.

Instruction form:

G520.	Rn
G520.	Nn
G520.	In
G520.	On
G520.	Mn

Example:

```
Program : BEISPIEL
1   G500.2.6.1.0    ;Initializes the selected interface
2   G520.R2         ;Outputs the contents of R2 via the current serial interface
3   G520.N5         ;Outputs the contents of N5 via the current serial interface
4   G520.I7         ;Outputs the state of input 7 via the current serial inter-
                     face
5   G520.O12        ;Outputs the state of output 12 via the current serial inter-
                     face
6   G520.M25        ;Outputs the state of flag 25 via the current serial inter-
                     face
7   END
```

Description:

G520. determines the state of the operand (register, input, output, flag) and outputs the corresponding value via the current serial interface.

Operand	State	Output(ASCII character)
Input	Energized	1
Input	Not energized	0
Output	Set	1
Flag	Set	1
Flag	Reset	0
Real number register	546.06	546.06
Integer register	34	34

Note:

No CR and no LF are sent via output.

3.43.10 Value output

G521.

Instruction form:

G521.	Rn
G521.	Nn
G521.	In
G521.	On
G521.	Mn

Example:

```
Program : BEISPIEL
1   G500.2.6.1.0    ;Initalizes the selected interface
2   G521.R2         ;Outputs the contents of R2 via the current serial interface
3   G521.N5         ;Outputs the contents of N5 via the current serial interface
4   G521.I7         ;Outputs the state of input 7 via the current serial inter-
                     face
5   G521.O12        ;Outputs the state of output 12 via the current serial inter-
                     face
6   G520.M25        ;Outputs the state of flag 25 via the current serial inter-
                     face
7   END
```

Description:

G521. determines the state of the operand (register, input, output, flag) and outputs the corresponding value via the current serial interface.

Operand	State	Output(ASCII character)
Input	Energized	1
Input	Not energized	0
Output	Set	1
Flag	Set	1
Flag	Reset	0
Real number register	546.09	546.09
Integer register	34	34

Note:

CR and LF are also sent, besides the actual value.

3.44 Value acceptance from the current data channel

G5??,...

Introduction:

This command group allows registers, output states and marker states to be read in via the current data channel during the program sequence (automatic operation).

The following data channels are possible:

- Keyboard on the PA-Control front panel
- Serial interfaces 1 and 2

The current data channel can be selected with the command "**G500.**".
In addition, the selected serial interface is initialized (baud rate, number of data bits, parity, ...) via the command "**G500.**".

Please note:

The desired data channel (with initialization in the case of a serial interface) should be defined with the command "G500." at the start of every program. Otherwise, it cannot be guaranteed that all operations take place via the data channel defined in the parameters as current.

3.44.1 Value transfer

G531.

Ionstruction form:

G531.Rn	Label
G531.Nn	Label
G531.On	Label
G531.Mn	Label

Example.

```
Program : BEISPIEL
1   G500.2.6.1.0          ;Initializes the selected interface
2   G531.R2 FEHLER        ;Waits for reception of a message at the current serial in-
                          terface and assigns the information to register R2; the
                          program is continued at label "FEHLER" in the event of a
                          transmission error
3   G531.M3 FEHLER        ;Waits for reception of a message at the current serial in-
                          terface and sets flag 31 on the basis of the information
4   JMP ENDE
5   $FEHLER               ;Error message program part
6   ...
7   $ENDE
8   END
```

Description:

G531. waits for reception of characters at the current serial interface. When CR is received, transmission is considered as having been completed and the received characters are further processed in accordance with the operand.
A time monitoring function is not implemented in the G531 instruction and can be realized with instructions G401, G421 and G422, if required.
The program is continued in the label if a transmission error (parity error, ...) is established during reception or if an invalid character string is received.
The program is aborted and the corresponding message is shown on the display if the "Stop key" is recognized.

Operand	Received ASCII characters	Action/State
Output	1	Set output
Output	0	Reset output
Flag	1	Set flag
Flag	0	Reset flag
Real number register	546.33	546.33
Integer register	25	25

Please note:

No further command may be in this line after G531.

Output of a log on a printer:

Preconditions:

-Printer connected to serial interface 2 with setting: 300 baud, 7 data bits, 1 stop bit, even parity, no hardware handshake
-Integer register 10 number of good parts
-Integer register 11 number of defective parts

```
Program : PROTOKO
1    G500.2.2.3.0                ;Initialization of interface
2    G511.Production log         ;Output title
3    G512.10                     ;Insert blank line
4    G510.Machine group: 1234
5    G511. System: Test
6    G512.10                     ;Insert blank line
7    G510.Number of good parts:
8    G521.R10
9    G510.Number of defective parts:
10   G521.R11
11   G512.10
12   END
```

3.44.2 Character transfer in the buffer

G532.

Instruction form:

G532.n Label

Example:

```
Program : EXAMPLE
1   G500.2.6.1.0              ;Initialization of serial interface 2
2   G532.13 U_ERROR           ;Receives characters at the serial interface and stores
                              them in the character buffer until the end criterion 13 =
                              0D hex = CR is received
3   N3:=COPY.2.5 C_ERROR      ;As from the 2nd character, convert the next 5 characters
                              from the character buffer into a number and store this
                              number in the integer register 3.
4   JMP END
5   $U_ERROR
6   G24.N1.0; TRANSFER ERROR
7   JMP END
8   $C_ERROR
9   G24.N1.0; CONVERSION ERROR
10  $END
11  END
```

Description:

The command G532.n takes characters (max. 80) from the current serial interface until the end criterion "n" has been received or until a fault occurs, and stores these characters in the character buffer. The end criterion "n" is a number greater than 0 and less than 255 and represents the corresponding ASCII character. If an error occurs during transfer of the characters, the transfer is aborted and the program is continued at the marker.

Error possibilities:

- Parity error, Overrun error, Framing error in character transfer
- Character buffer full

Please note:

The program remains in line 2 until the end criterion is recognized or until a transfer error occurs.

3.44.3 Character transfer in the background into the buffer

G533.

Instruction form:

G533.n

Example

```
Program: EXAMPLE
1   G500.2.6.1.0                ;Initialization of the serial interface 2
2   G533.13                     :Receives characters in the background at the serial inter-
                                face and stores them in the character buffer until the end
                                criterion 13 = 0D hex = CR is received
3   $LOOP
4   SUB WORK
5   N1:=CHN                     ;Character string completely received ?
6   CASE.JMP.N1                 ;Evaluation of the transfer
7   (ZEI_DA)                    ;End criterion was received
8   (U_ERROR)                   ;Error during transfer of a character
9   (P_FULL)                    ;Character buffer is full and the end criterion was not yet
                                received
10  ELSE LOOP                   ;End criterion was not yet received
11  $ZEI_DA
12  N3:=COPY.2.5 C_ERROR        ;From the 2nd character, convert the next 5 characters
                                from the character buffer into a number and store this
                                number in the integer register 3
13  JMP END
14  $U_ERROR
15  G24.N2.0; TRANSFER ERROR
16  JMP END
17  $P_FULL
18  G24.N2.0; BUFFER FULL
19  $END
20  END
```

Description:

The command G533.n accepts characters in the background (max. 80) from the current serial interface until the end criterion n" is received or until an error has occurred, and stores these characters in the character buffer.
The end criterion "n" is a number greater than 0 and less than 255 and represents the corresponding ASCII character.

The addition "in the background" means that the PA-CONTROL starts a system routine which accepts characters and the next command of the PNC program is then executed.

The command NI:=CHN allows you to check whether the character string was completely received.
If an error occurs during transfer of the characters, the transfer is aborted and the program is continued at the marker.

Error possibilities :

- Parity error, Overrun error, Framing error during character transfer
- Character buffer full

Note:

When making communication via the serial interface a loss of information can occur under certain circumstances.

```
G510.Please send!
...                    ;Attention: Send to communications partner immediately
...                     possible data loss !
G533.13                ;From now on the PAC is ready to receive

Better solution:
G533.13                ;From now on the PAC is ready to receive
...
G510.Please send!      ;PAC is already ready to receive
```

3.44.4 Check whether character transfer in the background has been completed

CHN

Instruction form:

Nn:=CHN

Example:

```
Program: EXAMPLE
1   G500.2.6.1.0                ;Initialization of serial interface 2
2   G533.13                     ;Receives characters in the background at the serial inter-
                                face and stores them in the character buffer until the end
                                criterion 13 = 0D hex = CR is received
3   $LOOP
4   SUB WORK
5   N1:=CHN                     ;Character string completely received ?
6   CASE.JMP.N1                 ;Evaluation of the transfer
7   (ZEI_DA)                    ;End criterion was received
8   (U_ERROR)                   ;Error during transfer of a character
9   (P_FULL)                    ;Character buffer is full and the end criterion has not yet
                                been received
10  ELSE LOOP                   ;End criterion has not yet been received
11  $ZEI_DA
12  N3:=COPY.2.5 C_ERROR        ;From the 2nd character, convert the next 5 characters
                                from the character buffer into a number and store this
                                number in the integer register 3.
13  JMP END
14  $U_ERROR
15  G24.N2.0; TRANSFER ERROR
16  JMP END
17  $P_FULL
18  G24.N2.0; BUFFER FULL
19  $END
20  END
```

Description:

The command Nn:=CHN checks whether reception of a character string in the background has been completed (whether the end character was received). The result of the CHN function is stored in the integer register.

The result in the integer register is interpreted as follows:

Contents of Ni	Description
0	End criterion not yet received
1	End criterion received
2	Transfer error at the serial interface (parity,...)
3	Receive buffer full but end criterion not yet received

If the end criterion was recognized (in example Nn=1), the character buffer is copied into a processing buffer. The commands "POS" and "Copy" act on this processing buffer. This allows a further character string to be received with a renewed G533 command during execution of the commands "POS" and "Copy".

Note:

The function Nn:=CHN may only be used several times with Nn=0.

3.44.5 Search for the position of a character in the buffer

POS.

Instruction form:

Ni:=POS.n.m
Ni:=POS.Nn.m

Example:

```
Program : EXAMPLE
1    G500.2.6.1.0                ;Initialization of serial interface 2
2    G532.13 U_ERROR             ;Receives characters at the serial interface and stores
                                  them in the processing buffer until the end criterion 13 =
                                  0D hex = CR is received
3    N2:=POS.1.35                ;Search in processing buffer for character 35= 23 hex = #
                                  and store the item in N2
4    M1:=N2<1                    ;Character found ?
5    G21 M1.1 Z_NO               ;No!
6    N3:=COPY.N2.5 C_ERROR       ;As from the value of N2, convert the next 5 characters
                                  from the processing buffer into a number and store them
                                  in the integer register 3
7    JMP END
8    $U_ERROR
9    G24.N1.0; TRANSFER ERROR
10   JMP END
11   $C_ERROR
12   G24.N1.0; CONVERSION ERROR
13   JMP END
14   Z_NO
15   G24.N1.0; CHARACTER NOT FOUND
16   END
17   END
```

Description:

With the command, it is possible to search in the processing buffer with the criterion to determine whether a particular character ("m") occurs.
"m" is a number greater than 0 and less than 255 and represents the corresponding ASCII character.
The search in the processing buffer commences as from the position "n".

If the character is found, the item is stored in the integer register. If the character was not found, the value "0" is stored in the integer register.

3.44.6 Convert character into number

COPY.

Instruction form:

Ni:=COPY.n.m Label
Ri:=Copy.n.m Label
Ni:=COPY.Nn.Nm Label
Ri:=COPY.Nn.Nm Label

Example:

```
Program : EXAMPLE
1   G500.2.6.1.0                ;Initialization of the serial interface 2
2   G532.13 U_ERROR             ;Receives characters at the serial interface and stores
                                 them in the processing buffer until the end criterion 13 =
                                 0D hex = CR is received
3   N3:=COPY.2.5 C_ERROR        ;As from the 2nd character, convert the next 5 characters
                                 from the processing buffer into a number and store this
                                 number in the integer register 3
4   JMP END
5   $U_ERROR
6   G24.N1.0; TRANSFER ERROR
7   JMP END
8   $C_ERROR
9   G24.N1.0; CONVERSION ERROR
10  $END
11  END
```

Description:

The command COPY allows characters from the processing buffer to be converted into a number and stored in an integer or real number register.

COPY starts as from the "n"th character and takes the next "m" characters, provided that characters are contained in the processing buffer. If less than "m" characters are available, the conversion process is ended without an error.

If an error occurs, the conversion process is aborted. The register remains unchanged and the program is continued at the marker.

Possible errors:

- "n" is less than 1 or greater than 80
- "m" is less than 1 or greater than 10
- Not a single character from the quantity 0 to 9 was found during the conversion process.

3.44.7 Check whether a key is actuated

G540.

Instruction form:

G540.Nn

Example:

```
Program : Example
1   G11.0                ;Switch off sequence displays
2   G500.0               ;Switches the G500 commands to the current display
                          medium (LC display or, in the case of "Simulation of the
                          front panel" the serial interface)
3   G501                 ;Delete display
4   N1:=0
5   $LOOP
6   O1:=1 T10
7   O1:=0 T10
8   G540.N1              ;Checks whether a key is pressed and stores the key co-
                          de in N1. Otherwise N1 becomes 0
9   M1:=N1>0             ;Was a key pressed ?
10  G21 M1.0 LOOP        ;No !
11  G503.4.1             ;Position cursor in the 4th column of the 1st line
12  G502                 ;Delete up to the end of the line
13  G510.N1
14  G512.32
15  G520.N1
16  JMP LOOP
17  END
```

Description:

The command G540 checks whether a key on the PAC front panel has been pressed. If so, the key code is stored in the N register. If no key has been pressed, the N register is set to 0.

The command serves to interrogate the keyboard whilst a program part is being processed in order to react to possible requests of the operator.

Please note:

The command works only with the front panel of the PA-CONTROL (LCD display, membrane keyboard).
Refer to the table "PAC key code" in the "Technical appendix" section for the key codes.

3.44.8 Fetch a character from the keyboard

G541.

Instruction form:

G541.Nn

Example:

```
Program : Example
1   G11.0          ;Switch off sequence displays
2   G500.0         ;Switches the G500 commands to the current display
                    medium (LC display or, in the case of "Simulation of the
                    front panel", the serial interface)
3   G501           ;Delete display
4   N1:=0
5   $LOOP
6   G541.N1        ;Waits until a key is pressed and stores the key code in
                    N1
7   G503.4.1       ;Position cursor in the 4th column of the 1st line
8   G502           ;Delete up to the end of the line
9   G510.N1
10  G512.32
11  G520.N1
12  JMP LOOP
13  END
```

Description:

The command G541 waits until a key on the PAC front panel has been pressed and stores the key code in the N register.

The command G541 can be used to realize internal menus in the PA-CONTROL.

Please note:

The command works only with the front panel of the PA-CONTROL (LCD display, membrane keyboard).
Refer to the table "PCA key code" in the "Technical appendix" section for the key codes.

3.44.9 Input a register value

G542.

Instruction form:

G542.s.z.l.Ni
G542.s.z.l.Ri
G542.Ns.Nz.Nl.Ni
G542.Ns.Nz.Nl.Ri

Example:

```
Program : Example
1   G11.0                       ;Switch off sequence displays
2   G500.0                      ;Switches the G500 commands to the current display
                                medium (LC display or, in the case of "Simulation of the
                                front panel", the serial interface)
3   G501                        ;Delete display
4   $LOOP
5   G503.1.1
6   G510.Input time
7   G542.30.1.3.N1              ;Input of the N register 1 via the front panel
                                    The input field appears in the 30th column in the
                                1st line and is 3 characters long
8   G501
9   G503.1.2
10  G510.Time for output 1 is
11  G512.32
12  G520.N1
13  G512.32
14  G510. * 10ms
15  O1:=1 TN1 O1:=0
16  JMP LOOP
17  END
```

Description:

The command G542.s.z.l.Ni allows the operator to enter a register value during the program sequence.
The command opens an input field (see Chapter "User interface"). The length and position of the input field is defined by the operators (parameters) of the command.

The following applies:

s	:	stands for column
z	:	stands for line
l	:	stands for length of the input field
Ni	:	register worked with (storage)

3.45 Map to register / Map from register

G60?.

Introduction:

This instruction group permits register contents to be mapped to outputs or flags during automatic operation or to load the image of inputs to a register.

