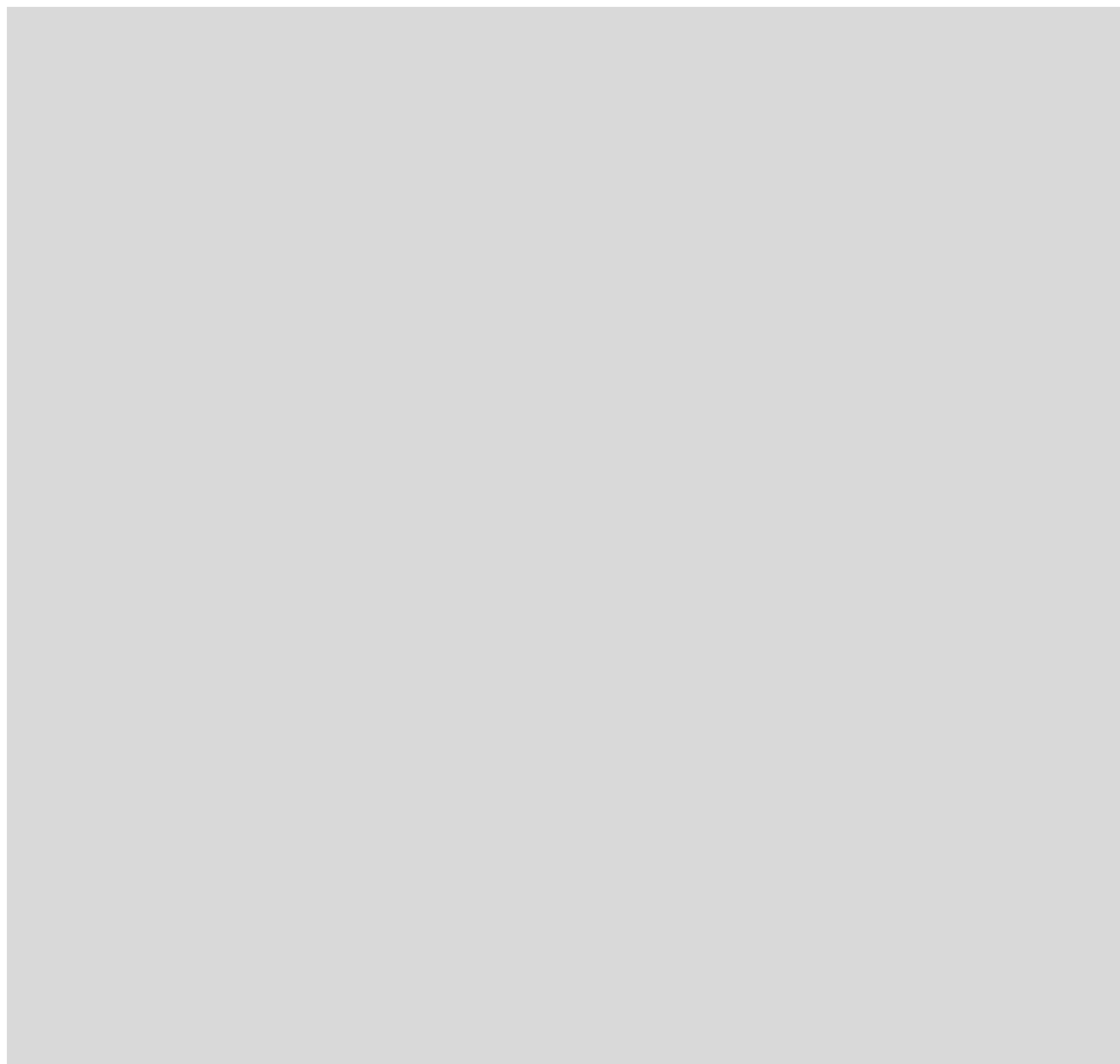


rho 3

# Releases Part 1 Versions TO01, TO02 and TO03



Version

# 107



*rho 3*

# Releases Part 1 Versions T001, T002 and T003

1070 073 078-107 (97.01) GB



Reg. Nr. 16149-03

© 1995 - 1997

by Robert Bosch GmbH,

All rights reserved, including applications for protective rights.  
Reproduction or handing over to third parties are subject to our written permission.

Discretionary charge 50.– DM

## Contents Part 1

	Page
<b>1</b>	<b>Safety notes</b>
	regarding rho 3 operations ..... 1 – 1
<b>2</b>	<b>Changes to the PHG -</b>
	<b>Operation and Display</b> ..... 1 – 2
<b>3</b>	<b>Machine parameters</b> ..... 1 – 3
<b>4</b>	<b>Signals and PLC functions</b> ..... 1 – 4
<b>5</b>	<b>Testing BAPS2 programs</b>
	<b>with the PHG</b> ..... 1 – 5
<b>6</b>	<b>Other changes</b> ..... 1 – 6
<b>7</b>	<b>Machine status display</b> ..... 1 – 7
7.1	File "MSD.DAT" ..... 1 – 8
7.2	MSD interface assignment ..... 1 – 9
7.3	Restrictions and error messages ..... 1 – 9
<b>8</b>	<b>Coded text output</b> ..... 1 – 10
8.1	File "TEXT.DAT" ..... 1 – 11
8.2	Interface assignment, coded text output ..... 1 – 12
8.3	Limitations and error messages ..... 1 – 12

## Contents TO02F

<b>1</b>	<b>General notes on using TO02F software</b> .....	<b>2 – 1</b>
<b>2</b>	<b>High-speed measuring</b>	
2.1	Programming .....	2 – 2
2.1.1	Program example .....	2 – 4
2.2	Hardware requirements .....	2 – 5
2.3	Machine parameters .....	2 – 6
2.4	Restrictions .....	2 – 6
2.5	Achievable accuracies .....	2 – 7
<b>3</b>	<b>Interruptible axes</b>	
3.1	RC input signals .....	2 – 8
3.2	Control-internal effect of the DRIVE-ON signals .....	2 – 8
3.3	Applications .....	2 – 9
3.3.1	Manual mode .....	2 – 9
3.3.2	Automatic mode .....	2 – 9
3.3.3	Safety notes .....	2 – 9
3.3.4	Example 1 for using DRIVE_ON .....	2 – 9
3.3.5	Example 2 for using DRIVE-ON .....	2 – 11
3.3.6	PLC program for examples 1 and 2 (segment) .....	2 – 12
3.4	Monitoring functions .....	2 – 12
3.5	Example 3 (PLC program) .....	2 – 13
<b>4</b>	<b>Fast, smooth start-off</b>	
4.1	Programming .....	2 – 14
4.2	Example: .....	2 – 16
<b>5</b>	<b>Setting the belt counter (special function 28)</b>	
5.1	Example: .....	2 – 17
5.2	Belt simulation .....	2 – 18

<b>6</b>	<b>Spatial passing, optimum processing</b> .....	<b>2 – 19</b>
6.1	Spatial passing for program slopes .....	2 – 20
6.1.1	PTP passing for program slopes .....	2 – 20
6.1.2	Path passing for program slope .....	2 – 22
6.1.3	Passing when changing the type of interpolation .....	2 – 22
6.1.4	Special cases .....	2 – 22
6.2	Passing for block slopes .....	2 – 24
6.2.1	PTP passing for block slopes .....	2 – 24
6.2.2	Path passing for block slopes .....	2 – 25
6.2.3	Passing when changing the type of interpolation .....	2 – 25
6.3	BAPS2 - Language elements .....	2 – 26
6.3.1	Programming .....	2 – 26
6.4	Machine parameters .....	2 – 27
6.5	Restrictions .....	2 – 27
<b>7.</b>	<b>Protocol 3964/R</b>	
7.1.	Implementing the protocol driver .....	2 – 28
7.2.	Interfacing with the BAPS program .....	2 – 30
7.3.	WRITE device .....	2 – 31
7.4.	READ device .....	2 – 31
7.5.	Special cases and restrictions .....	2 – 32
<b>8.</b>	<b>Multiple use of the serial interfaces</b>	
8.1.	Automatic reactions to a conflict .....	2 – 33
8.2.	Controlled avoidance of conflicts .....	2 – 34
8.3.	Replugging or reswitching the interface .....	2 – 34
8.4.	Status display .....	2 – 35
8.5.	Example .....	2 – 35
8.6.	Notes on implementation .....	2 – 36
<b>9</b>	<b>CONDITION inquiry of interfaces</b>	
9.1	Compiler instruction SER_IO_STOP .....	2 – 37
9.2	CONDITION function and interface .....	2 – 38
9.3	CONDITION function and file .....	2 – 39
<b>10</b>	<b>Standard procedures</b>	
10.1	Standard procedure INT_ASC .....	2 – 40
10.2	Standard procedure ASC_INT .....	2 – 41
<b>11</b>	<b>PUBLIC variable</b> .....	<b>2 – 42</b>