General instruction format:

G60?.Rn.m.i
G60?.Nn.m.i

Rn / Nn:	Source/destination
m:	First element of the destination/source address (corresponds to the least significant bit)
i:	Number of elements to which/from which mapping is to occur ($1 \leq i \leq 32$)

Note:

m + i must be equal to or less than the maximum number of elements used.

The register content is used as follows:

-Digits after the decimal point are truncated
-Negative number becomes positive number
-Maximum 32 bits are mapped.

Different formats are available:

-Binary image
-BCD image

3.45.1 Binary format to outputs

G600.

Instruction form:

G600.Rn.m.i
G600.Nn.m.i

Example 1:

```
Program : BEISPIEL
1    R3:=10.7
2    G600.R3.5.8          ;Content of R3 represented in binary code as of output 5
                          with 8 bits
3    END
```

States of the outputs after execution of the above instructions:

-Output 5	Reset (0)
-Output 6	Set (1)
-Output 7	Reset (0)
-Output 8	Set (1)
-Output 9	Reset (0)
-Output 10	Reset (0)
-Output 11	Reset (0)
-Output 12	Reset (0)

Example 2:

```
Program : BEISPIEL
1    N3:=18
2    G600.N3.2.4          ;Content of N3 represented in binary code as of output 2
                          with 4 bits
3    END
```

States of the outputs after execution of the above instructions:

-Output 2	Reset (0)
-Output 3	Set (1)
-Output 4	Reset (0)
-Output 5	Reset (0)

Note:

In example 1 the digits after the comma are omitted.
In example 2 only the 4 least-significant digits of the binary number are represented (The contents of the integer number register are nevertheless interpreted correctly!).

3.45.2 BCD format to outputs

G601.

Instruction form:

G601.Rn.m.i
G601.Nn.m.i

Example 1:

```
Program : BEISPIEL
1    R3:=10.7
2    G601.R3.5.8          ;Content of R3 represented in BCD code as of output 5
                          with 8 bits
3    END
```

States of the outputs after execution of the above instructions:

- Output 5 Reset (0)
- Output 6 Reset (0)
- Output 7 Reset (0)
- Output 8 Reset (0)
- Output 9 Set (1)
- Output 10 Reset (0)
- Output 11 Reset (0)
- Output 12 Reset (0)

Example 2:

```
Program : BEISPIEL
1    N3:=12
2    G601.N3.2.4          ;Content of N3 represented in BCD code as of output 2
                          with 4 bits
3    END
```

States of the outputs after execution of the above instructions:

- Output 2 Reset (0)
- Output 3 Set (1)
- Output 4 Reset (0)
- Output 5 Reset (0)

Note:

The digits after the decimal point are truncated in example 1.
Only the lowest-order digit of the BCD number is represented in example 2.

3.45.3 Binary format to flags

G602.

Instruction form:

G602.Rn.m.i
G602.Nn.m.i

Example 1

```
Program : BEISPIEL
1    R3:=11.3
2    G602.R3.4.8          ;Content of R3 represented in binary code as of flag 4
                          with 8 bits
3    END
```

States of the flags after execution of the above instructions:

-Flag 4 Set (1)
-Flag 5 Set (1)
-Flag 6 Reset (0)
-Flag 7 Set (1)
-Flag 8 Reset (0)
-Flag 9 Reset (0)
-Flag 10 Reset (0)
-Flag 11 Reset (0)

Example 2:

```
Program : BEISPIEL
1    R3:=65535
2    G602.R3.17.32        ;Content of R3 represented in binary code as of flag 1
                          with 32 bits
3    END
```

State of flags 17 to 49 after execution of the above instructions:

-Flags 17 - 31 are set
-Flags 32 - 49 are reset

Example 3:

```
Program : BEISPIEL
1    N3:=11
2    G602.N3.1.4          ;Content of N3 represented in binary code as of flag 1
                          with 4 bits
3    END
```

States of the flags after execution of the above instructions:

-Flag 1 Set (1)
-Flag 2 Set (1)
-Flag 3 Reset (0)
-Flag 4 Set (1)

3.45.4 Inputs in binary format to register

G603

Instruction form:

G603.Rn.m.i
G603.Nn.m.i

Example 1:

```
Program : BEISPIEL
1    G603.R3.2.8          ;Inputs 2 to 9 (2 = least-significant bit) are interpreted as a
                           binary number and the value is transferred to register 3
2    END
```

States of the inputs on execution of the above instructions:

-Input 2	Not energized (0)	Value = 1
-Input 3	Energized (1)	Value = 2
-Input 4	Not energized (0)	Value = 4
-Input 5	Energized (1)	Value = 8
-Input 6	Not energized (0)	Value = 16
-Input 7	Not energized (0)	Value = 32
-Input 8	Not energized (0)	Value = 64
-Input 9	Energized (1)	Value = 128

In binary code, this results in the following number "10001010" and, in register 3, value 138 (8AHex).

Example 2:

```
Program : BEISPIEL
1    G603.N3.5.4          ;Inputs 5 to 8 (5 = LSB) are interpreted as a binary num-
                           ber and the value is transferred to integer register 3
2    END
```

States of the inputs on execution of the above instruction:

-Input 5	Not energized (0)
-Input 6	Energized (1)
-Input 7	Not energized (0)
-Input 8	Energized (1)

In binary code, this results in the following number "1010" and, in register 3, value 10 (0AHex).

3.45.5 Flag in Binary Format in Register

G604.

Command form:

G604.Rn.m.i
G604.Nn.m.i

Example 1:

```
Programme:BEISPIEL
1    G604.R3.2.8          ;The flags 2 to 9 (2= lowest-significant bit) are interpreted
                          as a binary number and the value transferred to the R3
                          register
2    END
```

Input statuses when carrying out the above command:

- Flag 2	(0)	Value= 1
- Flag 3	(1)	Value= 2
- Flag 4	(0)	Value= 4
- Flag 5	(1)	Value= 8
- Flag 6	(0)	Value= 16
- Flag 7	(0)	Value= 32
- Flag 8	(0)	Value= 64
- Flag 9	(1)	Value= 128

In the binary code this is equivalent to the number "10001010" and in register 3 the value 138 (8AHex).

Example 2:

```
Programme: BEISPIEL
1    G604.N3.5.4          ;the flags  5 to 8 (5=LSB) are interpreted as binary num-
                          bers and the value transferred to the integer number reg-
                          ister 3
2    END
```

Input statuses when carrying out the above command:

- Flag 5	(0)
- Flag 6	(1)
- Flag 7	(0)
- Flag 8	(1)

In the binary code this is equivalent to the number "1010" and in register 3 the value 10 (0ahex).

3.46 Arithmetic operation

+ - * /

The following applies generally to assignments:

Destination : = Source

- Register
  - Constant (K)
  - Register
  - Absolute position of an axis (only in the case of real number registers)

The following applies generally to arithmetic operations:

Destination operand	**: =**	**Source operand 1**	**Operand**	**Source operand 2**
- Register		- Register	+ - * /	- Unsigned constant - Register

Note:

The result is dependent upon the operation and the sign of the register contents. K (constant) is also used as a real number in the description for integer. In the real application, only permitted type of number may be used. The source operands remain unchanged. One exception is if the destination operand is also the source operand.

Example:

```
R1:=5
R2:=3
R3:=R1+R2          ;the registers then have the following values: R1=5,
                    R2=3, R3=8
R1:=R1+R2          ;the registers then have the following values: R1=8, R2=3
```

Note:

Registers can be addressed directly and indirectly in value assignments and for basic arithmetic operations.

Direct addressing:

The letter R(N) is followed by a number (1-512) which identifies the desired register

Indirect addressing:

The letter R(N) is followed by "!" and then by a number (1-512) which refers to the register whose content identifies the desired register.

3.46.1 Load register

Rn:=
Nn:=

Instruction form:

Rn:=	K	Nn:=	K
R!n:=	K	N!n:=	K
Rn:=	A	Nn:=	Nm
R!n:=	A	N!n:=	Nm
Rn:=	Rm	Nn:=	Rm
R!n:=	Rm	Rn:=	Nm

Example:

```
Program : BEISPIEL
1   R2:=45          ;Load real number register 2 with 45
2   R3:=R2          ;Load real number register 3 with value of real number
                    register 2
3   R!2:=100.79     ;Load real number register 45 with 100.79
4   R21:=R!2        ;Load real number register 21 with value 100.79 from reg-
                    ister 45
5   R5:=X           ;Load real number register 5 with absolute position of the
                    X axis
6   N7:=87          ;Load integer register 7 with value 87
7   N65:=N7         ;Load integer register 65 with value from integer register
                    7
8   N1:=R45         ;Load integer register 1 with value from real number reg-
                    ister 45, value is rounded
9   R25:=N65        ;Load real number register 25 with value from integer reg-
                    ister 65
10  N2:=4
11  N!2:=17         ;Load integer register 4 (N2=4) with 17
```

Description:

Values can be loaded into the registers using these instructions.

Note:

No spaces (blanks) are permitted within the instruction.

3.46.2 Load Register with Axis Parameter

Rn:=

Command form:

Rn:= RAi

Example:

```
Programme:BEISPIEL
1   G90
2   G25.X          ;Return to datum X axis
3   FX1000         ;Set traversing speed to 1000Hz
4   X100           ;Move to position X100
5   R34:=RX1       ;Load real number register 34 with the traversing speed
                    of the X axis
6   FXR34          ;Set traversing speed at the preset parameter
7   X10            ;Move to position X10
8   END
```

Description:

With the aid of this command the axis parameters can be loaded in the register.

The following applies:

RX : X axis
RY : Y axis
...
RP : P axis

the parameter is selected by the digit:

1 : Traversing speed
2 : Return to datum speed
3 : Manual speed
4 : Slow traverse
5 : Start/stop speed
6 : Acceleration
7 : Gear factor
8 : Min. traversing range
9 : Max. traversing range

Note:

Blanks are not permitted in the command.

3.46.3 Addition

+

Instruction form:

Rn:=Rm+K
R!n:=Rm+K
Rn:=Rm+Ri
Rn:=R!m+Ri
Nn:=Nm+K
Nn:=Ni+Nm
N!n:=Nm+K
Nn:=N!m+Ni

Example:

```
Program : BEISPIEL
1    R2:=R2+57          ;Add content of R2 by 57
2    R2:=R4+R3          ;Add the content of R4 and the content of R3 and assign
                        result R2
3    R1:=5
4    R5:=100
5    R24:=R!1+R2        ;Add content of R5 and content of R2 and assign result to
                        register R24 Content of R1=5. The content of R5 is thus
                        accessed owing to indirect addressing. The content
                        of R5 and R2 is added and the result is assigned to R24.
                        R1, R5 and R2 remain unchanged.
6    N3:=N3+87          ;Add content of N3 by 87
7    N3:=N3+N5          ;Add content of N5 to content of N3
8    END
```

3.46.4 Subtraction

-

Instruction form:

Rn:=Rm-K
R!n:=Rm-K
Rn:=Rm-Ri
Rn:=R!m-Ri
Nn:=Nm-K
Nn:=Nm-Ni
N!n:=Nm-Ni
Nn:=N!m-Ni

Example:

```
Program : BEISPIEL
1    R2:=R2-57          ;Subtract content of R2 by 57
2    R2:=R2-R3          ;Subtract the content of R3 from content of R2

3    R1:=5
4    R5:=100
5    R24:=R!1-R2        ;Subtract content of R2 from content of R5 and assign re-
                        sult to register R24 content of R1 = 5. The content of R5
                        is thus accessed owing to indirect addressing. The con-
                        tent of R2 is now subtracted from this value and the result
                        is assigned to R24. R1, R5 and R2 remain unchanged.
6    N23:=N23-99        ;Subtract content of N23 by 99
7    N23:=N23-N4        ;Subtract content of N4 from content of N23
8    END
```

3.46.5 Multiplication

*****

Instruction form:

Rn:=Rm*K
R!n:=Rm*K
Rn:=Rm*Ri
Rn:=R!m*Ri
Nn:=Nm*K
Nn:=Nm*Ni
N!n:=N!m*Ni

Example:

```
Programm : BEISPIEL
1    R2:=R2*4          ;Multiply content of R2 by 4 and assign result to R2
2    R4:=R2*R3         ;Multiply content of R2 by content of R3 and assign result
                       to R4
3    R1:=5
4    R5:=100
5    R24:=R!1*R2       ;Multiply content of R5 by content of R2 and assign result
                       to register R24
6    N2:=N2*4          ;Multiply content of N2 by 4 and assign result to N2
7    N2:=N3*N6         ;Multiply content of N3 by content of N6 and assign result
                       to N2
8    END
```

3.46.6 Division

/

Instruction form:

Rn:=Rm/K
R!n:=Rm/K
Rn:=Rm/Ri
Rn:=R!m/Ri
Nn:=Nm/K
N!n:=Nm/K
Nn:=Nm/Ni
Nn:=N!m/Ni

Example:

```
Program : BEISPIEL
1    R2:=R2/3          ;Divide register content of R2 by 3 and assign result to R2
2    R4:=R2/R3         ;Divide register content of R2 by register content of R3
                       and assign result to R4
3    R1:=5
4    R5:=100
5    R24=R!1/R2        ;Divide register content of R5 by register content of R2
                       and assign result to register R24
6    N34:=N34/2        ;Divide content of N34 by 2 and assign result to N34. The
                       result is rounded (if necessary)
7    N25:=N67/N88      ;Divide content of N67 by content of N88 and assign re-
                       sult to N25. The result is rounded (if necessary)
8    END
```

Note:

A system error occurs if dividing by 0. This is indicated by message "SE5" on the display.

3.46.7 Compare operations

> = <

Compare operations can take place between registers and between registers and constants. The result of compare operations (logical 0 or 1) is assigned to a flag. The flag is always processed dependent upon the result.

Result true (logical 1)	:	Flag is set (logical 1)
Result false (logical 0)	:	Flag is reset (logical 0)

The following applies in general:

Destination	: =	Operand 1	Operation	Operand 2
- Flag		- Register	< = >	- Register
				- Constant

Note:

Operation "=" must be used with extreme care. If it is intended to check the limits of inputs or loop conditions, operations ">" or "<" are more reliable since they supply a clear and unequivocal result even if rounding problems occur!