<b>12</b>	<b>Kinematic-related automatic - manual mode</b>	
12.1	New Signals and their meanings .....	2 – 43
12.1.1	Signal meanings .....	2 – 44
12.2	Automatic mode .....	2 – 44
12.3	Referencing and Set-up .....	2 – 44
12.4	Teach in .....	2 – 45
12.5	Testing .....	2 – 45
12.6	Changing from global AUTOMATIC/MANUAL mode to AUTOMATIC/MANUAL MODE per kinematics .....	2 – 45
12.7	Notes on proper use of options .....	2 – 45
12.8	Diagnosis on the PHG .....	2 – 45
<b>13</b>	<b>External program/process deselection</b>	
13.1	RC input signals .....	2 – 46
13.2	RC output signals .....	2 – 47
13.3	Restrictions .....	2 – 47
13.4	EXPROG.DAT structure .....	2 – 47
<b>14</b>	<b>Setting potentiometer values</b>	
14.1	Description of function .....	2 – 48
14.1.1	Option byte=0 .....	2 – 49
14.1.2	Option byte<>0 .....	2 – 49
<b>15</b>	<b>INFO function on the PHG</b>	
15.1	Availability of the INFO function .....	2 – 51
15.2	Restrictions .....	2 – 52
15.3	Operation .....	2 – 52
15.4	Possible error states, Info texts .....	2 – 53
15.5	Output on the PHG .....	2 – 56
<b>16</b>	<b>Coded error output</b>	
16.1	Structuring the errors in groups .....	2 – 57
16.2	Address assignment of the error code signals .....	2 – 59
16.3	Coded error output of runtime errors .....	2 – 59
16.3.1	Principle of the output of error messages .....	2 – 59
16.3.2	Output of several error codes .....	2 – 60
16.3.3	Parity .....	2 – 60
16.3.4	Response for successive errors .....	2 – 60
16.4	Coded error output of system errors .....	2 – 61
<b>17.</b>	<b>ROPS3/IQPRO Version W1J</b>	
17.1.	Compatibility between ROPS 3 – rho 3 .....	2 – 62

## Software version TO03

<b>1</b>	<b>General notes on software version TO03</b>	
1.1	System process priority .....	3 – 1
1.2	Bit coupler with CAN drive interface .....	3 – 1
1.3	25MHz CPU module .....	3 – 1
1.4	Intelligent modular servo board .....	3 – 1
<b>2</b>	<b>Direct function selection with the PHG .....</b>	<b>3 – 2</b>
2.2	Restrictions, handling errors .....	3 – 5
<b>3</b>	<b>Displaying BAPS2 source texts in the TEST mode</b>	
3.1	Writing a program line (Mode = 5.11) .....	3 – 6
3.2	Displaying BAPS2 source text in single-step mode (Mode = 5.9) .....	3 – 7
<b>4</b>	<b>New or modified machine parameters</b>	
4.1	Modified machine parameters .....	3 – 8
4.1.1	Previous meanings of the machine parameters .....	3 – 8
4.1.2	Modified meanings of the machine parameters .....	3 – 8
4.2	New machine parameters .....	3 – 8
4.3	Compatibility of machine parameters .....	3 – 9
4.4	Parameter input for singleturn absolute encoder .....	3 – 9
<b>5</b>	<b>Servodyn-G(C) software limit switch</b>	
5.1	Function .....	3 – 10
5.2	Setting rule .....	3 – 10
<b>6</b>	<b>Referencing with Servodyn-G(C) drive systems</b>	
6.1	Setting referencing modes .....	3 – 12
6.2	REF- MODE 0 ("Normal") .....	3 – 13
6.2.1	Function .....	3 – 13
6.2.2	Setting the reference point switch .....	3 – 13
6.3	REF- MODE 1 ("Correct orientation") .....	3 – 14
6.3.1	Function .....	3 – 14
6.3.2	Setting the reference point switch .....	3 – 15
6.4	REF- MODE 2 ("Resolver suitable") .....	3 – 16
6.4.1	Function .....	3 – 16
6.4.2	Setting the reference point switch .....	3 – 16
6.5	REF- MODE 3 (Combination of 1 and 2) .....	3 – 17
6.5.1	Function .....	3 – 17
6.5.2	Setting the reference point switch .....	3 – 17
<b>7</b>	<b>Reversing the direction of rotation for the Servodyn-G(C) .</b>	<b>3 – 18</b>
<b>8</b>	<b>Configuration recognition for the PLC program</b>	
8.1	Description .....	3 – 19

<b>9</b>	<b>Interface signal error acknowledgement</b>	
9.1	Description of function .....	3 – 21
9.2	RC signals .....	3 – 21
9.3	Restrictions, handling errors .....	3 – 21
<b>10</b>	<b>Counter and PIC timer expansion .....</b>	<b>3 – 22</b>
10.1	Flag assignment of PIC250 timers and counters .....	3 – 23
<b>11</b>	<b>Special function 3, Setting machine position</b>	
11.1	Declaration .....	3 – 24
11.2	How the function for setting machine position works .....	3 – 25
<b>12</b>	<b>Expansion of special functions 1 (IOL) and 2 (PPO) .....</b>	<b>3 – 27</b>
12.1	Response to interface signals and system states .....	3 – 28
<b>13</b>	<b>BAPS2 - Language expansion</b>	
13.1	CASE – instruction .....	3 – 29
13.2	STANDARD - CONSTANTS .....	3 – 31
13.2.1	Standard - Constant CLS (clear screen) .....	3 – 31
13.2.2	Standard - Constant VERSION .....	3 – 31
13.3	Constant declarations .....	3 – 31
<b>14.</b>	<b>ROPS3/IQPRO Version W1L</b>	
14.1.	Compatibility with ROPS3 .....	3 – 32



## Contents Part 2

### Software version T004

<b>1</b>	<b>General notes on software version T004</b>	
1.1	CP/MEM5 module .....	4 – 1
1.2	30MHz CPU module .....	4 – 1
1.3	25 MHz CPU module .....	4 – 1
<b>2</b>	<b>Expansion of the function "Fast, smooth start-off" .....</b>	<b>4 – 3</b>
<b>3</b>	<b>Directly approaching points in Teach IN</b>	
3.1	Operation .....	4 – 4
3.2	Preventing points which have been taught in from being overwritten .....	4 – 5
3.3	Restrictions .....	4 – 6
<b>4</b>	<b>Saving the user memory in the EEPROM</b>	
4.1	Requirements .....	4 – 7
4.2	Retrieving the user memory from the EEPROM .....	4 – 8
4.3	Marginal conditions .....	4 – 8
<b>5</b>	<b>Selecting PHG functions using interface signals</b>	
5.1	Description of function .....	4 – 9
5.1.1	Selection .....	4 – 9
5.2	Allocation file .....	4 – 9
5.2.1	Allocation file structure .....	4 – 10
5.3	Deselection, return to the basic level .....	4 – 10
5.4	Interface assignment .....	4 – 11
5.5	Restrictions, handling errors .....	4 – 11
<b>6.</b>	<b>Displaying "high-speed inputs" on the PHG</b>	
6.1.	Selecting the display on the PHG .....	4 – 12
6.1.1	Structure and contents of the display .....	4 – 13
<b>7</b>	<b>Changes to the machine parameter input process</b>	
7.1	Entering a password .....	4 – 14
7.2	Scrolling when selecting a group .....	4 – 14
<b>8</b>	<b>New parameters starting with version T004A .....</b>	<b>4 – 15</b>
8.1	Displaying set options .....	4 – 17
<b>9</b>	<b>Recording the reference path (movement history)</b>	
9.1	Activating the reference path recording function, special function 29 .....	4 – 18
9.2	Storing the position values in the transformation cycle or rough interpolation cycle (clock pattern) .....	4 – 20
9.3	Deactivating the reference path recording function, special function 30 .....	4 – 21
9.4	Reading the reference path value, special function 31 .....	4 – 22
9.5	Program examples (partial) .....	4 – 24

<b>10</b>	<b>Limiting axis speeds in linear mode</b> .....	<b>4 – 27</b>
10.1	Machine parameters for limiting axis speed .....	4 – 28
10.2	Previous axis speed limits .....	4 – 28
<b>11</b>	<b>Manual axes as endless axes</b> .....	<b>4 – 29</b>
<b>12</b>	<b>The BAPS2 instruction ASSIGN</b>	
12.1	Syntax ASSIGN instruction .....	4 – 31
12.2	ASSIGN instruction .....	4 – 32
12.3	Compatibility .....	4 – 32
12.4	Using ASSIGN on standard channels .....	4 – 32
12.5	Runtime conditions .....	4 – 33
12.6	Compiler instruction FILE_ERROR .....	4 – 33
12.7	Restrictions on selecting file names .....	4 – 34
12.8	Application example for ASSIGN .....	4 – 34
<b>13</b>	<b>Calling operating system functions from the BAPS2 user programs</b>	
13.1	Operating system commands .....	4 – 35
13.2	Declaration part .....	4 – 35
13.3	Commands .....	4 – 37
13.3.1	Compiling BAPS programs .....	4 – 37
13.3.2	Copying files .....	4 – 37
13.3.3	Deleting files .....	4 – 38
13.3.4	Starting processes .....	4 – 38
13.3.5	Stopping processes .....	4 – 38
13.4	Example .....	4 – 39
13.5	Restrictions, error treatment .....	4 – 41
<b>14</b>	<b>Belt type selection, special function 21</b>	
14.1	Declaring special function 21 .....	4 – 42
14.2	Using special function 21 .....	4 – 43
14.3	Notes and error messages regarding special function 21 .....	4 – 44
<b>15</b>	<b>Belt synchronization without the robot being able to move parallel to the belt</b> .....	<b>4 – 45</b>
15.1	Operating conditions for belt synchronization type 2 .....	4 – 46
15.2	Special function for parameterisation of the belt characteristics .....	4 – 47
15.3	Belt stop/belt start for belt synchronization type 2 .....	4 – 49
15.3.1	Meaning of the threshold value for the belt status "Belt stopped" .....	4 – 49
15.3.2	Meaning of the threshold value for the belt status "Belt running" .....	4 – 50
15.4	Bypassing the error message "Ax velocity exceeded" .....	4 – 50
15.5	General notes on synchronization type 2 .....	4 – 50
15.6	Restrictions due to this type of belt synchronization in conjunction with other options .....	4 – 51
15.7	Program example .....	4 – 52
<b>16</b>	<b>Belt synchronization type 3 (cam interpolation)</b>	
16.1	Operating conditions for belt synchronization type 3 .....	4 – 54
16.2	Description of the parameterized curve .....	4 – 56
16.3	Programming the sections .....	4 – 56
16.4	Special cases .....	4 – 57
16.4.1	Comparison with normal traverse blocks .....	4 – 57
16.5	Program example .....	4 – 58

**17**      **Several axes on one setpoint output** ..... **4 – 60**  
17.1      Setting machine parameters ..... 4 – 61

**18**      **ROPS3/IQPRO**  
18.1      Compatibility list: ..... 4 – 62

## Description of software extensions TO05

<b>1</b>	<b>Generating the PHG display</b> .....	<b>5 – 1</b>
1.1	READ PHG expansion .....	5 – 2
<b>2</b>	<b>Fast position controller on intelligent servo boards</b>	
2.1	Activation of the software function .....	5 – 3
<b>3</b>	<b>Asynchronous speed and target position stipulation</b>	
	<b>Special functions 41, 42</b>	
3.1	Required options .....	5 – 4
3.2	Modified "MOVE UNTIL" instruction MOVE UNTIL 'probe input' .....	5 – 4
3.3	Asynchronous distance stipulation .....	5 – 5
3.4	Asynchronous speed stipulation .....	5 – 5
3.5	Restrictions .....	5 – 5
<b>4</b>	<b>Writing, reading of binary files</b> .....	<b>5 – 7</b>
4.1	Read operations .....	5 – 8
4.2	Write operations .....	5 – 9
4.3	Data format .....	5 – 9
4.4	Restrictions .....	5 – 10
<b>5</b>	<b>Move_file</b>	
5.1	Configuration of the binary file .....	5 – 11
5.1.1	File header .....	5 – 11
5.1.2	Data division .....	5 – 12
5.2	The special function MOVE_FILE .....	5 – 14
5.2.1	Opening the BNR file .....	5 – 15
5.3	Integrity checks .....	5 – 17
<b>6</b>	<b>CAN bus for 12 axes SERVODYN-GC</b>	
6.1	Variable assignment of the CAN interface .....	5 – 20
6.2	Possible configurations of the CAN module .....	5 – 20
6.3	Axis control via CAN .....	5 – 21
6.4	Transmission rate .....	5 – 21
6.5	Restrictions .....	5 – 21
6.6	Digital I/O via CAN .....	5 – 22
6.7	Possible configurations .....	5 – 22
6.8	Transmission rate .....	5 – 22
6.9	Machine parameters .....	5 – 23
6.9.1	Fixing the axis configuration .....	5 – 23
6.10	I/O configuration of the CAN bus .....	5 – 25
6.11	Address ranges of the CAN inputs .....	5 – 26
6.12	Address ranges of the CAN outputs .....	5 – 26
6.13	Diagnosis of the local I/O .....	5 – 27
6.13.1	Display CAN input/output signals (MODE 7.14/7.15/7.9) .....	5 – 27
6.14	Error messages .....	5 – 28
6.14.1	Start-up and initialisation phase .....	5 – 28
6.14.2	Runtime errors .....	5 – 29

<b>7</b>	<b>Drive parameter download to Servodyn – GC via the CAN bus</b>	
7.1	Input of the drive parameters .....	5 – 30
7.1.1	Input and optimisation directly at the drive booster .....	5 – 30
7.1.2	Input via rho3 machine parameter program .....	5 – 30
7.2	Transmission of the parameters .....	5 – 30
7.3	Allocation of drive parameters – rho3 parameters .....	5 – 31
7.4	Machine parameters rho3 .....	5 – 33
7.5	Calculation of drive parameters from rho3 parameters available .....	5 – 33
<b>8</b>	<b>“Flying” measurement by means of probe input</b>	
	<b>Special functions 43 and 44 .....</b>	<b>5 – 35</b>
8.1	Restrictions .....	5 – 36
<b>9</b>	<b>Bit coupler with I/O board in the RC cardrack</b>	
	<b>I/O configurations 15 and 16</b>	
9.1	Description of function .....	5 – 37
9.2	Setting machine parameters .....	5 – 37
9.3	Address range .....	5 – 38
9.4	Restrictions .....	5 – 38
<b>10</b>	<b>Deleting the write/read buffer</b>	
10.1	Introduction .....	5 – 39
10.2	Command syntax .....	5 – 39
10.3	Deleting the write buffer .....	5 – 39
10.4	Deleting the read buffer .....	5 – 40
10.5	Availability/restrictions .....	5 – 41
10.6	Examples .....	5 – 41
<b>11</b>	<b>Disabling individual axes</b>	
11.1	Function .....	5 – 43
11.2	Safety notes .....	5 – 43
<b>12</b>	<b>The special function 46 (BLOCK_SEARCH)</b>	
12.1	Declaration .....	5 – 44
12.2	Call-up in BAPS .....	5 – 44
<b>13</b>	<b>Asynchronous inputs</b>	
13.1	Declaration .....	5 – 45
13.2	Asynchronous inputs in the BAPS program .....	5 – 45
13.3	Application example of asynchronous inputs .....	5 – 46