3.46.8 Compare

> = <

Instruction form:

Mn:=Rm=K
Mn:=Rm>K
Mn:=Rm<K
Mn:=Rm=Ri
Mn:=Rm>Ri
Mn:=Rm<Ri
Mn:=R!m=Ri
Mn:=Nm=K
Mn:=Nm>K
Mn:=Nm<K
Mn:=Nm=Ni
Mn:=Nm>Ni
Mn:=Nm<Ni
Mn:=N!m=Ni

Example:

```
Programm : BEISPIEL
1    M34:=R1=56.77      ;Flag 34 is set if the content of R1 is equal to 56.77; otherwise, flag 34 is reset
2    M35:=R1>56.34      ;Flag 35 is set if the content of R1 is greater than 56.34 otherwise, flag 35 is reset
3    M36:=R1<56         ;Flag 36 is set if the content of R1 is less than 56; otherwise, flag 36 is reset
4    R2:=10
5    M12:=R!2=0         ;Flag 12 is set if the content of R10 is equal to 0; otherwise, flag 12 is reset
6    M34:=N1=56         ;Flag 34 is set if the content of N1 is equal to 56; otherwise, flag 34 is reset
7    M35:=N1>56         ;Flag 35 is set if the content of N1 is greater than 56; otherwise, flag 35 is reset
8    M36:=N1<56         ;Flag 36 is set if the content of N1 is less than 56; otherwise, flag 36 is reset
9    END
```

3.46.9 Complex examples

1. Load real number registers 10 to 20 with 0 at program start (reset counters)

```
Porgram : BEISPIEL
1    R1:=10                  ;R1 serves the purpose of indirect addressing
2    $SCHLEIFE
3    R!1:=0.0                ;Assign value to register to which the content of R1 points
4    R1:=R1+1                ;Next register
5    M1:=R1<20               ;Load all 10 registers (R10-R20) with value?
6    G21 M1.1 SCHLEIFE       ;Jump if not all registers have yet been loaded
7    END
```

2. Production sequence of pallet with uneven distances

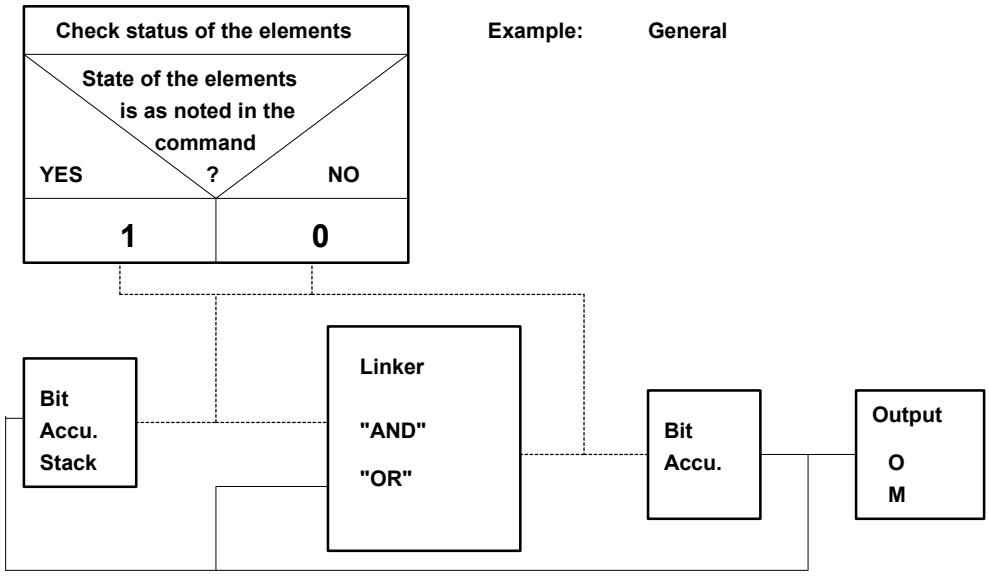
```
Program : PALETT_1
1    R11:=1
2    $EINLESEN
3    G24.R!11.0              ;Entry of the pallet positions
4    R11:=R11+1              ;Pointer to next pallet position
5    M1:=R11>5               ;All pallet positions read in?
6    G21 M1.0 EINLESEN       ;No!
7    R11:=1                  ;Position pointer for processing to 1st position
8    $NAECHST
9    +X100                   ;Approach fetch position
10   SUB TEILHOLE            ;Pick up part
11   +XR!11                  ;Approach deposit position
12   SUB TEILABLE            ;Deposit part
13   R11:=R11+1              ;Position pointer to next pallet position
14   M1:=R1>6                ;All pallet positions occupied?
15   G21 M1.0 NAECHST        ;No!


Program : TEILHOLE
1    I1.1                    ;Wait until part is in fetch position
...                          ;Start sequence for part
...
2    END


Program : TEILABLE
1    I7.1                    ;Pallet present
...                          ;Sequence for depositing part
...
2    END
```

3.47 Logic operations

A logic operation must always start with a LD command and end with an OUT command. Only commands of the logic operations may be used within a logic string (starting with LD, ending with OUT).

Logic operations are processed on the basis of the following functional mechanism.

Functional mechanism for logic operations



In order to also use with logic operations the writing method from the commands "Wait for the state of an element" (e.g. I3.1, M12.0), not only the element and the number such as input, marker or output, but also the desired state of the element was also assigned to the LD, AND and OR commands. If the current state of the element (I, M, O) corresponds to the state of the element with the logic command, a logic "1" is used for further work on converting the command (LD, AND, OR). Otherwise, further work is carried out with a logic "0".

The result of the logic operations LD, AND and OR is stored in the bit accumulator. The state of the bit accumulator is first transferred into the bit accumulator stack with an LD command and only then is the result of the element check stored in the bit accumulator.

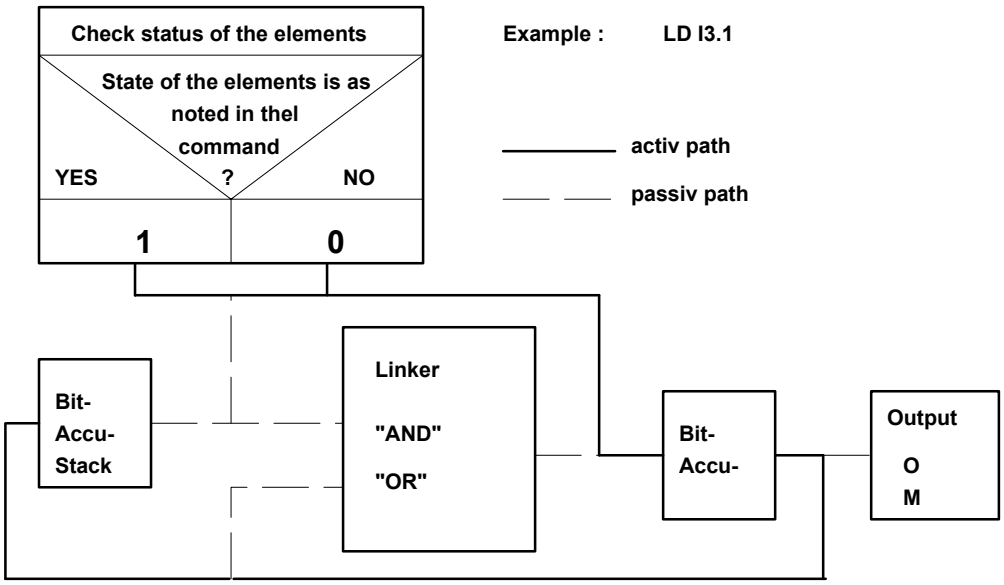
The logic operations AND-LD and OR-LD carry out a logic operation between the bit accumulator and bit accumulator stack and store the result in the bit accumulator.

The OUT command sends the state of the bit accumulator to the addressed element. If the bit accumulator is set to logic "1", the output or marker is set. Otherwise, it is reset. Several OUT commands can follow directly one after the other.

In the case of the PAB, every parallel sequence has its own bit accumulator and bit accumulator stack.

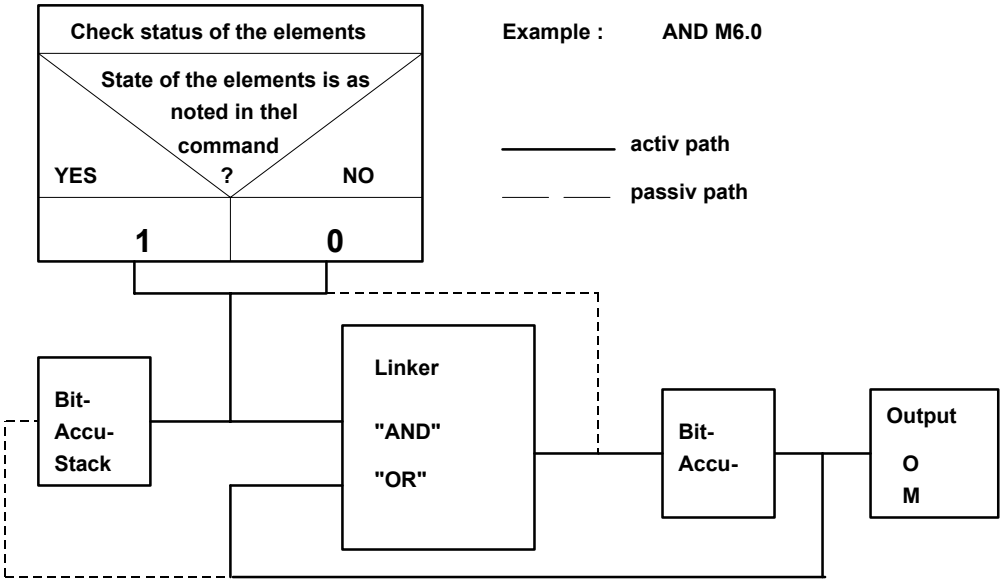
Command: LD

Functional mechanism for logic operations



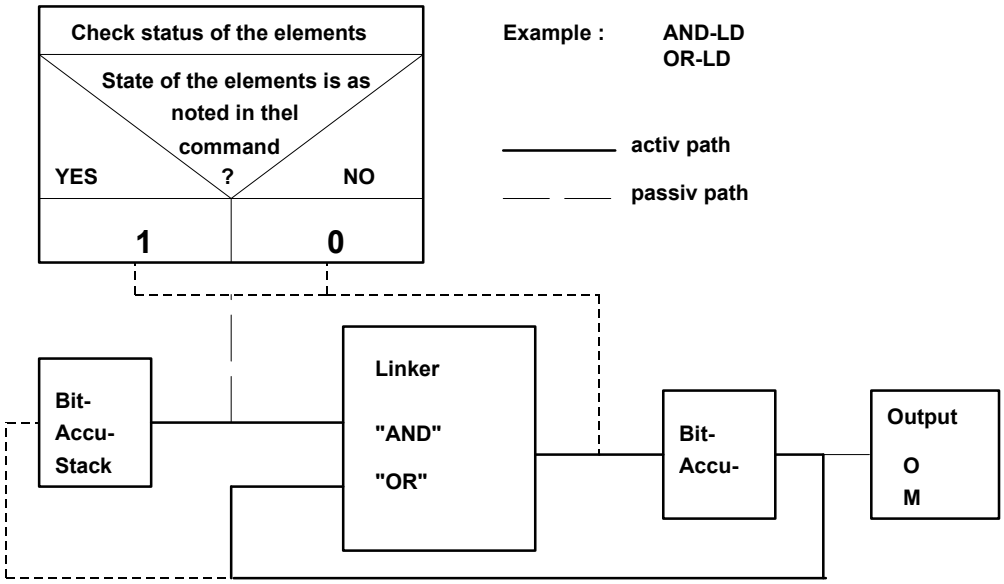
Command: AND or OR

Functional mechanism for logic operations



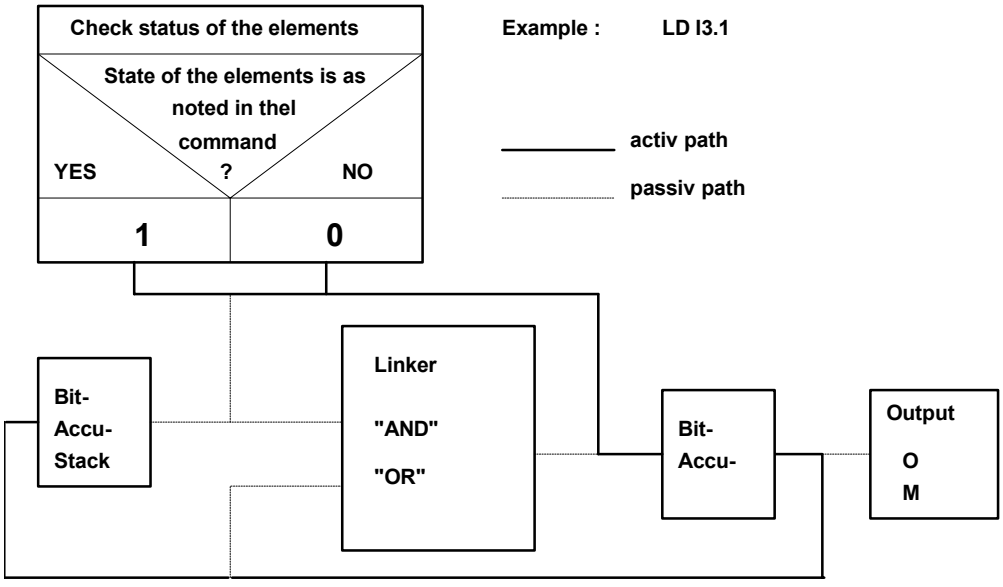
Command: AND-LD or OR-LD

Functional mechanism for logic operations



Command: OUT

Functional mechanism for logic operations



3.47.1 Logic AND operation

LD, AND, OUT

Instruction form:

AND Ii.n
AND Oi.n
AND Mi.n

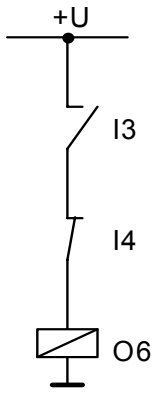
Example:

```
Program : EXAMPLE
1  LD I3.1        ;Begin the logic operation and load the bit accumulator
                  with logic "1", if input 3 is set. Otherwise load bit accumu-
                  lator with logic "0".
2  AND I4.0       ;If the input 4 is not set, make an AND logic operation
                  with logic "1" and the bit accumulator and store the result
                  in the bit accumulator
3  OUT O6         ;If the bit accumulator is set to logic "1", the output 6 is
                  set. Otherwise, output 6 is reset.
4  END
```

Application:

Building up logic AND operations between inputs, markers and outputs. The result of the logic operation can be assigned to a marker or to an output.

Schematic circuit diagram to above example:



3.47.2 Logic OR operation

LD, OR, OUT

Instruction form:

OR Ii.n
OR Oi.n
OR Mi.n

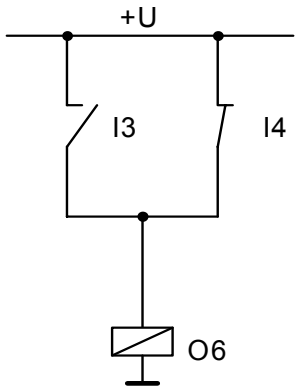
Example:

```
Program : EXAMPLE
1  LD I3.1          ;Begin logic operation and load bit accumulator with logic
                    "1" if input 3 is set. Otherwise, load bit accumulator with
                    logic "0".
2  OR I4.0          ;If input 4 is not set, make an OR logic operation with lo-
                    gic "1" and the bit accumulator and store the result in the
                    bit accumulator.
3  OUT O6           ;If the bit accumulator is at logic "1", output 6 is set. Oth-
                    erwise, output 6 is reset
4  END
```

Application:

Establishing logic OR operations between inputs, markers and outputs. The result of the logic operation can be assigned to a marker or to an output.

Schematic circuit diagram to above example:



3.47.3 Multi-stage logic AND operation

AND-LD

Instruction form:

AND-LD

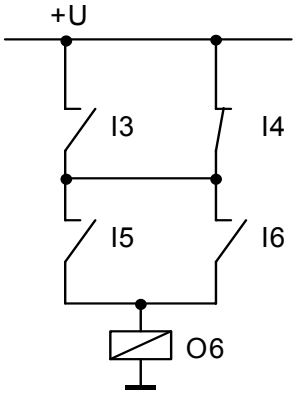
Example:

```
Program : EXAMPLE
1  LD I3.1      ;Start logic operation and load bit accumulator with logic
                "1" if input 3 is set. Otherwise, load bit accumulator with
                logic "0".
2  OR I4.0      ;If input 4 is not set, make an OR logic operation with lo-
                gic "1", and with the bit accumulator and store the result
                in the bit accumulator
3  LD I5.1      ;Begin logic operation and load the bit accumulator with
                logic "1" if input 5 is set. Otherwise, load it with logic "0".
4  OR I6.1      ;If input 6 is set, make an OR logic operation with logic
                "1", and the bit accumulator and store the result in the bit
                accumulator
5  AND-LD       ;Make an AND logic operation with the bit accumulator
                and with the bit accumulator stack and store the result in
                the bit accumulator
6  OUT O6       ;If the bit accumulator is set to logic "1", output 6 is set.
                Otherwise, output 6 is reset
7  END
```

Application:

Establishing complex (multi-stage) logic AND operations between inputs, markers and outputs. The results of the logic operation can be assigned to a marker or to an output.

Schematic circuit diagram to above example:



3.47.4 Multi-stage logic OR operation

OR-LD

Instruction form:

OR-LD

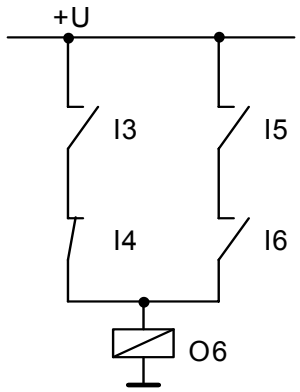
Example:

```
Program : EXAMPLE
1  LD I3.1      ;Begin logic operation and load bit accumulator with logic
                "1" if input 3 is set. Otherwise, load it with logic "0".
2  AND I4.0     ;If input 4 is not set, make an AND logic operation with
                logic "1"          and with the bit accumulator and store
                the result in the bit accumulator
3  LD I5.1      ;Begin logic operation and load the bit accumulator with
                logic "1" if input 5 is set. Otherwise, load it with logic "0".
4  AND I6.1     ;If input 6 is set, make an AND logic operation with logic
                "1" and with the bit accumulator and store the result in the
                bit accumulator
5  OR-LD        ;Make an OR logic operation with the bit accumulator and
                with the bit accumulator stack and store the result in the
                bit accumulator
6  OUT O6       ;If the bit accumulator is set to logic "1", the output 6 is
                set. Otherwise, output 6 is reset.
7  END
```

Application:

Establishing complex (multi-stage) logic OR operations between inputs, markers and outputs. The result of the logic operation can be assigned to a marker or to an output.