<b>14</b>	<b>Change of function operation</b>	
	<b>Directly approaching points in the Teach in mode</b>	
14.1	Operation .....	5 – 49
14.2	Restrictions .....	5 – 50
14.3	Preventing points which have been taught in from being overwritten .....	5 – 51
<b>15</b>	<b>Expansion of the function "fast, smooth start-off" .....</b>	<b>4 – 52</b>

## Contents Part 3

### TO06

<b>1</b>	<b>Workpiece coordinate system</b>	
1.1	General .....	6 – 1
1.2	BAPS2 Syntax .....	6 – 2
1.2.1	Declaration of variables .....	6 – 3
1.2.2	Assignments in BAPS .....	6 – 3
1.2.3	Changing over the coordinate system .....	6 – 4
1.3	Machine parameter P313: WCSYS–ROB– ASSIGNMENT .....	6 – 4
1.4	Selection and effect of a workpiece coordinate system in the movement program .....	6 – 6
1.4.1	World coordinate points (WC points) .....	6 – 6
1.4.2	Joint coordinate points (JC points) .....	6 – 6
1.4.3	POS , @POS , @MPOS .....	6 – 7
1.4.4	LIMIT_MIN, LIMIT_MAX .....	6 – 7
1.4.5	Standard functions JC ( ) , WC ( ) .....	6 – 7
1.4.6	Special functions 1, 2: IO/PPO Logic .....	6 – 8
1.4.7	Special function 17: MIRROR .....	6 – 8
1.4.8	Translating the world coordinate system (P310) .....	6 – 8
1.4.9	Limit switch monitoring .....	6 – 8
1.4.10	Program end, abort, control start–up .....	6 – 8
1.4.11	Axis displays .....	6 – 9
1.4.12	Displaying the active WC_SYSTEM .....	6 – 9
1.5	Selection and effect in Manual mode .....	6 – 9
1.6	Safety instruction .....	6 – 10
1.7	Examples of special workpiece coordinates .....	6 – 11
1.7.1	Example 1: Pure height offset .....	6 – 11
1.7.2	Example 2: Height offset and rotation .....	6 – 12

<b>2</b>	<b>rho 3.0</b>	
2.1	Software PLC for the rho3.0 .....	6 – 13
2.2	Runtime display of the software PLC .....	6 – 13
2.3	Communication between rho3.0 – PLC .....	6 – 13
2.4	Using the high-speed inputs and outputs of the rho3.0	6 – 13
2.5	Double assignment of the PHG interface .....	6 – 14
2.6	New or expanded machine parameters .....	6 – 14



**TO07**

	Page
<b>1 PHG 2000 .....</b>	<b>7 – 1</b>
1.1 PHG 2000 modes .....	7 – 3
1.1.1 PHG 3 compatible mode .....	7 – 3
1.1.2 Transparent mode .....	7 – 3
1.1.2.1 Transparent mode with existing menu structure ....	7 – 3
1.1.2.2 Transparent mode with user-defined menu structure	7 – 4
1.2 Key assignment .....	7 – 4
1.2.1 Mask-specific key assignment .....	7 – 4
1.3 BDT editor .....	7 – 5
1.4 Configuration and test .....	7 – 6
<b>2 PHG 2000 System Variables .....</b>	<b>7 – 7</b>
<b>3 Controlling the PHG 2000 output</b>	
3.1 General information .....	7 – 15
3.1.1 Examples of cursor positioning for different character sets .....	7 – 15
3.2 BAPS2 declaration section .....	7 – 16
3.3 Calling subroutines in BAPS2 .....	7 – 17
3.4 Outputs to the PHG 2000 in subroutine technique ..	7 – 18
<b>4 Define / Teach In</b>	
4.1 Introduction .....	7 – 23
4.2 Display layout .....	7 – 23
4.2.1 "Define" layout .....	7 – 23
4.2.2 "Teach In" layout .....	7 – 25
4.3 Operation .....	7 – 26
4.3.1 Operation of "Define" .....	7 – 26
4.3.1.1 Operation of "Define" with three axes .....	7 – 27

4.3.2	Operation of "Teach In" .....	7 – 28
4.3.2.1	Operation of "Teach In" with three axes .....	7 – 28
4.3.2.2	Operation of "Teach In" with five axes .....	7 – 29
<b>5</b>	<b>Dateiliste (File list)</b>	
5.1	File list mask in the BDT editor .....	7 – 31
5.2	Creating the objects in the BDT editor .....	7 – 32
5.2.1	Heading "File list" on the PHG 2000 .....	7 – 32
5.2.2	Listbox for the file list .....	7 – 33
5.3	Displaying the memory allocation .....	7 – 34
5.4	PHG 2000 display .....	7 – 35
<b>6</b>	<b>Process Information</b>	
6.1	Mask in the BDT editor .....	7 – 37
6.2	Generating the objects in the BDT editor .....	7 – 38
6.2.1	Listbox for the process list .....	7 – 40
6.2.2	Global process information .....	7 – 41
6.2.3	Information on the selected process .....	7 – 41
6.3	PHG 2000 display .....	7 – 42
6.4	Stopping processes .....	7 – 43
<b>7</b>	<b>Variable assignment of PHG keys</b>	
7.1	Description .....	7 – 45
<b>8</b>	<b>Selecting a PKT file or point name</b>	
8.1	General .....	7 – 59
8.2	Declaration .....	7 – 59
8.3	Example .....	7 – 60
<b>9</b>	<b>Reversing an absolute measuring system</b>	
9.1	General .....	7 – 61
9.2	Function .....	7 – 61
9.3	Compatibility .....	7 – 61

**10 IAT Programming**

10.1	General .....	7 – 63
10.2	Requirements .....	7 – 63
10.2.1	Hardware .....	7 – 63
10.2.1.1	Switch setting on the IAT module .....	7 – 63
10.2.2	Software .....	7 – 64
10.2.2.1	Programming environment .....	7 – 64
10.2.2.2	Software for supporting PLC programming .....	7 – 64
10.3	Creating PLC programs .....	7 – 65
10.3.1	Recommended procedure .....	7 – 65
10.3.2	Special features .....	7 – 66
10.3.2.1	Data types .....	7 – 66
10.4	Executing the PLC program .....	7 – 66
10.4.1	Displays .....	7 – 66
10.5	Automatic start-up .....	7 – 67

**11 rho3 coupling via CAN bus**

11.1	Coupling of several rho3 controls .....	7 – 69
11.1.1	Linking the controls .....	7 – 69
11.1.2	Setting the identifiers .....	7 – 69
11.1.3	Structure .....	7 – 71

**12 Memory expansion with SRAM board**

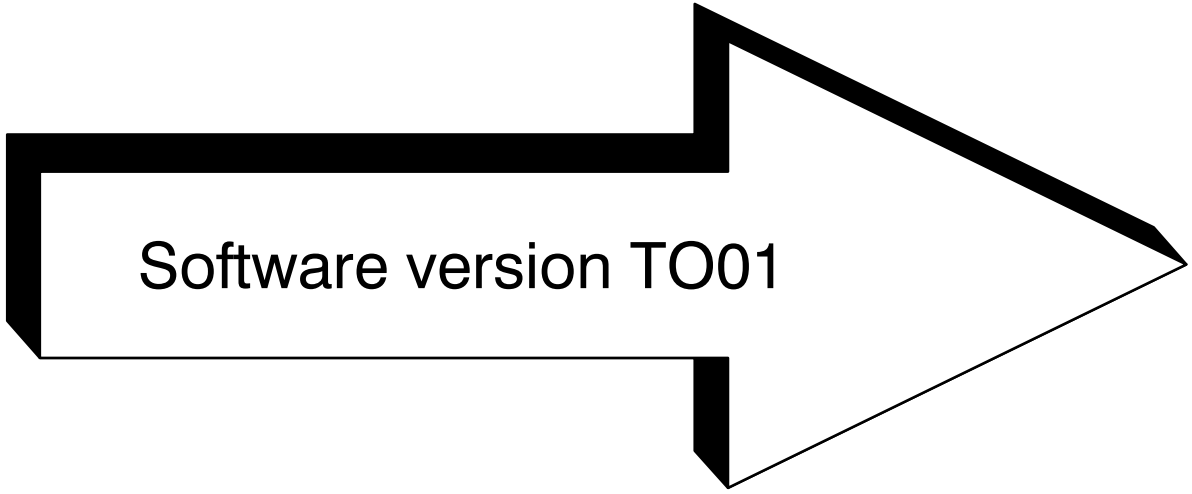
12.1	General .....	7 – 73
12.2	Formatting the SRAM board .....	7 – 73
12.3	Inserting the SRAM board into the rho3.0 .....	7 – 73
12.4	Removing the SRAM board .....	7 – 73
12.5	Restrictions .....	7 – 74
12.6	Data protection .....	7 – 74

<b>13</b>	<b>PLC Interface expansion</b>	
13.1	General .....	7 – 75
13.2	Expanded PLC interface for rho3.0 withdrawable module version .....	7 – 75
13.2.1	PLC programming .....	7 – 76
13.2.2	Setting the rho3 machine parameters .....	7 – 77
13.2.2.1	P20 I/O module configuration .....	7 – 77
13.2.2.2	P21 address range for PLC bit coupling .....	7 – 77
13.3	Expanded PLC interface for rho3.0 (bit coupling) ...	7 – 77
13.3.1	PLC programming .....	7 – 78
13.3.2	Setting the rho3 machine parameters .....	7 – 78
13.3.2.1	P20 I/O module configuration .....	7 – 79
13.3.2.2	P21 address range for PLC bit coupling .....	7 – 79
13.4	Expanded PLC interface for rho3.1 bit coupling in the extended I/O field .....	7 – 79
13.4.1	PLC programming .....	7 – 80
13.4.2	Setting the rho3 machine parameters .....	7 – 80
13.4.2.1	P20 I/O module configuration .....	7 – 80
13.4.2.2	P21 address range for PLC bit coupling .....	7 – 80
13.5	Overview of the programs for the Expanded PLC interface .....	7 – 81
13.5.1	Programs for withdrawable rho3.0 module .....	7 – 81
13.5.2	Programs for rho3.1 bit coupler .....	7 – 81
13.5.3	Programs for rho3.1 bit coupler in the extended I/O field .....	7 – 81
<b>14</b>	<b>Expansion of the BAPS2 function CONDITION .....</b>	<b>7 – 83</b>

**TO08**

	Page
<b>1</b>	<b>BAPS FUNCTION "BREAK"</b>
1.1	BAPS syntax ..... 8 – 1
1.2	Special features ..... 8 – 2
<b>2</b>	<b>Multi-function I/O's</b>
2.1	General ..... 8 – 3
2.1.1	General conditions ..... 8 – 3
2.2	Multi-function I/O's via SoftPIC for stand-alone rho3.0 .. 8 – 3
2.3	Multi-function I/O's via PLC with withdrawable rho3.0 version ..... 8 – 4
2.4	Machine parameters ..... 8 – 4
<b>3</b>	<b>Initializing SRAM boards</b>
3.1	General ..... 8 – 5
3.1.1	General conditions ..... 8 – 5
3.2	Formatting and initializing the SRAM board ..... 8 – 6
3.2.1	Procedure for initializing an SRAM board ..... 8 – 6
3.2.2	Error messages ..... 8 – 7
3.3	Data protection ..... 8 – 9
3.4	Replacing the SRAM board ..... 8 – 10
<b>4</b>	<b>Belt synchronization with endless belt</b>
4.1	General ..... 8 – 11
4.2	Function ..... 8 – 11
4.3	Restrictions ..... 8 – 11
4.4	Program examples ..... 8 – 12
4.4.1	Example 1 ..... 8 – 12
4.4.2	Example 2 ..... 8 – 13

<b>5</b>	<b>Remote use of the rho3.0</b>	
5.1	General .....	8 – 15
5.2	PLC program examples for remote use of the rho3.0	8 – 16
<b>6</b>	<b>Start-up behavior of multiple rho3.0's (in PLC) .....</b>	<b>8 – 17</b>
<b>A</b>	<b>Annex</b>	
A.1	Abbreviations .....	A-1
A.2	Safety instructions .....	A-2
A.2.1	Dansk .....	A-2
A.2.2	Deutsch .....	A-4
A.2.3	Ελληνικά .....	A-6
A.2.4	English .....	A-8
A.2.5	Español .....	A-10
A.2.6	Français .....	A-12
A.2.7	Italiano .....	A-14
A.2.8	Nederlands .....	A-16
A.2.9	Português .....	A-18
A.2.10	Suomi .....	A-20
A.2.11	Svenska .....	A-22



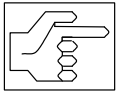
Software version T001

## Contents Part 1

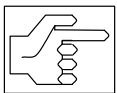
	Page
<b>1</b>	<b>Safety notes</b>
	regarding rho 3 operations ..... 1 – 1
<b>2</b>	<b>Changes to the PHG -</b>
	<b>Operation and Display</b> ..... 1 – 2
<b>3</b>	<b>Machine parameters</b> ..... 1 – 3
<b>4</b>	<b>Signals and PLC functions</b> ..... 1 – 4
<b>5</b>	<b>Testing BAPS2 programs</b>
	<b>with the PHG</b> ..... 1 – 5
<b>6</b>	<b>Other changes</b> ..... 1 – 6
<b>7</b>	<b>Machine status display</b> ..... 1 – 7
7.1	File "MSD.DAT" ..... 1 – 8
7.2	MSD interface assignment ..... 1 – 9
7.3	Restrictions and error messages ..... 1 – 9
<b>8</b>	<b>Coded text output</b> ..... 1 – 10
8.1	File "TEXT.DAT" ..... 1 – 11
8.2	Interface assignment, coded text output ..... 1 – 12
8.3	Limitations and error messages ..... 1 – 12



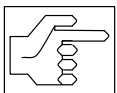
## 1 SAFETY NOTES REGARDING RHO 3 OPERATIONS



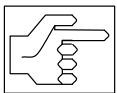
Pressing the jog key during continuous jog mode switches the jogging speed from slow to fast continuous after approx. 1.5 seconds. If other axes are then activated in jog mode, they move at the current fast speed.



In accordance with VDI 2853, the tool holding flange must not exceed a speed of 25 cm/s in Manual mode. Enter correct machine parameters (P111, P112, P113, P114) to ensure that this speed is not exceeded.



The PHG 3 must be removed from the system if it is not connected to the system controller. This is because the PHG's EMERGENCY STOP is not connected to the system's EMERGENCY STOP circuit.



Refer also to the safety notes in the following documents:

- rho 3      Description of machine parameters      1070 073 027
- rho 3      Signal description      1070 073 029
- rho 3      Interface conditions      1070 073 025
- rho 3      BAPS2 - Programming instructions      1070 073 033
- rho 3      PHG - Operation      1070 073 031
- ROPS3IQPRO Readme.TXT - file

## 2 Changes to the PHG - Operation and Display

- PHG output has been accelerated.
- In all of the functions, the listed files (START, EDIT, DEF, etc.) can be scrolled backwards.

CURSOR DOWN => forward

As before, CURSOR LEFT => exit mode;

As before, CURSOR UP => backward;

CURSOR RIGHT => to the beginning of the list.

It is also possible to scroll "in circles", i.e., the first file comes again after the last file, and vice-versa. Use CURSOR RIGHT to go to the beginning of the list.

The name of the most recently selected file appears the next time a function is called.

### **EXAMPLE :**

After DANA.QLL is compiled, for instance, DANA.PKT is offered for Define.

### **Directly entering file names**

Up to 8 characters, including "\_" can be used for entering file names. The files TEXT.DAT and MZA.DAT can now contain comments and space lines. Comments are distinguished by ";". Specifying the channel "PHG" is permissible in TEXT.DAT. The default channel (the only output channel) is still "PHG".

### **Compressing the user memory:**

After control run-up, during RESET, and after compilation, free areas in the user memory are compressed and files are arranged contiguously.

However, this happens only if no processes are active and all files are closed.

### **Auto-repeat for signal display by pressing keys**

CURSOR DOWN or CURSOR UP.

- Pressing ALT-SHIFT-P prints the PHG display on a connected printer.
- The functions MSD, coded text output, external program selection, particularly for long \*.DAT files, have been considerably accelerated.
- All CAN status errors are now shown under Diagnosis (previously, only the first error was shown).