Schematic circuit diagram to above example:



3.47.5 Complex Logic Operations

If a multi-step logic operation is carried out, the result of a logic operation must be temporarily stored as the logic operation result is repeatedly overwritten in the Bit-accu-stack.
The example shows a multi-step logic operation. If the programming is carried out without temporary storage in flag 82, the Bit-accu-stack will be overwritten after the second LD command and logic operation result will be incorrect.

Example :

Programme: EXAMPLE

False

```
1 LD M4.1
2 OR M5.0
3 LD M14.1
4 OR M15.0
5 LD M24.1
6 OR M25.0
7 LD M34.1
8 AND M35.0
9 AND-LD
11 OUT M82
12 END
```

Right

```
1 LD M4.1
2 OR M5.0
3 OUT M82
4 ;
5 LD M14.1
6 OR M15.0
7 AND M82.1
8 OUT M82
9 ;
10 LD M24.1
11 OR M25.0
12 AND M82.1
13 OUT M82
14 ;
15 LD M34.1
16 AND M35.0
17 AND M82.1
18 OUT M82
19  END
```

Circuit diagram

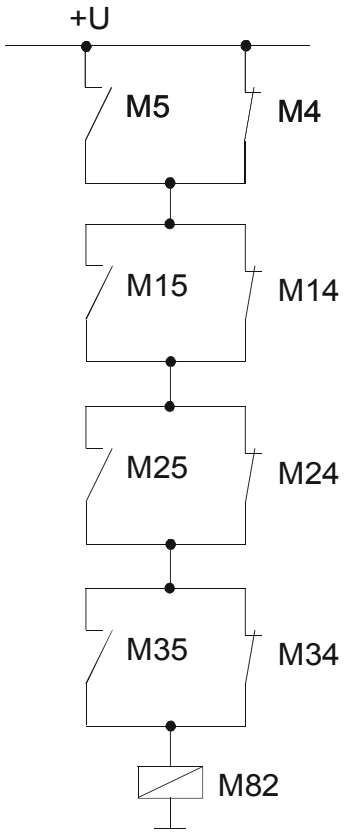


BILD056A

3.48 Communication with the parallel sequence controller

Please note:

> **This commands can only be used in conjunction with the PAB board (optional).**

3.48.1 Starting a parallel sequence

RUN

Instruction form:

RUN Name

Example:

```
Program : EXAMPLE.PNC
1  M1:=0          ;Reset marker 1
2  M2:=0          ;Reset marker 2
3  RUN OUT1       ;Start the program "OUT1"
4  RUN OUT2       ;Start the program "OUT2"
5  M1.1           ;Wait until marker 1 is set to logic "1"
6  M2.1           ;Wait until marker 2 is set to logic "1"
7  O3:=1          ;Set the output 1
8  END            ;End program "EXAMPLE".
```

```
Program : OUT1.PAB
1  O1:=1          ;Set the output 1
2  T10            ;Waiting time of 100 ms
3  O1:=0          ;Reset output 1
4  M2:=1          ;Set marker 2
5  END            ;End program "OUT1"; it is removed from the parallel sequence interpreter
```

```
Program : OUT2.PAB
1  O2:=1          ;Set output 2
2  T20            ;Waiting time of 200 ms
3  O2:=0          ;Reset output 2
4  M1:=1          ;Set marker 1
5  END            ;End program "OUT2"; it is removed from the parallel sequence interpreter
```

Application:

Starting sequence programs.

Description :

The RUN command allows sequence programs (programs of the type "PAB") to be started on the parallel sequence controller. The programs are taken into the sequence interpreter of the PAB where they are processed.

Please note:

The programs must be present in the memory of the PAB.
A program can only be entered once into the sequence interpreter.
Up to 32 programs can be processed simultaneously in the sequence interpreter.

3.48.2 Stopping a parallel sequence

SLEEP

Instruction form:

Sleep Name

Example:

```
Program : EXAMPLE.PNC
1  M1:=0                    ;Reset marker 1
2  M2:=0                    ;Reset marker 2
3  RUN OUT1                 ;Start program "OUT1"
5  M1.1                     ;Wait until marker 1 is set to logic "1"
6  SLEEP OUT1               ;Stop the program "OUT1"
7  M1:=0                    ;Reset marker 1
8  I1.1                     ;Wait until input 1 is set to logic "1"
9  RUN OUT1                 ;Continue the program "OUT1"
10 M2.1                     ;Wait until marker 2 is set to logic "1";
11 O3:=1                    ;Set output 1
12 END                      ;End program "EXAMPLE"

Program : OUT1.PAB
1  O1:=1                    ;Set output 1
2  M1:=1                    ;Set marker 1
3  M1.0                     ;Wait until marker 1 is set to logic "0"
4  T10                      ;Waiting time of 100 ms
5  O1:=0                    ;Reset output 1
6  M2:=1                    ;Set marker 2
7  END                      ;End program "OUT1"
```

Application:

Stopping sequence programs, e.g. when a safety door is opened.

Description :

Sequence programs (programs of the type "PAB") can be stopped on the parallel sequence controller with the SLEEP command. The program is put into SLEEP mode by the SLEEP command in the sequence interpreter of the PAB, i.e. it is not processed further. It remains at the current command. This program can then be continued as from the current command with the RUN command.

Please note:

The SLEEP command can only be used on programs which are in the RUN mode. Dwell and monitoring time commands are also stopped.

3.48.3 Ending a parallel sequence

CANCEL

Instruction form:

Cancel Name

Example:

```
Program : EXAMPLE.PNC
1  M1:=0                ;Reset marker 1
2  M2:=0                ;Reset marker 2
3  RUN OUT1             ;Start program "OUT1"
4  I7.1                 ;Wait until input 7 is energized
5  CANCEL OUT1          ;End program "OUT1"
6  END                  ;End "EXAMPLE" program

Program : OUT1.PAB
1  $START
1  O1:=1                ;Set output 1
2  T10                  ;Waiting time of 100ms
3  O1:=0                ;Reset output 1
4  T10                  ;Waiting time of 100ms
5  JMP START            ;Jump to "START" marker
6  END
```

Application:

Immediate ending of sequence programs. This makes space in the sequence interpreter for a new program to be started, e.g. operating mode change of a production system.

Description :

The CANCEL command allows sequence programs (programs of the type "PAB") to be ended on the parallel sequence controller. The program is removed from the sequence interpreter of the PAB by the CANCEL command and is no longer further processed.

Please note :

The CANCEL command can only be used on programs which are in RUN mode or SLEEP mode.
The program is aborted at the command currently being processed, i.e. elements which are still influenced are supplied (marker, outputs, register).

3.48.4 Starting parallel sequences

CASE.RUN

Command form:

CASE.RUN.Ni
(Name1)
(Name2)
...
ELSE Marke_e

Example:

```
Programme : BEISPIEL.PNC
1    $FEHLER
2    G24.N8.0                ;Enter value via keyboard
3    CASE.RUN.N8             ;Check contents of N8 and start PAB programme at:
4    (PROG-1)                ;N8=1 : start programme PROG-1
5    (PROG-2)                ;N8=2 : start programme PROG-2
6    (PROG-3)                ;N8=3 : start programme PROG-3
7    ELSE FEHLER             ;Jump to "ERROR" mark, if the value of N8 is beyond the
                             intended limits (here when N8<1 or N8>3)
8    O12:=0
9    I1.1
10   END


Programme: PROG-1.PAB
1    O1:=1 T100 O1:=0        ;Set output 1 for 1 s
2    END


Programme: PROG-2.PAB
1    O2:=1 T100 O2:=0        ;Set output 2 for 1 s
2    END


Programme: PROG-3.PAB
1    O3:=1 T100 O3:=0        ;Set output 3 for 1 s
2    END
```

Application:

Depending on e.g. different types of manufacturing, different types of processing operations can be executed in parallel to other operations. These operations can be initiated variable with CASE.RUN.

Description:

With the command CASE.RUN parallel sequences where startet depending on a integer number register. Programme distributors can be easily created using this command.
The PA-CONTROL checks the contents of the integer number register and starts a parallel sequence in accordance with the value. If the contents of the integer number register are smaller than 1 or greater than the number of elements of the name table (in the example given: PROG-1, PROG-2, PROG-3), a jump is made to the mark that is defined after ELSE. Otherwise the respective PAB programme is called up and continued with the programme line after the ELSE branch (in the example given: Line 8 with O12:= 0).

The following allocations apply:

Value in register	Name in name table
1	1. Name
2	2. Name
...	...
n	n. Name

Note:

The PAB programmes have to be available, otherwise the error message „programme not available“ will be displayed.
If a parallel sequence has already been startet it cannot be startet again. Otherwise the error message “programme already running“ will be displayed.
If the command is set in a PAB programme, a maximum of 24 CASE branches is possible. Otherwise the error message „PAB: in CASE too much (...) switches“.

3.48.5 Stopping parallel sequences

CASE.SLEEP

Command form:

CASE.SLEEP.Ni
(Name1)
(Name2)
...
ELSE Marke_e

Example:

```
Programme : BEISPIEL.PNC
1    RUN PROG-1              ;Starting the programme PROG-1
2    RUN PROG-2              ;Starting the programme PROG-2
3    $ANFANG
..   ...
25   G24.N10.0               ;Enter value via keyboard
26   CASE.SLEEP.N10          ;checks the contents of N10 and stops the corresponding
                             PAB programme at N10:
27   (PROG-1)                ;N10=1: stops the programme PROG-1.PAB
28   (PROG-2)                ;N10=2: stops the programme PROG-2.PAB
29   ELSE FEHLER             ;Jump to mark "FEHLER", if the value of N10 is beyond
                             the intended limits (here when N10<1 or N10>2).
..   ...
90   $FEHLER
100  END

Programme: PROG-1.PNC
1    $ANFANG                 ;Jump mark
2    O1:=1                   ;Set output 1
3    T100                    ;wait 1s
4    O1:=0                   ;reset output 1
5    T100                    ;wait 1s
6    JMP ANFANG              ;go to mark ANFANG
7    END


Programme: PROG-2.PAB
1    $ANFANG                 ;Jump mark
2    O2:=1                   ;set output 2
3    T50                     ;wait 0,5s
4    O2:=0                   ;reset output 2
5    T50                     ;wait 0,5s
6    JMP ANFANG              ;go to mark ANFANG
7    END
```

Application:

Depending on e.g. different types of manufacturing, different types of processing operations can be executed in parallel to other operations. They can be stopped variable with CASE.SLEEP.

Description:

With the command CASE.SLEEP parallel sequences can be stopped depending on a integer number register. Programme distributors can be easily created using this command.
The PA-CONTROL checks the contents of the integer number register and starts a parallel sequence in accordance with the value. If the contents of the integer number register are smaller than 1 or greater than the number of elements of the name table (in the example given: PROG-1, PROG-2), a jump is made to the mark that is defined after ELSE. Otherwise the respective PAB programme is called up and continued with the programme line after the ELSE branch.

The following allocations apply:

Value in register	Name in name table
1	1. Name
2	2. Name
...	...
n	n. Name

Note:

The PAB programmes have to be available, otherwise the error message „programme not available“ will be displayed.
If a parallel sequence has already been stopped it cannot be stopped again. Otherwise the error message “programme already in Sleep“ will be displayed.
If the command is set in a PAB programme, a maximum of 24 CASE branches is possible. Otherwise the error message „PAB: in CASE too much (...) switches“.

3.48.6 Terminating parallel sequences

CASE.CANCEL

Command form:

```
CASE.CANCEL.Ni
(Name1)
(Name2)
...
ELSE Marke_e
```

Example:

```
Programme : BEISPIEL.PNC
1    RUN PROG-1              ;Starting the programme PROG-1
2    RUN PROG-2              ;Starting the programme PROG-2
3    $ANFANG
..   ...
25   G24.N10.0               ;Enter value via keyboard
26   CASE.CANCEL.N10         ;checks the contents of N10 and terminates the corre-
                             sponding PAB programme at N10:
27   (PROG-1)                ;N10=1: terminates programme PROG-1.PAB
28   (PROG-2)                ;N10=2: terminates programme PROG-2.PAB
29   ELSE FEHLER             ;Jump to mark "FEHLER", if the value of N10 is beyond
                             the intended limits (here when N10<1 or N10>2).
..   ...
90   $FEHLER
100  END

Programme: PROG-1.PNC
1    $ANFANG                 ;Jump mark
2    O1:=1                   ;Set output 1
3    T100                    ;wait 1s
4    O1:=0                   ;reset output 1
5    T100                    ;wait 1s
6    JMP ANFANG              ;go to mark ANFANG
7    END


Programme: PROG-2.PAB
1    $ANFANG                 ;Jump mark
2    O2:=1                   ;set output 2
3    T50                     ;wait 0,5s
4    O2:=0                   ;reset output 2
5    T50                     ;wait 0,5s
6    JMP ANFANG              ;go to mark ANFANG
7    END
```

Application:

Depending on e.g. different types of manufacturing, different types of processing operations can be executed in parallel to other operations. These operations can be terminated variable with CASE.CANCEL.

Description:

With the command CASE.CANCEL parallel sequences can be terminated depending on a integer number register. Programme distributors can be easily created using this command.
The PA-CONTROL checks the contents of the integer number register and starts a parallel sequence in accordance with the value. If the contents of the integer number register are smaller than 1 or greater than the number of elements of the name table (in the example given: PROG-1, PROG-2), a jump is made to the mark that is defined after ELSE. Otherwise the respective PAB programme is called up and continued with the programme line after the ELSE branch.

The following allocations apply:

Value in register	Name in name table
1	1. Name
2	2. Name
...	...
n	n. Name

Note:

If the command is set in a PAB programme, a maximum of 24 CASE branches is possible. Otherwise the error message „PAB: in CASE too much (...) switches“.

4 Commissioning

Contents

4.1 Installation of a PAC-SC Compact

Ambient conditions, such as temperature, soiling or humidity, influence operation of a positioning and sequential control and may cause errors. This fact must be taken in consideration as early as the installation stage. The PAC-SC Compact may not be installed in the area of a strong electrical or magnetic field (e.g. welding transformer) and may not be mechanically stressed (e.g. by vibration). Ensure that the fan inlet and the fan outlet openings are unobstructed.

4.2 Wiring the connections

Important:

Plugs and connectors may be connected and disconnected only with the power supply switched off!

Please refer to Chapter "Technical annex" for the pin assignments of the individual plugs and connectors with connection examples.

-Limit switches of all available axes
-Stepping motors
-Inputs
-Outputs
-External start (if used)
-External stop (if used)

Note :

Always connect the housing to the PE wire!

The Stop switch must be closed before the Start switch is closed (external start/stop) in order to guarantee proper functioning of the start and stop.
Please refer to the function schematic in Chapter Operator interface ⇒ Automatic.

4.3 Adjusting the position controller

Please refer to the section "Putting position controller into operation" for details about commissioning and adjusting the position controller.

4.4 Function and status check of the inputs and outputs

The inputs and outputs of the PA-CONTROL can be checked in menu sub-option "Program test and diagnosis" (see Chapter Operator interface - program test and diagnosis for a description).

- Checking the function of the limit switches
- Checking the function of the connected inputs
- Checking the function of the connected outputs

4.5 Setting the parameters

Please refer to Chapter Parameters for setting the parameters.

Note:

The parameter values must be checked to ensure that they are correct for the application!

4.6 PNC programming

In order to create a new PNC program, you must access the PA-CONTROL program editor from the main menu by selecting menu options "Programming - Create new program" and by entering the program name "BEISPIEL" with type "PNC".

Note:

Please always check that the program can be executed on the available machine.

The PCN program is to perform the following functions:

-Reference travel with the X axis
-Set absolute dimension system
-Set positioning speed to 50 mm/s
-Move X axis to position 100
-Wait 2 seconds
-Move X axis to position 10
-Wait 1 second

Press the following keys in the program editor to create this program.

Program : BEISPIEL

Program	Enter by pressing keys					
G90	G	9	0	ENTER		
G25.X	G	2	5	.	X	ENTER
FX50	F	X	5	0	ENTER	
X100	X	1	0	0	ENTER	
T200	T	2	0	0	ENTER	
X10	X	1	0	ENTER		
T100	T	1	0	0	ENTER	
END	END	ENTER				

Program entry can be terminated and quit by pressing key "ESC" and answering the prompt to save the program.

ESC
1
1

If errors have been made when entering the program or if incorrect instructions have been entered, these errors must be rectified. Otherwise, it is not possible to save the program.

4.7 Defining the start program

Before the program which you have created can be run, the PAC-SC Compact must be informed of the particular PNC program with which the automatic sequence is to commence. This is defined in menu sub-option Sequence definitions (see Chapter Operator interface - Sequence definitions).

4.8 Running the program created

In order to run the program, the PA-CONTROL must be switched to Automatic mode (main menu 1st line). When you press the Start key, the program runs. The program run can be interrupted by pressing the Stop key.

The following is displayed during the program run:

1st line:	Program name and subroutine nesting depth
2nd line:	Program line number with the instructions

e.g.:

```
Beispiel 0
 1 G90
```

The program performs the following steps.

Display (2nd line)	Step executed by the PA-CONTROL
1 G90	Positioning is switched over to absolute dimension system
2 G25.X	The PA-CONTROL performs reference travel with the X axis
3 FX50	The traversing speed for the X axis is set to 50 mm/s
4 X100	The PA-CONTROL moves the X axis to position 100
5 T200	Waiting time 1 second
6 X10	The PA-CONTROL moves to position 10
7 T100	Waiting time 1 second
8 END	Automatic sequence is terminated.

The PA-CONTROL returns to the main menu.

5 Parameters

Contents

5.1 General information on parameters

We make a distinction between two parameters types on a PAC-SC Compact

- System parameters : Parameters which relate to the general system, such as operator language and serial interface, ...

- Axis parameters : Parameters relating to the axes, such as traversing speed and acceleration, ...

Various basic settings must be made before the program can be run feasibly. This means that differently dimensioned and differently loaded axes must be moved under different conditions as regards acceleration, maximum speed or gear factor.

All parameters can be loaded with default values and checked individually and, if necessary, modified (see Chapter Operator interface).

5.2 System parameters

List of system parameters:

Parameter name	Min. value	Actual value	Max. value
Operator language	1	1	2
Number of axes	0	2	6
External start input No.	0	0	128*
External stop input No.	0	0	128*
Manual enable input No.	0	0	128*
Standby output No.	0	0	128*
Fault output No.	0	0	128*
Autostart	0	0	1
Front panel	0	0	1
Serial interface No.	1	1	2
Baud rate of the serial interface	1	6	7
Format of the serial interface	1	1	4
Handshake of the serial interface	0	0	1
PA-CONTROL address	1	1	31

* A maximum of 112 inputs and 112 outputs are available with PAC-Servo.

Operator language

This parameter is used to switch over to the various operator languages.

1 = German
2 = English
3 = French (Not yet implemented in this version)

Number of axes

The number of axes available to the operator will differ, dependent upon device configuration. This maximum number of axes depends upon the positioning method and is stored in parameter "Maximum number of axes". Dependent upon application, the operator can now set the current number of axes which must, however, always be equal to or less than the maximum number of axes of the system.
This set current number of axes is assumed as the maximum limit for the operator in the case of axis-related actions (positioning, processing parameters, ...).

External start input No.

Any input can be defined for the external start signal on the PAC-SC Compact. If 0 is entered, corresponding to the default value, there is no external start option.

The following applies to start:

Transition from not energized to energized triggers the start (NO contact function, positive edge evaluation, see Operator interface - automatic).

External stop input No.

Any input of the PAC-SC Compact can be defined for the external stop signal. If 0 is entered, corresponding to the default value, there is no external stop option.

The following applies to stop:

If the external stop is defined, the selected input must be energized. Automatic mode is interrupted if the input is de-energized (NC contact function, level-operated, see Operator interface - automatic).

Manual enable input No.

On the PA-CONTROL, any input can be defined for enabling the manual functions. If "0" is entered, corresponding to the default value, the manual function is freely accessible. If a value greater than 0 is entered, the input of the corresponding number must be energized so as to permit access to the manual functions.
Not yet activated in this version.

Standby output No.

Any output can be defined for output of standby (PA-CONTROL OK) on the PAC-SC Control.

Fault output No.

Any output can be defined for output of a fault which has occurred in the automatic sequence (e.g.: limit switch actuated, value too high etc.) on the PAC-SC Compact. If a fault occurs in Automatic mode, the output is set. If the automatic sequence is then aborted and you return to the menu, the output is then reset.

Autostart

If the Autostart parameter is set to 1, the PAC-SC Compact starts execution of the program after switching on (Reset), i.e. it automatically changes to Automatic mode.

Preconditions:

- Start program must be defined
- An input must be defined as external stop and must be energized
- An input must be defined as external start

Important:

The Autostart function is executed once only after switching on (Reset). If the program is to be executed several times, either an endless loop must be programmed or it must be executed again with "Start".

Front panel

Not yet implemented in this version

Serial interface number

The PAC-SC Compact has 2 serial interfaces for communication (transfer of programs and parameters) with other systems. The serial interface number defines whether the first or second interface is used for this task.

Baud (baud rate)

The Baud parameter defines the baud rate for transfer of programs and parameters.

Parameter value	Transfer rate
1	110 baud
2	300 baud
3	1200 baud
4	2400 baud
5	4800 baud
6	9600 baud
7	38800 baud

Format

The Format parameter defines the transfer format for transfer of programs and parameters via the serial interface.

Parameter value	Transfer format
1	8 bits, no parity
2	7 bits, no parity
3	7 bits, even parity
4	7 bits, odd parity

Handshake

The Handshake parameter defines what handshake procedure is used when transferring programs and parameters via the serial interface.

Parameter value	Handshake
0	No handshake
1	Hardware handshake

PA-CONTROL address

The PA-CONTROL address parameter defines the number of the PA-CONTROL if it is operated in a network.
The possible value lies between 0 and 31.
Not yet activated in this version.

5.3 Axis parameters

List of axis parameters

Parameter name	Min. value	Actual value	Max. value
Max. autospeed	10	5000	50000
Reference speed	10	1000	50000
Manual speed	10	500	2000
Slow manual speed	10	300	1000
Start/stop speed	10	400	1000
Acceleration	1	20	500
Transmission factor	0.001	1	1000
Traversing range, minimum	-8000000	0	+8000000
Traversing range, maximum	-8000000	10000	+8000000

Max. autospeed

This parameter value is used by the control as the traversing speed in Automatic mode if no other value has been preset in the program, e.g. with instruction FAn. It is also the limit value for value input within a program.

Reference speed

Reference travel, i.e. the travel to the limit switch, is executed at this speed.

Important:

This speed must be adapted to the available overtravel distance. The overtravel distance is the distance between the limit switch and the mechanical stop. When the limit switch is detected, the motor is decelerated to stop with the preset acceleration ramp.

Manual speed

Positioning in Manual mode is performed at this speed unless set otherwise.

Slow manual speed

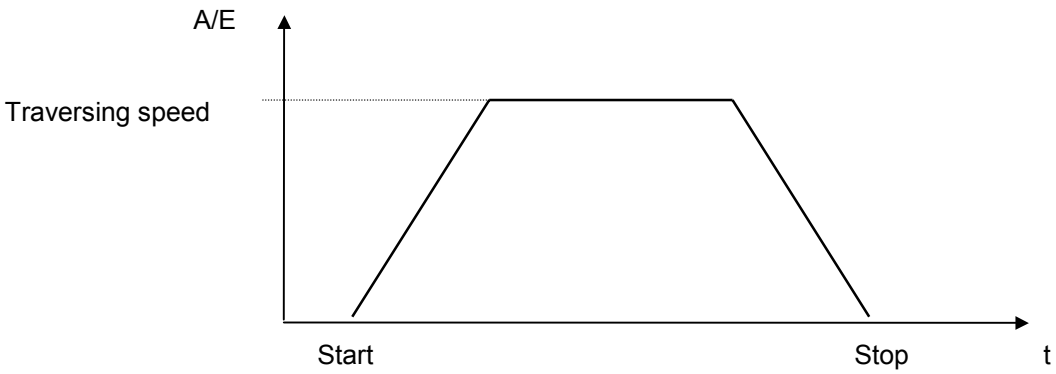
The distance from the limit switch to leaving the limit switch is travelled at this speed during reference travel.

Start/stop speed

This is the speed at which the positioning operation is commenced and terminated again (see sketch).

Acceleration

All positioning operations are executed on the basis of a ramp function. A low value for the acceleration
corresponds to a gentle ramp, e.g. a low acceleration. A high value leads to a high acceleration with a steep ramp. (Acceleration in indication unit (AE) / s^2)



Transmission factor

This parameter makes it possible to perform program input in mm, inches or degrees etc.
The following formula is used to determine this parameter:

$$\text{Transmission factor (GF)} = \frac{\text{Increments per revolution}}{\text{Feed per motor revolution}} \times \text{Unit of measurement}$$

Example 1:
No gear unit

Encoder with 500 incements per revolution
Spindle lead 5 mm
Input in mm

$$GF = \frac{400}{5\text{mm}} \times 1\text{ mm} = 80$$

If a gear unit is used between motor and driven shaft, the above calculation formula for the gear factor changes as follows:

$$\text{Transmission factor (GF)} = \frac{\text{Incremets per revolution x gear ratio}}{\text{Feed per encoder revolution}} \times \text{Unit of measurement}$$

Example 2:
with gear unit

Encoder with 500 increments per revolution
Feed per Encoder revolution 300 mm
Gear reduction ratio 15:1
Input in mm

$$GF = \frac{400 \times 15}{300\text{ mm}} \times 1\text{mm} = 20$$

Table for transmission factor calculation

This table makes it possible to determine the gear factor simply by reading it off.

Increments per encoder revolution and feed per motor revolution must be selected. The gear factor can be read off at the intersection point.

Steps per revolution	Feed per motor revolution (mm) 2	4	5	10	20	40	80
200	100	50	40	20	10	5	2.5
400	200	100	80	40	20	10	5
500	250	125	100	50	25	12.5	6.25
1000	500	250	200	100	50	25	12.5

Traversing range minimum/Traversing range maximum

These parameters serve as software limit switches.
The positioning range can be restricted by defining the values for maximum and minimum traversing range.
The check is conducted before each positioning operation. If necessary, an error message is issued and the program is aborted.

6 Options

Contents

6.1 Interbus-S connection

Installation of the optional Interbus-S board in the units of the PA-CONTROL family opens up new and expanded application capabilities for the PA-CONTROL.
The PA-CONTROL is integrated as the slave in the 2-wire remote bus user in the Interbus-S system. The PA-CONTROL uses 64 bits or data points in the Interbus-S. Various significances are assigned to the 64 bits in order to fully utilize the diverse capabilities and fully comply with the requirements in respect of the PAC.

The PAC can be addressed in two operating modes via the Interbus-S:

- **Program and parameter transfer between host and PAC in both directions.** These capabilities are also implemented in the PROPAC programming environment.

- **Execution of individual commands whilst the PAC is running its program.** In this case, a host (PC or PLC) can exchange individual data with a PAC simply via the Interbus-S.

An overview of the functions available is shown below. IEF has developed driver functions in the Turbo Pascal and Microsoft C languages for a PC as master.

6.1.1 Instructions for the PA-CONTROL via Interbus-S:

cmd_stop	Stop Automatic, stop positioning (corresponds to the STOP key)
cmd_abort	Abort Automatic mode (halt positioning) and change to initial state
cmd_start	Change from initial state to Automatic mode and process programs as of the "Start program"
cmd_data_transmission	Start data transmission; the mode and direction of data transmission are defined in the instruction parameter
cmd_get_aktual_posaktuelle	Fetch position of an axis
cmd_put_single_reminder	Change flag
cmd_get_reminder_word	Read flag word (marker)
cmd_put_int_reg	Write to integer register
cmd_get_int_reg	Read integer register
cmd_put_float_reg	Write to floating point register
cmd_get_float_reg	Read floating point register
cmd_put_single_output	Change output
cmd_get_output_word	Read output word
cmd_get_input_word	Read input word (input)
cmd_put_reminder_refresh	Change flag and with consecutive updating
cmd_get_reminder_refresh	Read double-flag word (marker) with consecutive updating
cmd_get_int_reg_refresh	Read integer register with consecutive updating
cmd_get_float_reg_refresh	Read floating point register with consecutive updating
cmd_put_output_refresh	Change output with consecutive updating
cmd_get_output_refresh	Read output word with consecutive updating
cmd_get_input_refresh	Read input word with consecutive updating

6.1.2 Interbus-S remote bus slave board for PA-CONTROL

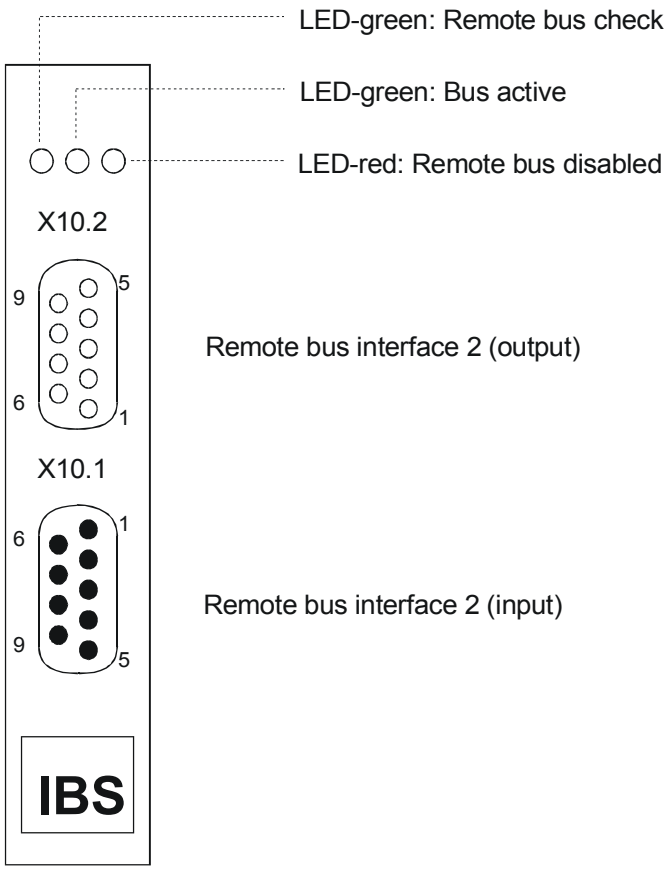


BILD058E

Pin assignment (input)	Remote bus interface 1	Pin assignment (output)	Remote bus interface 2
1	TPDO1	1	TPDO2
2	TPDI1	2	TPDI2
3	GND	3	GND
4	Not occupied	4	Not occupied
5	Not occupied	5	BR
6	/TPDO1	6	/TPDO2
7	/TPDI1	7	/TPDI2
8	Not occupied	8	Not occupied
9	Not occupied	9	BR

Use the remote bus cable complying with the specifications of Messrs. Phoenix Contact.

6.1.3 Interbus-S monitor:

If the Interbus-S system (network) is in operation, you will see sub-menu option "8 = Interbus-S monitor" in the "System diagnosis" menu.

The Interbus-S monitor serves to test communication with the PA-CONTROL via Interbus-S.

You will see the following display after you switch to this menu option

```
IBS-IN     : 00  00  00  00  00  00  00  00
IBS-OUT    : 49  45  46  20  00  00  3F  00
```

The current 64 bit (8 bytes) PA control input data and PA-CONTROL output data are shown on the display. This data is displayed byte-serially in hexadecimal notation. The instruction interpreter for Automatic mode is active on the PA-CONTROL, i.e. all instructions permitted in Automatic mode of the PA-CONTROL can be executed. This permits the Interbus-S operator to check the protocol running on the Interbus-S network between Master and PA-CONTROL.

You can quit the Interbus-S monitor by pressing the "ESC key".