### 3 Machine parameters

- Representation of the Timers and Counters (P13, P14) for machine parameter printout has been matched to the display on the PHG.
- SHIFT-UP/DOWN is now possible for P301, P304, P305, P506.
- The actual speed of an axis or belt can be output via a free analog output. Assignment is by means of machine parameters.  
P405 --> 7XX : XX is the number (sequential, via the servo boards) of the measuring system of the corresponding axis.
- For LINEAR interpolation, it is possible to monitor the axis speed for the max. permissible axis speed (V\_PTP\_MAX). Activation is with machine parameter P306.
- The values for P18 must be entered again; see rho 3 description of machine parameters, release 6/92.
- Using machine parameter P24, it is now possible to set the number of user outputs which are to be deleted during Control reset. The default option is Delete all user outputs. See also the description of machine parameters, release 6/92.
- Additional transformations for special robot kinematics are now available. Please refer to the respective transformation documentation for detailed information.

## 4 Signals and PLC functions

- Changed signal name :  
DIS\_PHGM\_RCI instead of SP\_DIS23\_RCI  
new signal:  
DISCOUPL\_RCI = 38; { O 4.6 | 3.7 }  
deleted signal DISSER\_1\_RCI .. DISSER\_4\_RCI, comments corrected.
- The function of the signal **Manual–Test,not** has been changed to match the signal description.
- Beginning with TO01I, whenever PIC or PLC timer errors occur the PIC is stopped, the READY contact is opened, and all RC inputs and outputs are deleted.
- If write protection is activated when the PIC is loaded, the corresponding procedure is aborted with an error message.
- The function that blocked the Referencing and Manual functions for set "TEST\_MN\_RCI" RC input has been removed.
- PIC250 programming is now performed in the Page mode which significantly accelerates PIC programming.
- The number of user outputs to be deleted during Control reset can be set with machine parameter P24; see above.
- Coded error output at the interface has been expanded to 16 bits. This is because the number of possible error codes the rho3 can issue exceeds 256.

The new bits 8..15 are addressed as follows:

Symbol.Name	Seq. no.	PIC address	CL300 address	Byte.Bit	Signal description
ERROR_8_RCO	160	I20.1	I18.0	13.1	Coded error output Bit-8
ERROR_9_RCO	161	I20.2	I18.1	13.2	Bit-9
ERROR10_RCO	162	I20.3	I18.2	13.3	Bit-10
ERROR11_RCO	163	I20.4	I18.3	13.4	Bit-11
ERROR12_RCO	164	I20.5	I18.4	13.5	Bit-12
ERROR13_RCO	165	I20.6	I18.5	13.6	Bit-13
ERROR14_RCO	166	I20.7	I18.6	13.7	Bit-14
ERROR15_RCO	167	I20.8	I18.7	13.8	Bit-15

Previously, "Outputs 1..8 from PG" were at these addresses; "Output 1..8 PG" are now found starting at Ser. no. 136..143; see also rho3 Signal descriptions, as of release 5/92.

- **New signal in the interface :**

RC start-up bit no. 107 I13.3

This signal is set when the rho3 has completed its run-up and an INIT program can be selected. The RC start-up signal comes too soon for selecting the INIT program and is therefore unsuitable.

## 5 TESTING BAPS2 PROGRAMS WITH THE PHG

- Outputting source lines in the test system has been accelerated.
- It is now possible to test **external** programs in the test system.
- In the test system, the sub-program name is now displayed when sub-program variables are displayed and set. The name is in the second PHG line preceding the variable name and is separated from this name by a blank. This makes it possible to distinguish sub-program variables from main program variables of the same name.
- When scrolling through the variable names, the subroutine variables of the current subroutine are offered directly (previously this was only possible if a subroutine variable was entered by name).

### **Note:**

If the subroutine variables in the subroutine have not yet been described, the values which are on the IRDATA stack at that moment are displayed. If type REAL variables or coordinate or axis values are displayed, this can lead to "**System error 23 FPU-Trap**".

The IRDATA stack requirements when external programs are called (without Or with parameters) have slightly increased. In case of programs which work near the limits of the IRD stack, this can lead to the error message IRD Stack Overflow. The size of the IRD stack must then be increased using machine parameter P16.

- In the test system it is possible to test programs without working with the respective kinematics. Under Mode 10, all the kinematics occurring in the program can be switched between **WITH** or **WITHOUT** movement. The WITH/WITHOUT movement function only applies to **all** of the available kinematics simultaneously.

## 6 OTHER CHANGES

- New special functions (27) for correcting world coordinate overspeeds for specific kinematics.
- An address is always provided together with system errors in order to better be able to locate the error.
- MOVE UNTIL commands only work in previous versions if the same kinematics are used in the MOVE UNTIL command as in the previous command.

As of version TO01I, it is possible to execute a kinematic other than the kinematic that is actually programmed in the command directly before the MOVE UNTIL command.

### CAUTION:

For offline programming this functional capability is available as of the ROPS3 software version **W1F**.

- For BAPS2 I/O files, the variable types BINARY can also be read and written.
- The BAPS2 vocabulary was increased by the following functions:

- **ORD CHARACTER\_VARIABLE**,  
This function provides the INTEGER value of the TYPE variable characters.

Example :

ZEICH\_NUM = ORD (ASC\_ZEICH) or ZEICH\_NUM = ORD( 'i' )

- **CHR (INT)**

provides a value of the type CHAR corresponding to the calculation INTEGER MOD 256

Example :

ASC\_ZEICH = ORD (34)

- **WRITE FELD\_VAR**

This means that in the WRITE instruction, field variables of the character type can be output, whereby the following limitation applies:

Field[1..<=80] Character; i.e., the field variable can contain a max. of 80 characters.

- Compiler instruction  
The following compiler instructions were introduced :

**;;ERROR = # ,**

The compiling operation is concluded at # number of errors.

**;;WARNING [+] | - ,**

**;;WARNING [-]** can be used to suppress the warnings issued by the compiler.

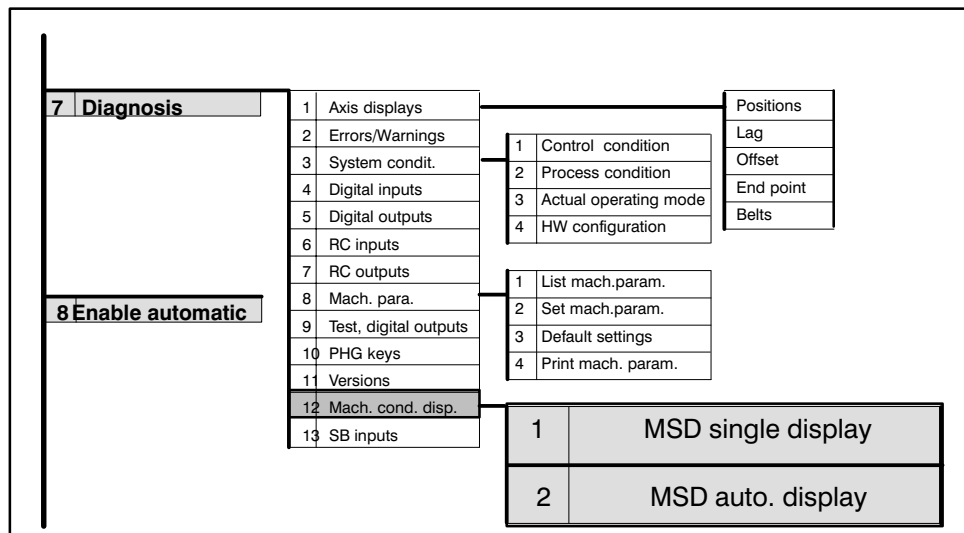
- In the ROPS3 version **W1F** , DOS errors are given in plain text.

## 7 Machine status display

The user can provide operating personnel with information on the system status by means of the MSD (machine status display). To keep work in the PIC program at a minimum and because generally, most statuses are also used as user inputs in BAPS programs, a specific text can be filed in "MSD.DAT" for each of the first 99 user inputs.



The RC input 60 (PIC address O7.4), "MSD-active" is used for notifying the operator of a new status. As long as the signal from the PIC is set to "1", "MSD" is shown in line 4 on the PHG.