6.2 I/O Card

Board with 16 inputs and outputs
Specification in accordancewith the standard inouts an outputs
(see section 1 “Technical data”)

7 Technical annex

Contents

7.1 Error message of the PA-Control

7.1.1 Error messages in the automatic sequence and during manual traversing

Error text:	Number:	Possible cause of error:
No positioning module		Jumper for positioning module selection not set correctly on the CPU
Illegal axis selection		Axis name not enabled in system parameter (number of axes)
Power section not ready	3	Power section not connected, power sections defective, motor not connected, rotation monitoring trip-ped, —
Limit switch defective	4	Limit switch approached during positioning
Value too high	5	Value for speed higher than parameter value
Value too low	6	Value for speed less than parameter value
Overflow, arithmetic operation	7	Numbers out of range
Program does not exist	8	Program does not exist in PA-CONTROL when subroutine is called
End of program does not exist	9	No END
Excessive nesting depth	A	Too many subroutines called
Limit switch not reached	B	Limit switch not reached during reference travel, limit switch damaged, limit switch not being actuated (mechanical fault), —
Range transgression	C	Parameter value for the traversing range has been exceeded when positioning
Illegal instruction	D	Instruction unknown or illegal for this configuration
Positioning module defective	E	Hardware fault
No reference point	F	A positioning instruction has been called without prior reference travel or G25.0

7.1.2 Error messages in the program editor during the syntax check

Error text:	**Description/Explanation**:
Illegal axis name	Axis name lies outside the range permitted in system parameter Number of axes
Instruction not known	Incorrect character
Instruction not possible	This instruction is not possible in this program type or with this configuration
CASE instruction, incorrect character	Incorrect characters in the CASE statement
CASE instruction, no Else branch	CASE statement has not been terminated with the ELSE instruction
CASE instruction, no open	Instructions of a CASE statement have been found with no CASE open (e.g.: CASE.JMP.N4)
Input instruction expected	An input instruction must follow in this instruction at this point
No end instruction in program	The program is not terminated with an "END"
Integer expected	An integer must follow in this instruction at this point
Identical axes names expected	Identical axes names must follow in this instruction
No subsequent instruction in this line	There may be no further instruction in this program line after this instruction
Status invalid	Only status '0' or status '1' are permitted when assigning or scanning inputs, flags or outputs
Blank expected	A blank must follow at this point in this instruction (or instruction strings)

Error messages in the program editor during the syntax check (continued)

Error text:	Description/Explanation:
N expected	An 'N' must follow at this point in this instruction
Name expected	A valid name (program or jump label) must follow in this instruction at this point
Positive number expected	A positive number must follow in this instruction at this point
Program write-protected	The program has the write-protected attribute and cannot be modified (cancel write-protected attribute)
Period expected	There must a period in this instruction at this point
Checksum error	A checksum error has been established when loading the program into the program editor. The program cannot be loaded correctly and must be deleted
R expected	There must an "R" for register in this instruction at this point
R, I, O or M expected	There must be an 'R', 'I', 'O' or 'M' in this instruction at this point
Real number expected	There must a real number in this instruction at this point, but this real number may not exceed the range in the parameter value
Double jump label	This jump label occurs several times in this program
Jump label not first character	The jump label must be at the first position of a program line
Jump label not found	This jump label does not exist in this program
Number out of range	The number lies outside of the scheduled range (parameter value, maximum numerical value limit)
Number expected	There must be a number in this instruction at this point
Line too long	The program line may not be longer than 35 characters

7.1.3 Status messages from the operating system

The operating system may output the following messages "SE=n". The error message can be reset only by switching the unit off.

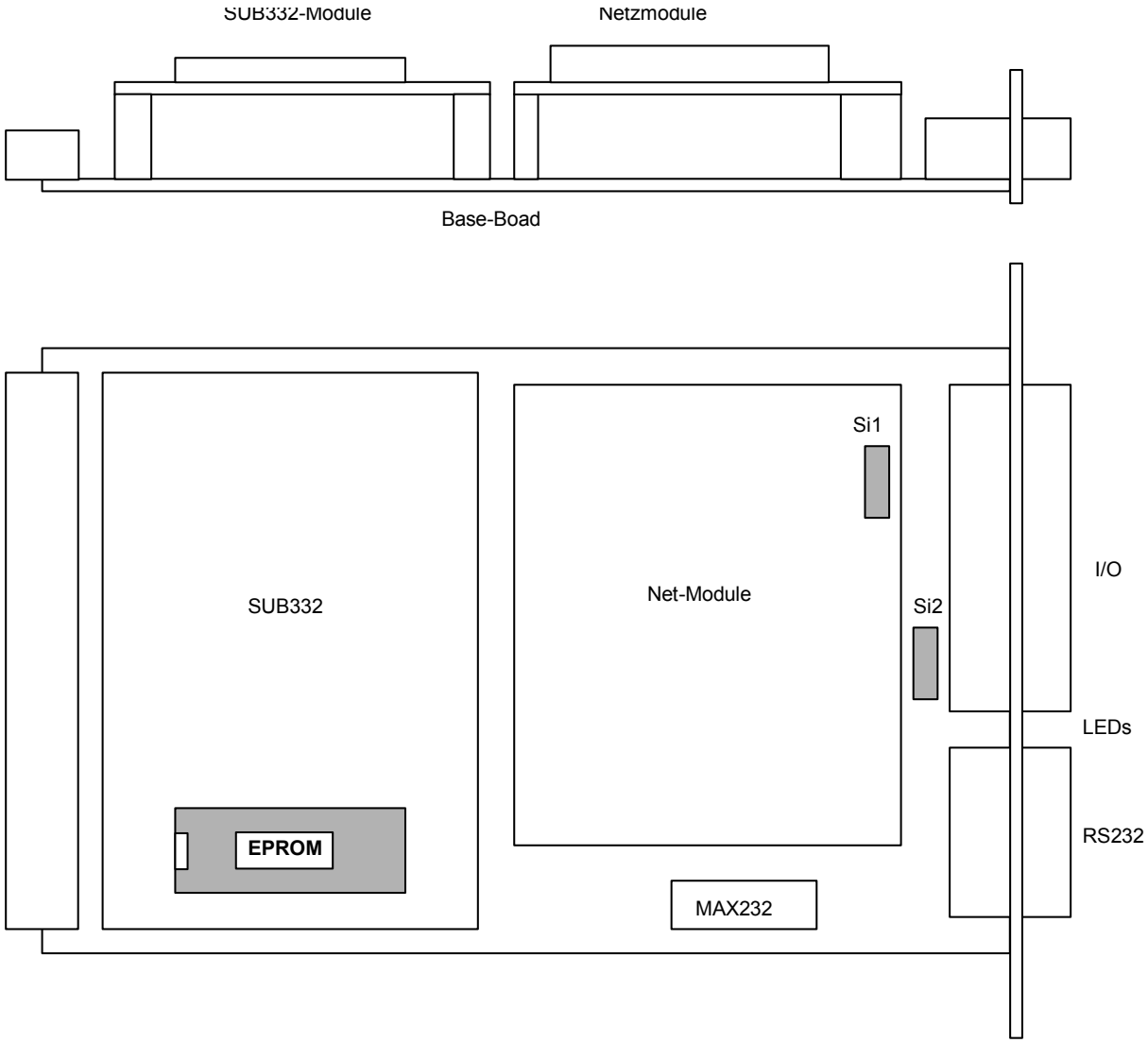
n = Error No.:	Error type:	Cause of error:
2	Bus error	Address does not respond during asynchronous access
3	Address error	Access to odd addresses may occur during arithmetic operations
4	Illegal instruction	Defective EPROM or problem relating to subroutine return
5	Division by 0	Register values out of range
6	CHK	Not in PA-CONTROL
7	Trapv	Not in PA-CONTROL
8	PRIV	Not in PA-CONTROL
9	Trace	Not in PA-CONTROL
A-D	Instruction not implemented	May occur as a consequence of error number 3
E	Illegal format	Not in PA-CONTROL
F	Interrupt not initialized	Not in PA-CONTROL
G	Interrupt error	Not in PA-CONTROL
H-N	Autovector not initialized	Not in PA-CONTROL
O	Non-autovector not initialized	Not in PA-CONTROL

7.2 CPU-Board PIO-1

The CPU-Board PIO-1 is to consist of 3 modules:

- Basic-Board
- SUB332-Module
- Net-Module

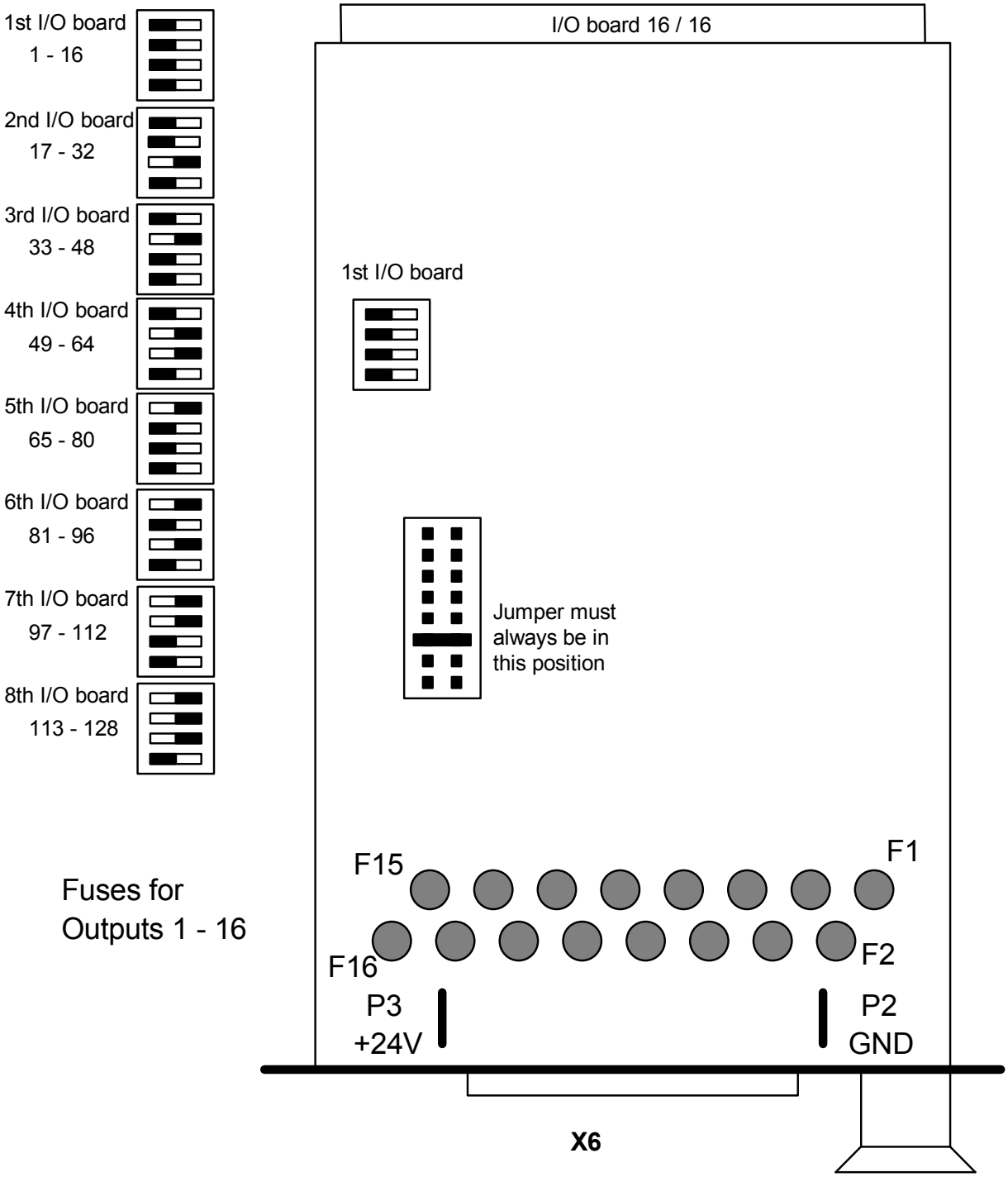
Overview:



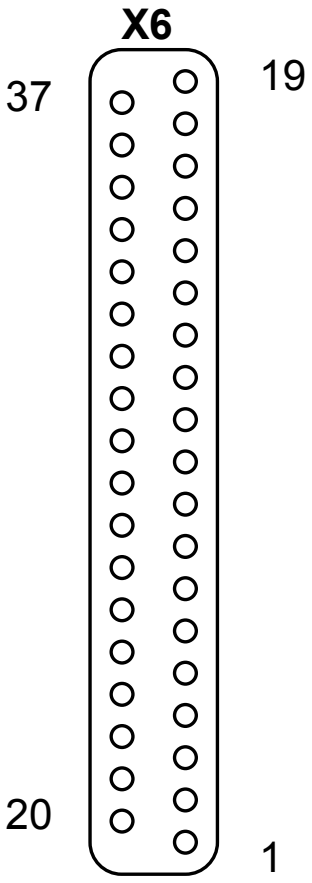
	Function	Value
Fuse 1 (Si1)	Protected by fuse of the 5V-Power supply for the SUB 332-Modul	3A
Fuse 2 (Si2)	Protected by fuse of the 24V; I/O	2A

7.3 I/O board 16/16

7.3.1 I/O board addressing



7.3.2 Connector pin assignment I/O board



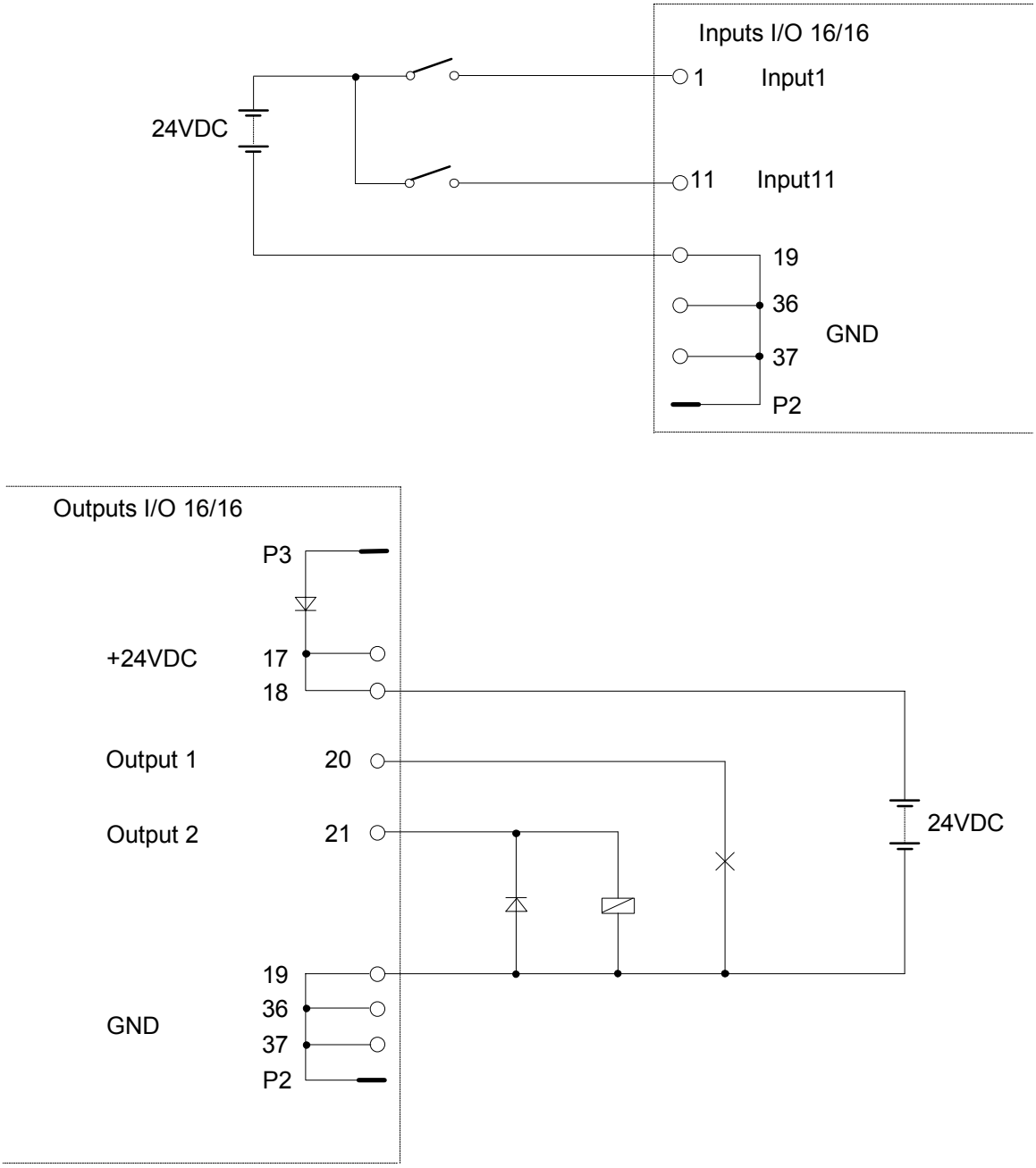
1	Input	1	20	Output	1
2	"	2	21	"	2
3	"	3	22	"	3
4	"	4	23	"	4
5	"	5	24	"	5
6	"	6	25	"	6
7	"	7	26	"	7
8	"	8	27	"	8
9	"	9	28	"	9
10	"	10	29	"	10
11	"	11	30	"	11
12	"	12	31	"	12
13	"	13	32	"	13
14	"	14	33	"	14
15	"	15	34	"	15
16	"	16	35	"	16
17	+24VDC		36	GND	
18	+24VDC		37	GND	
19	GND				

Sub-D 37-pin socket

Signal input:
- Opto-decoupled
- 24VDC
- Typical input current 5mA
- Low level 0 - 3VDC
- High level 10 - 30VDC

Signal output:
- Opto-decoupled
- Positiv switching
- 24VDC / 0,5A (ohmic load)
- Max. 2A per board

7.3.3 Connection example



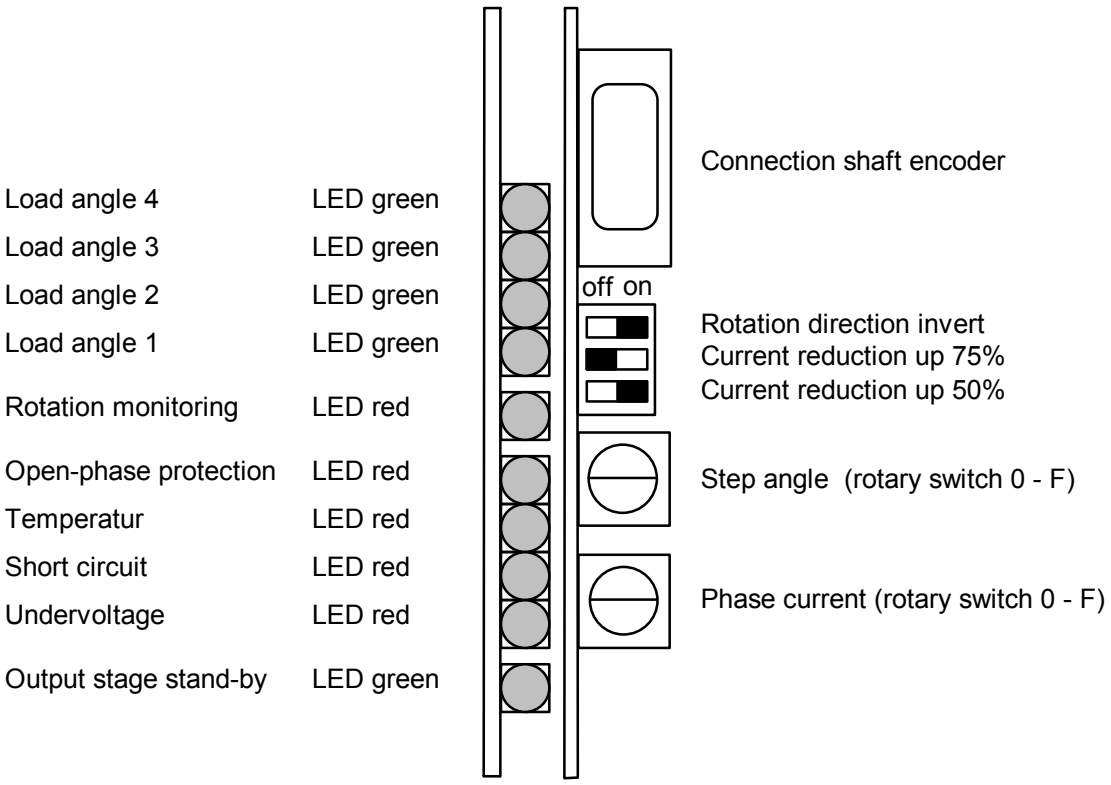
7.4 2-phase power output stage LE12-140

The PAC-SC Compact can controlled an 2-phase stepping motor. The PAC-SC Compact is equipped with an 2-phase power output stage LE12-140.

The features of these power output stages are as follows:

- Constant current drive in chopper mode
- Current setting with potentiometer
- Protection against motor power supply overvoltage and undervoltage
- Standby and fault indication by means of LEDs
- Temperature monitoring
- Current reduction at standstill
- Inputs and outputs are opto-decoupled

Front view:



Setting step angle:

Step angle	200	400	800	1600 */**					500	1000
Switchposition	0	1	2	3*/**	4	5	6	7	8	9

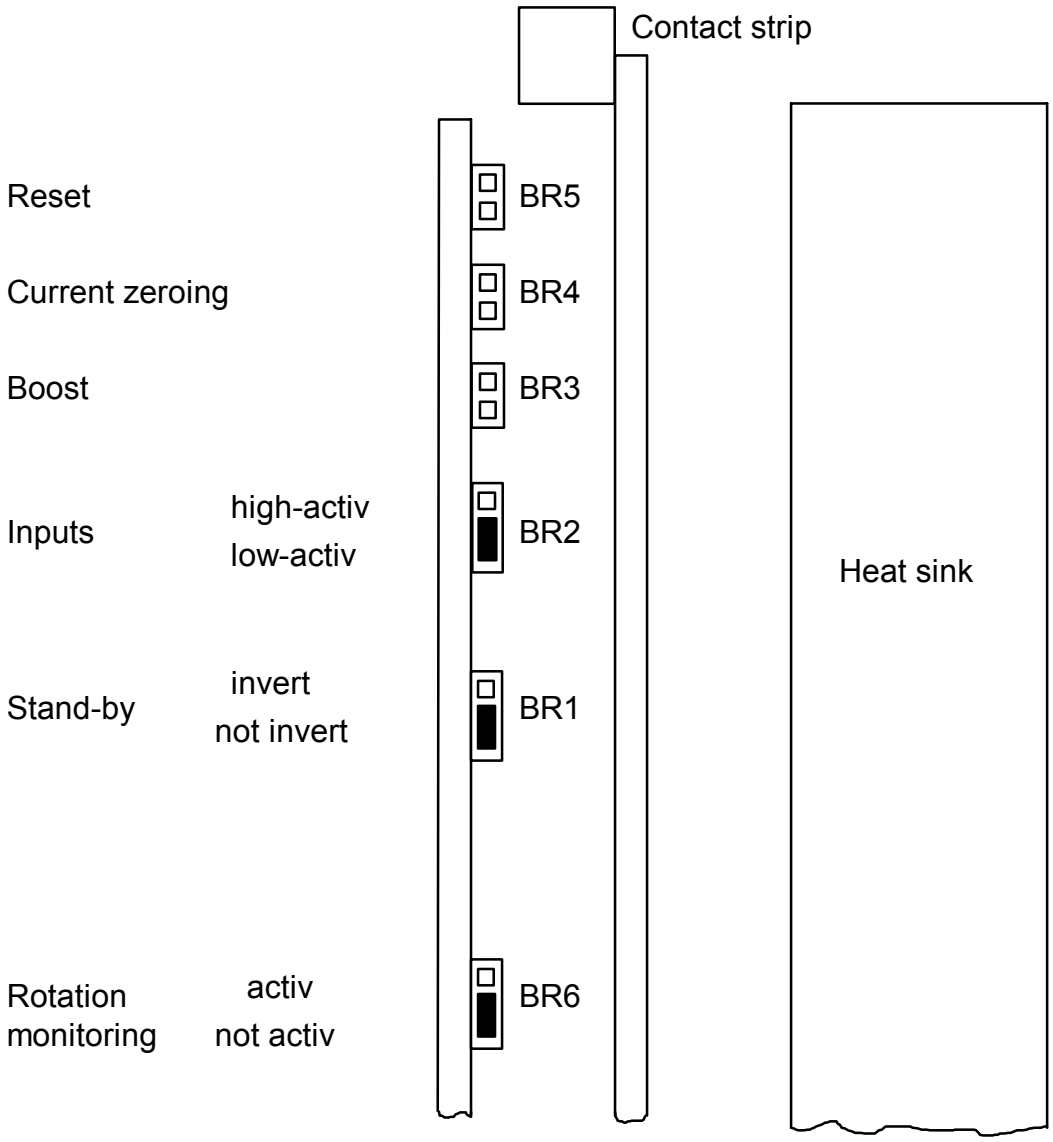
* Rotation monitoring by this step angle not available
** In preparing

Setting phase current:

Phase current (A)	1,0	1,7	2,5	3,2	4,0	4,6	5,4	6,1
Switchposition	0	1	2	3	4	5	6	7

Phase current (A)	6,8	7,5	8,3	9,0	9,7	10,4	11,3	12,0
Switchposition	8	9	A	B	C	D	E	F

Jumperposition:



BR1 – BR5 without function

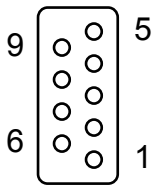
Rotation monitoring:

The rotation monitoring can be activated via the jumper BR6. When the rotation monitoring is active, a rotary transducer with 50 increments per motor revolution must be connected to the plug-in connector "Rotary transducer connection".

A supply voltage of 5 V with up to 150 mA current is available for the rotary transducer. The rotary transducer output signals channels A and B are rectangular signals with a 90° phase offset. They must be 5 V push-pull signals (RS422 agreement).

Rotary transducer connector:

Rotary transducer connector (plug connector seen from the solder side)



1 A
2 +5V / 100mA
3 GND
4 GND
5 $\overline{B}$
6 $\overline{A}$
7 +5V / 100mA
8 GND
9 B

Load angle display:

An assessment of the dynamic state of the drive can be made with the "Load angle 1-4" LEDs. When low demands are placed on the drive system, only the LEDs "Load angle 1" and "Load angle 2" light up when the motor is running. If the LEDs "Load angle 1-3" light up simultaneously, the drive is at the limit of its capabilities.

Rotation monitoring display:

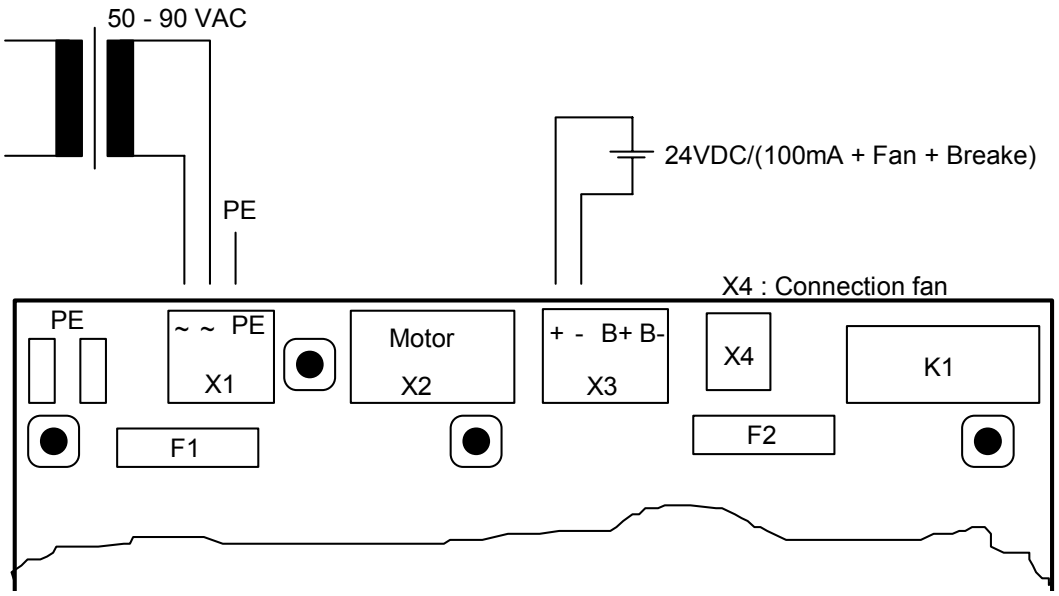
The following is displayed with the red LED "Rotation monitoring":

- If the LED "Rotation monitoring" lights up continuously together with the LEDs "Load angle 2 and 4", the rotation monitoring module signals that the maximum permissible load angle has been exceeded.

- If only the red LED "Rotation monitoring" lights up continuously, the output stage is in the Reset state.

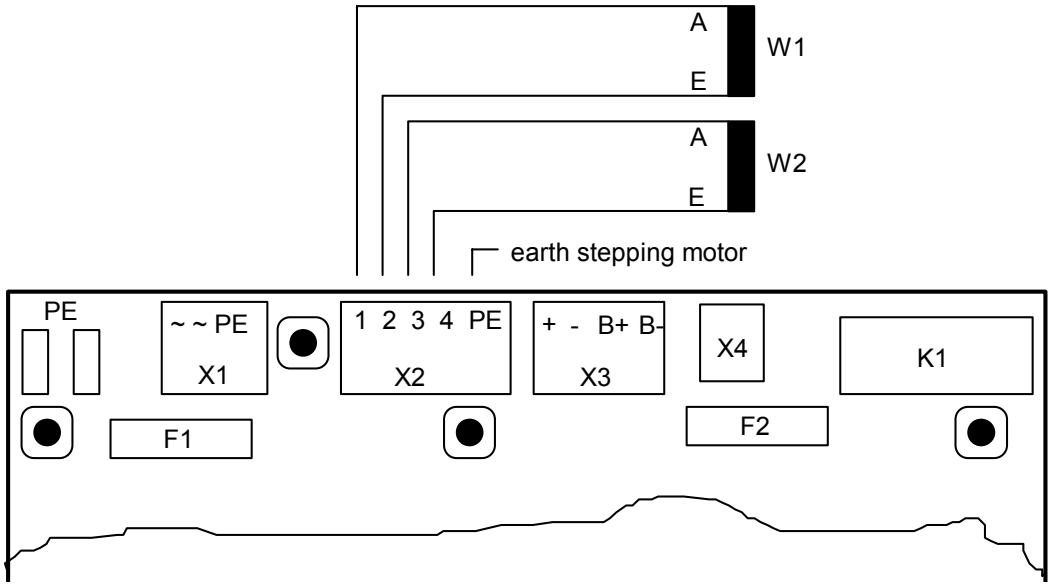
- If the red LED "Rotation monitoring" lights up with the green LEDs "Load angle 1-4" within a run light, the rotation monitoring is not active.