All active statuses, i.e., when the respective RC input is "1", can be displayed one by one by means of the PHG operating mode DIAGNOSIS (mode 7), sub-mode MSD (mode 12).




**There are two ways to display states:**

1. Manually by scrolling

by pressing **Shift** +  or **Shift** + 

(scrolling backwards is not possible).

2. Automatically and cyclically in seconds until the sequence is aborted with

**Shift** + 

In both cases, two statuses are simultaneously displayed in lines 2 and 3.

**Example :**

01 = 'first status'  
08 = 'this is a long text'  
99 = 'last text'

**PHG machine status display :**

```
Machine statuses
01 first status
08 this is a long
99 last text
```

Texts which go beyond the end of a PHG line are cut off.

**7.1 File "MSD.DAT"**

The MSD.DAT file is similar in form to the ASCII file EXPROG.DAT (external program selection).

The numbers of the statuses and their respective texts are BCD coded. They must always be specified as a two-place number. Spaces between numbers and '=', or between '=' and text are permissible, and are ignored during read operations. MSD texts must be delimited by inverted commas ( ' ). Control characters in texts are prohibited. Comments are identified by a semi-colon ( ; ). Texts can be up to 80 characters long.

Structure of the file <b>MSD.DAT</b>			
<b>Code value = 'assigned output text' ; Comment</b>			
01	=	'first text'	
02	=	'texttexttext'	; Comment
20	=	'text'	; line for additional ; information ; Comment to end of line ; if desired
32	=	'text'	
99	=	'last text'	

For German user interfaces the file name is "MZA.DAT" and for English interfaces the file name is "MSD.DAT" (machine status display).



## 7.2 MSD interface assignment

The message "MSD active" appears in the PHG display (line 4) for the RC output 60 (PIC address O7.4).

In addition to their function as user inputs (1..99), the RC inputs 464 - 562 (PIC addresses O58.0 - O70.2) can be used for the machine status display.

Selecting the diagnosis function "Machine condition display" (mode 12), sub-mode 1 or 2, sets RC output 135 (PIC address I14.3), and exiting mode 12 resets it. In this way, the user is able to recognize in the PIC program whether the MSD display was activated by the operator, and can control the signal "MSD active" (O7.4) accordingly.

<b>RC outputs:</b>				
BAPS name	Bit no.	PIC addr.	RC input.	Signal description
MSD_ACTI_RCO	115	I14.3	7.4	MSD display was activated

<b>RC inputs:</b>				
BAPS name	Bit no.	PIC addr.	RC input.	Signal description
MSD_DISP_RCI	60	O7.4	6.5	MSD active
USER_1_RCI	464	O58.0	57.1	User input 1 Machine status 1
.	.	.	"	" "
.	.	.	"	" "
USER_99_RCI	562	O70.2	69.3	User input 99 Machine state 99

## 7.3 Restrictions and error messages

- Control characters in texts are prohibited.
- If the MSD file cannot be found, the operator is informed with a message.

## 8 Coded text output

Coded text outputs are tools which allow the machine manufacturer to show the operator important information or inform him that errors or conditions have occurred. The PHG is used to show this information (lines 2 and 3).

Up to 256 different texts can be displayed. Selection the text to be displayed is by means of the internal RC interface (RC inputs 104 - 111, PIC address O13.0 - O13.7).

Transfer to the RC is by means of 8 hexadecimal coded bits and one strobe signal (RC input 85, PIC address O10.5). After the text number has been accepted, the selected text is output on the "PHG" output channel.

The acknowledgement signal "Text output ok" is set at the internal RC interface (RC output 108, PIC address I13.4) when the text is output.

If the output channel is occupied or not ready, the error strobe "Text output error" (RC output 109, PIC address I13.5) is set.

RC messages on lines 2 and 3 can be blocked by setting the RC output 70 (PIC address O8.6) "Output of system messages on the PHG blocked".

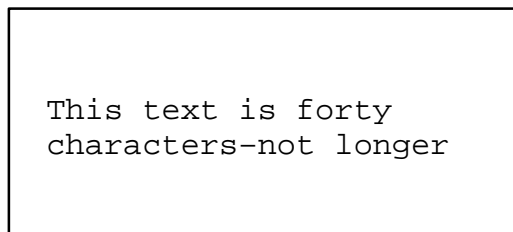
All texts are filed in "TEXT.DAT" and can even be created or changed in offline mode (ROPS).

Coded text outputs overwrite lines 2 and 3 on the PHG. To prevent this text from immediately being written over (for example, by a dynamic RC display), the RC output on lines 2 and 3 can be blocked by means of the RC input "Output of system messages on the PHG disabled" (bit no. 35, PIC address O4.3).

Example :

O1=PHG 'This text is forty characters –not longer'

### **Coded text output on the PHG :**



```
This text is forty
characters-not longer
```

Texts for output on the PHG can be up to 40 characters long. Texts longer than 40 characters are cut off

**8.1 File "TEXT.DAT"**

The TEXT.DAT file is similar in form to the ASCII file MSD.DAT.

The text code numbers are hexadecimal. They must always be specified as a two-place number. The characters 'A' - 'F' must be entered as upper case letters.

Spaces between numbers and '=', or between '=' and channel, as well as between channel and text are permissible, and are ignored during read operations. The channel designation is concluded with a comma ','. The text to be output must be between '''. Control characters in texts are prohibited. Comments are identified by a semi-colon (;). Texts can be up to 80 characters long. If no channel is specified, the PHG is the default.

Example :

Structure of the file <b>TEXT.DAT</b>			
<b>Code value = [CHANNEL,]'assigned output text' ; Comment</b>			
00	=	PHG,	'first text'
01	=	PHG,	'text text'
02	=	'texttexttext'	
.			
3E	=	'text' ;comment	
.			
FF	=	'last text'	

The name of the file for the German interface is "TEXTE.DAT", and "TEXT.DAT" for the English interface.

## 8.2 Interface assignment, coded text output

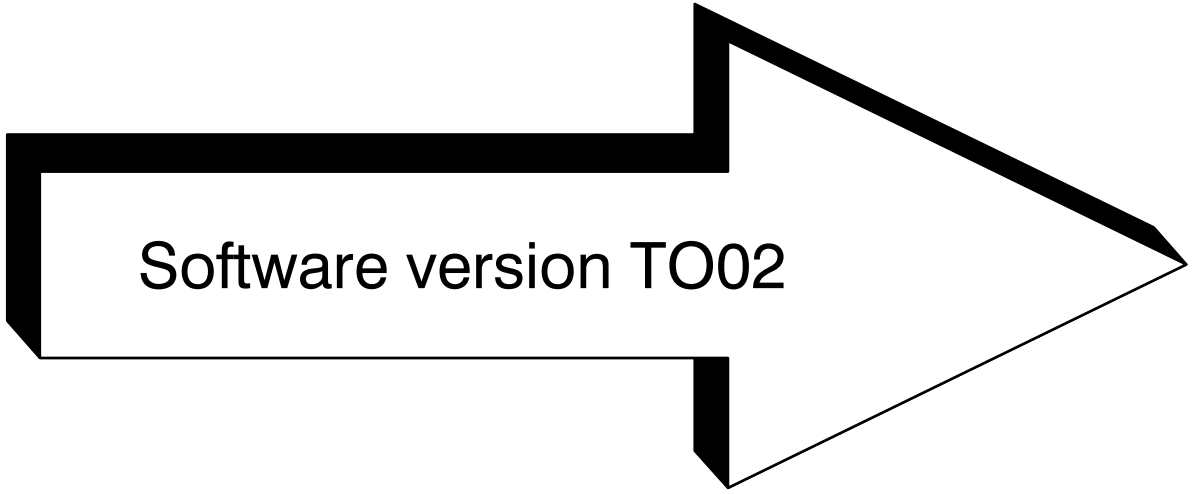
If RC input 85 (PIC address O10.5) is set, the HEX number at RC inputs 104 - 111 (PIC addresses O13.0 - O13.7) is accepted by the RC and text output is activated. At the conclusion of the output, RC output 108 (PIC address I13.4) is set; in case of an error, RC output 109 (PIC address I13.5) is set.