7.5 Connection power supply



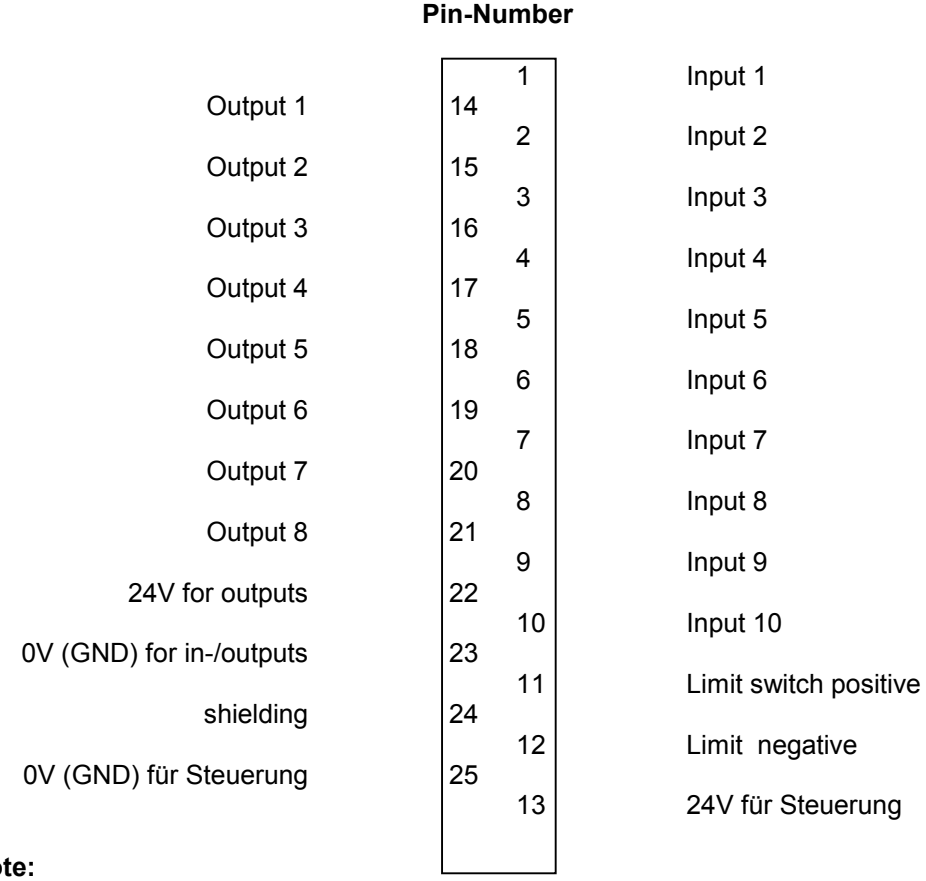
F1	:	Motor supply	:	6,3A
F2	:	Fan supply	:	200mA

7.6 Connection stepping motor 2-Phasen



7.7 Pin assignment 25-pin I/O-Connector

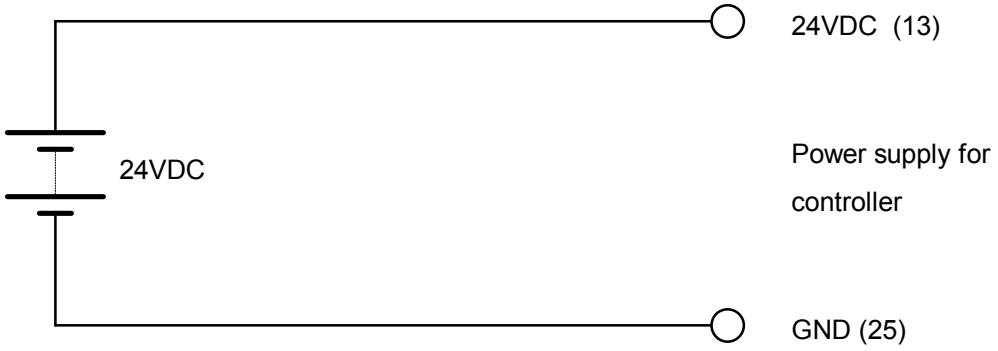
SUB-D-socket:



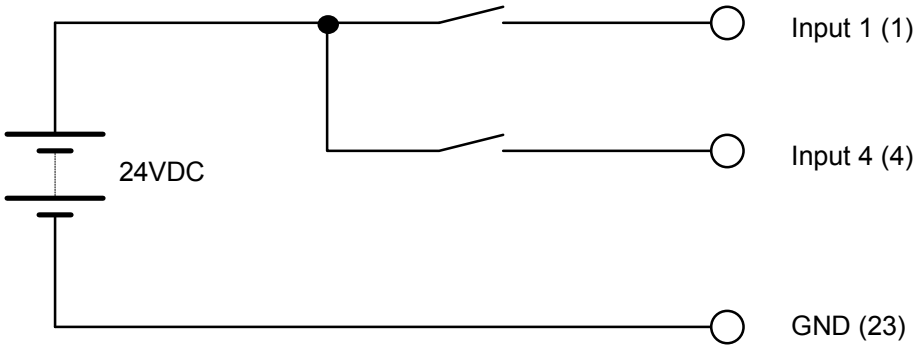
To note:

PIN 23 and PIN 25 are not internal connected!

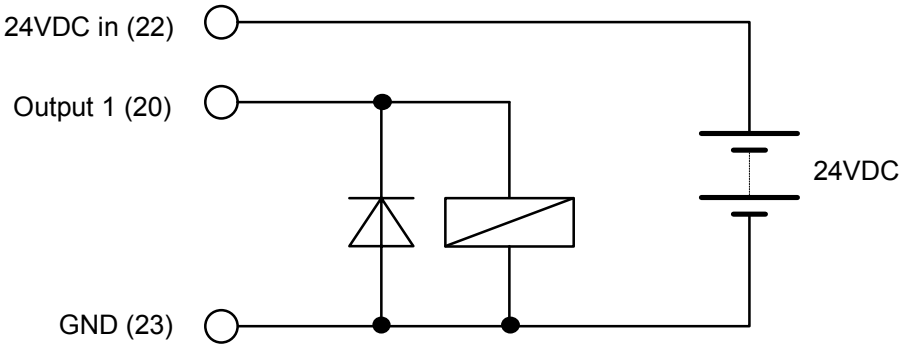
7.7.1 Supply of the controller



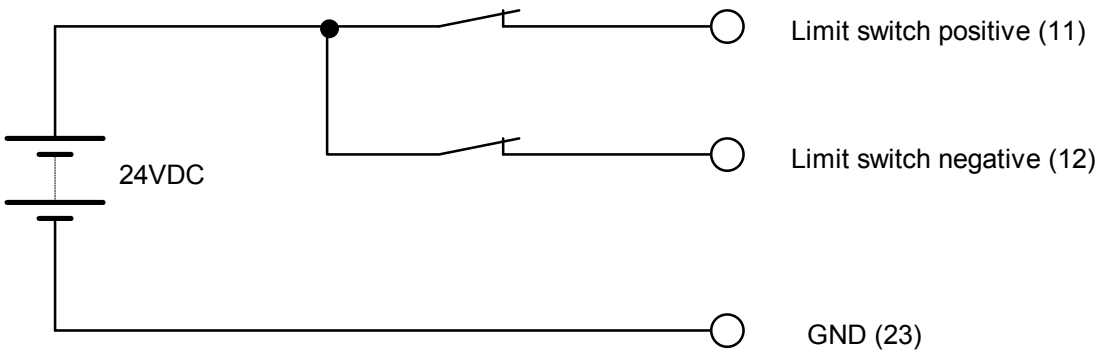
7.7.2 Connection of the inputs



7.7.3 Connection of the outputs



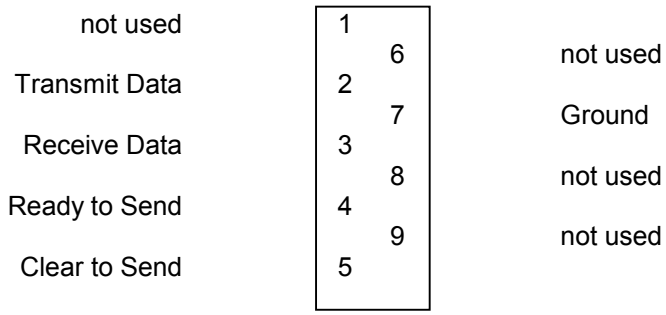
7.7.4 Connection of the limit switch



7.8 Connector pin assignment, serial interface connector

SUB-D-socket

Pin-Number

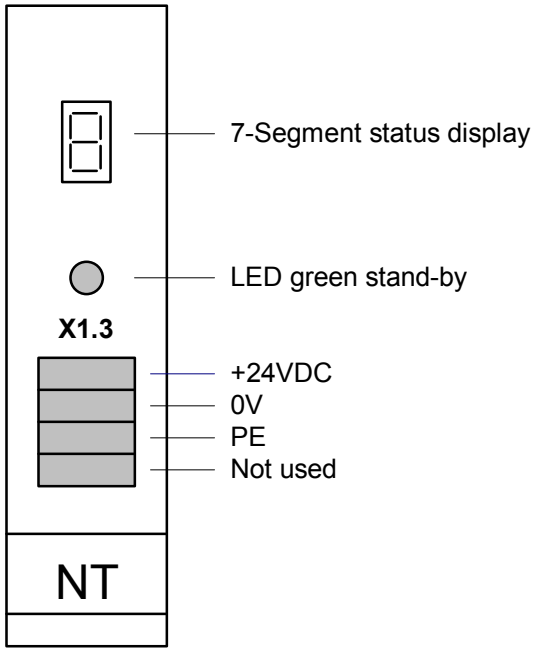


Example:

Connectioncable to PC

PAC-SC		PC (9 polig)		PC (25 polig)
TxD (2)	-	RxD (2)	-	RxD (3)
RxD (3)	-	TxD (3)	-	TxD (2)
GND (7)	-	GND (5)	-	GND (7)

7.9 NT1-5V power supply for µP.System



7.10 Connection between the PC and the PAC

The connection between the PC and the unit of a PAC range is necessary, for example when using the program development system PROPAC. The data exchange and the transfer of programs and parameters (in both directions) is via a serial interface. The PC must therefore be equipped with this interface (COM1 or COM2). The units of the PA control range are prepared as standard for this task.

The cable used must meet the following conditions:

PA-Control (socket = interface1)	**PC (9-pole or 25-pole)**		
(9-pole, Sub-D)	(9-pole, Sub-D	/	25-pole, Sub-D)
TXD (2)----------	RXD (2)	/	RXD (3)
RXD (3)----------	TXD (3)	/	TXD (2)
GND (7)----------	GND (5)	/	GND (7)

Please note:

Assignment does not correspond to standard RS232 !

7.11 PAC-keycode (+ extended ASCII-character set)

Dez	Hex	Sign	PAC Keyboard
0	0		
1	1		
2	2		
3	3	♥	
4	4	♦	
5	5	♣	
6	6	♠	
7	7		
8	8		Shift+DEL
9	9		Ctrl+I
10	A		
11	B		Ctrl+K
12	C		
13	D		ENTER/Ctrl+M
14	E		
15	F		
16	10		
17	11		
18	12	↕	
19	13		
20	14		
21	15	§	
22	16	¦	
23	17		
24	18	↑	
25	19	↓	
26	1A	→	
27	1B	←	ESC/Shift+ESC
28	1C	∟	
29	1D	↔	
30	1E		
31	1F		
32	20		Space
33	21	!	Shift+9
34	22	„	Shift+2
35	23	#	
36	24	$	Shift+4
37	25	%	Shift+5
38	26	&	Shift+6
39	27	‘	
40	28	(	Shift+8
41	29	)	Shift+9
42	2A	*	Shift++
43	2B	+	+
44	2C	,	,
45	2D	-	-
46	2E	.	.

Dez	Hex	Sign	PAC Keyboard
47	2F	/	Shift+7
48	30	0	0
49	31	1	1
50	32	2	2
51	33	3	3
52	34	4	4
53	35	5	5
54	36	6	6
55	37	7	7
56	38	8	8
57	39	9	9
58	3A	:	Shift+ .
59	3B	;	Shift+ ,
60	3C	<	<
61	3D	=	Shift+0
62	3E	>	Shift+<
63	3F	?	Shift+3
64	40	@	
65	41	A	Shift+A
66	42	B	Shift+B
67	43	C	Shift+C
68	44	D	Shift+D
69	45	E	Shift+E
70	46	F	Shift+F
71	47	G	Shift+G
72	48	H	Shift+H
73	49	I	Shift+I
74	4A	J	Shift+J
75	4B	K	Shift+K
76	4C	L	Shift+L
77	4D	M	Shift+M
78	4E	N	Shift+N
79	4F	O	Shift+O
80	50	P	Shift+P
81	51	Q	Shift+Q
82	52	R	Shift+R
83	53	S	Shift+S
84	54	T	Shift+T
85	55	U	Shift+U
86	56	V	Shift+V
87	57	W	Shift+W
88	58	X	Shift+X
89	59	Y	Shift+Y
90	5A	Z	Shift+Z
91	5B	[	
92	5C	\	
93	5D	]	

Dez	Hex	Sign	PAC Keyboard
94	5E	^	
95	5F	_	Shift+ -
96	60	´	
97	61	a	A
98	62	b	B
99	63	c	C
100	64	d	D
101	65	e	E
102	66	f	F
103	67	g	G
104	68	h	H
105	69	i	I
106	6A	j	J
107	6B	k	K
108	6C	l	L
109	6D	m	M
110	6E	n	N
111	6F	o	O
112	70	p	P
113	71	q	Q
114	72	r	R
115	73	s	S
116	74	t	T
117	75	u	U
118	76	v	V
119	77	w	W
120	78	x	X
121	79	y	Y
122	7A	z	Z
123	7B	{	
124	7C	|	
125	7D	}	
126	7E	~	
127	7F	∆	
128	80	Ç	
129	81	ü	
130	82	é	
131	83	â	
132	84	ä	
133	85	à	
134	86	å	
135	87	ç	
136	88	ê	
137	89	ë	
138	8A	è	
139	8B	ï	
140	8C	î	

Dez	Hex	Sign	PAC Keybord
141	8D	ì	
142	8E	Ä	
143	8F	Å	
144	90	É	
145	91	æ	
146	92	Æ	
147	93	ô	
148	94	ö	
149	95	ò	
150	96	û	
151	97	ù	
152	98	ÿ	
153	99	Ö	
154	9A	Ü	
155	9B	¢	
156	9C	£	
157	9D	¥	
158	9E	₱	
159	9F	ƒ	
160	A0		
161	A1	í	
162	A2	ó	
163	A3	ú	
164	A4	ñ	
165	A5	Ñ	
166	A6	ª	
167	A7	º	
168	A8	¿	
169	A9		
170	AA	¬	
171	AB	½	
172	AC	¼	
173	AD	¡	
174	AE	«	
175	AF	»	
176	B0	¦	
177	B1	¦	
178	B2	¦	
179	B3	³	
180	B4	´	
181	B5	µ	
182	B6	¶	
183	B7	·	
184	B8	¸	
185	B9	¹	
186	BA	º	
187	BB	»	

Dez	Hex	Sign	PAC Keyboard
188	BC	¼	
189	BD	½	
190	BE	¾	
191	BF	¿	
192	C0	À	
193	C1	Á	
194	C2	Â	
195	C3	Ã	
196	C4	Ä	
197	C5	Å	
198	C6	Æ	
199	C7	Ç	
200	C8	È	
201	C9	É	
202	CA	Ê	
203	CB	Ë	
204	CC	Ì	
205	CD	Í	
206	CE	Î	
207	CF	Ï	
208	D0	Ð	
209	D1	Ñ	
210	D2	Ò	
211	D3	Ó	
212	D4	Ô	
213	D5	Õ	
214	D6	Ö	
215	D7	×	
216	D8	Ø	
217	D9	Ù	
218	DA	Ú	
219	DB	¦	
220	DC	_	
221	DD	¦	
222	DE	¦	START
223	DF	¯	
224	E0	a	
225	E1	ß	
226	E2	G	
227	E3	p	
228	E4	S	
29	E5	s	
230	E6	µ	
231	E7	t	
232	E8	F	
233	E9	T	

Dez	Hex	Sign	PAC Keyboard
234	EA	O	
235	EB	d	
236	EC	8	
237	ED	f	
238	EE	e	
239	EF	n	
240	F0	=	
241	F1	±	
242	F2	=	
243	F3	=	
244	F4	(	
245	F5	)	
246	F6	÷	
247	F7	~	
248	F8	°	
249	F9	·	
250	FA	·	
251	FB	v	
252	FC	n	
253	FD	²	
254	FE	¦	
255	FF		

Completion

Dez	Hex	PAC Keyboard
294	126	Alt+L
327	147	Shift+Arrow left
328	148	Arrow up Alt+ Arrow up
329	149	Shift+Pfeil auf
331	14B	Arrow left Alt+ Arrow left
333	14D	Arrow right Alt+ Arrow right
335	14F	Shift+Arrow right
336	150	Arrow down Alt+ Arrow down
337	151	Shift+Arrow down
338	152	INS Ctrl, Alt, Shift+INS
339	153	DEL / Alt+DEL
371	173	Ctrl+ Arrow left
372	174	Ctrl+ Arrow right
397	18D	Ctrl+DEL

7.12 Pro - Demo

Example programs for the PA-Control controls:

The demonstration diskette with the examples is included with all PA-CONTROL operating instructions.
Please refer to the directory on the diskette for the list of demonstration programs.

7.13 Accessory list

	Order number Item number
Original version PROPAC (3,5" & 5,25" Disk) including manual	#230850
Connection cable PC-XT for the PA (lenght 3m)	#230929
Connection cable PC-AT for the PA (lenght 3m)	#230920

Technical annex