RC outputs :				
BAPS name	Bit no.	PIC – addr	RC – output	Signal description
TXOUT_OK_RCO	108	I13.4	6.5	Coded text output ok
TXOUT_ER_RCO	109	I13.5	6.6	Error in coded text output

RC inputs:				
BAPS name	Bit no.	PIC – addr	RC – Input	Signal description
STB_TXTO_RCO	85	O10.5	9.6	Strobe, coded text output
DIS_PHGM_RCI	35	O 4.3	3.4	Disable PHG display Lines 2+3 for RC outputs
TXTOUT_0_RCI	104	O 13.0	12.1	Coded text number 2 to the power of 0, value 1
	..	O ...	..	
TXTOUT_7_RCI	111	O 13.7	12.8	Coded text number 2 to the power of 7, value 128

## 8.3 Limitations and error messages

- Control characters in texts are prohibited.
- If the TEXT.DAT file cannot be found or if no text can be output because the output channel is occupied, the text output error strobe is set.
- Currently, only the **PHG** can be used as the output channel.



Software version T002

## Contents TO02F

<b>1</b>	<b>General notes on using TO02F software</b> .....	<b>2 – 1</b>
<b>2</b>	<b>High-speed measuring</b>	
2.1	Programming .....	2 – 2
2.1.1	Program example .....	2 – 4
2.2	Hardware requirements .....	2 – 5
2.3	Machine parameters .....	2 – 6
2.4	Restrictions .....	2 – 6
2.5	Achievable accuracies .....	2 – 7
<b>3</b>	<b>Interruptible axes</b>	
3.1	RC input signals .....	2 – 8
3.2	Control-internal effect of the DRIVE-ON signals .....	2 – 8
3.3	Applications .....	2 – 9
3.3.1	Manual mode .....	2 – 9
3.3.2	Automatic mode .....	2 – 9
3.3.3	Safety notes .....	2 – 9
3.3.4	Example 1 for using DRIVE_ON .....	2 – 9
3.3.5	Example 2 for using DRIVE-ON .....	2 – 11
3.3.6	PLC program for examples 1 and 2 (segment) .....	2 – 12
3.4	Monitoring functions .....	2 – 12
3.5	Example 3 (PLC program) .....	2 – 13
<b>4</b>	<b>Fast, smooth start-off</b>	
4.1	Programming .....	2 – 14
4.2	Example: .....	2 – 16
<b>5</b>	<b>Setting the belt counter (special function 28)</b>	
5.1	Example: .....	2 – 17
5.2	Belt simulation .....	2 – 18

<b>6</b>	<b>Spatial passing, optimum processing</b> .....	<b>2 – 19</b>
6.1	Spatial passing for program slopes .....	2 – 20
6.1.1	PTP passing for program slopes .....	2 – 20
6.1.2	Path passing for program slope .....	2 – 22
6.1.3	Passing when changing the type of interpolation .....	2 – 22
6.1.4	Special cases .....	2 – 22
6.2	Passing for block slopes .....	2 – 24
6.2.1	PTP passing for block slopes .....	2 – 24
6.2.2	Path passing for block slopes .....	2 – 25
6.2.3	Passing when changing the type of interpolation .....	2 – 25
6.3	BAPS2 - Language elements .....	2 – 26
6.3.1	Programming .....	2 – 26
6.4	Machine parameters .....	2 – 27
6.5	Restrictions .....	2 – 27
<b>7.</b>	<b>Protocol 3964/R</b>	
7.1.	Implementing the protocol driver .....	2 – 28
7.2.	Interfacing with the BAPS program .....	2 – 30
7.3.	WRITE device .....	2 – 31
7.4.	READ device .....	2 – 31
7.5.	Special cases and restrictions .....	2 – 32
<b>8.</b>	<b>Multiple use of the serial interfaces</b>	
8.1.	Automatic reactions to a conflict .....	2 – 33
8.2.	Controlled avoidance of conflicts .....	2 – 34
8.3.	Replugging or reswitching the interface .....	2 – 34
8.4.	Status display .....	2 – 35
8.5.	Example .....	2 – 35
8.6.	Notes on implementation .....	2 – 36
<b>9</b>	<b>CONDITION inquiry of interfaces</b>	
9.1	Compiler instruction SER_IO_STOP .....	2 – 37
9.2	CONDITION function and interface .....	2 – 38
9.3	CONDITION function and file .....	2 – 39
<b>10</b>	<b>Standard procedures</b>	
10.1	Standard procedure INT_ASC .....	2 – 40
10.2	Standard procedure ASC_INT .....	2 – 41
<b>11</b>	<b>PUBLIC variable</b> .....	<b>2 – 42</b>

<b>12</b>	<b>Kinematic-related automatic - manual mode</b>	
12.1	New Signals and their meanings .....	2 – 43
12.1.1	Signal meanings .....	2 – 44
12.2	Automatic mode .....	2 – 44
12.3	Referencing and Set-up .....	2 – 44
12.4	Teach in .....	2 – 45
12.5	Testing .....	2 – 45
12.6	Changing from global AUTOMATIC/MANUAL mode to AUTOMATIC/MANUAL MODE per kinematics .....	2 – 45
12.7	Notes on proper use of options .....	2 – 45
12.8	Diagnosis on the PHG .....	2 – 45
<b>13</b>	<b>External program/process deselection</b>	
13.1	RC input signals .....	2 – 46
13.2	RC output signals .....	2 – 47
13.3	Restrictions .....	2 – 47
13.4	EXPROG.DAT structure .....	2 – 47
<b>14</b>	<b>Setting potentiometer values</b>	
14.1	Description of function .....	2 – 48
14.1.1	Option byte=0 .....	2 – 49
14.1.2	Option byte<>0 .....	2 – 49
<b>15</b>	<b>INFO function on the PHG</b>	
15.1	Availability of the INFO function .....	2 – 51
15.2	Restrictions .....	2 – 52
15.3	Operation .....	2 – 52
15.4	Possible error states, Info texts .....	2 – 53
15.5	Output on the PHG .....	2 – 56
<b>16</b>	<b>Coded error output</b>	
16.1	Structuring the errors in groups .....	2 – 57
16.2	Address assignment of the error code signals .....	2 – 59
16.3	Coded error output of runtime errors .....	2 – 59
16.3.1	Principle of the output of error messages .....	2 – 59
16.3.2	Output of several error codes .....	2 – 60
16.3.3	Parity .....	2 – 60
16.3.4	Response for successive errors .....	2 – 60
16.4	Coded error output of system errors .....	2 – 61
<b>17.</b>	<b>ROPS3/IQPRO Version W1J</b>	
17.1.	Compatibility between ROPS 3 – rho 3 .....	2 – 62



## 1 General notes on using TO02F software

The following functions and options are available for the rho 3 control configuration with the TO02F software version.

### Caution:

**For reasons of safety, handling the EMERGENCY STOP signal has been changed in the TO02F software and is different from the previous software version.**

**Starting with this software version, an EMERGENCY STOP signal must be acknowledged with "Control reset" or a PHG operation in MODE 11.1 (reset).**

All other signals are downward compatible which means that the new TO02F software version can be installed in existing systems if new functions are required.

Rough interpolation operations and the position controller require more computing power for the "spatial passing movement" option. For this reason, when this option is used the machine parameter for rough interpolator cycle (P5), and when using a SERVO i, the increment factor for the position controller cycle must be reset.

In addition to the description of the options, please also refer to the following documents:

- rho 3 **PHG Operation**
- rho 3 **BAPS 2.0 Programming instructions**
- rho 3 **BAPS2 - Short description**
- rho 3 **Interface conditions**
- rho 3 **Machine parameters**
- rho 3 **Signal description**
- rho 3 **ROPS3/IQpro short description**

## 2 High-speed measuring

with **probe input** and "**high-speed inputs**" on the servo boards.

In order to be able to execute measuring operations on workpieces, pallets or tools at high speed, input signals are required to which the control can react with the shortest delay times possible. The **rho 3** option described in the following can meet this requirement by means of the high-speed inputs on the servo boards.

### 2.1 Programming

A movement can be aborted with a binary input if the BAPS program contains the following command:

**MOVE....**

**UNTIL** *Variable Rel\_Operator Expression* **ERROR** *Instruction*

**TO ....**

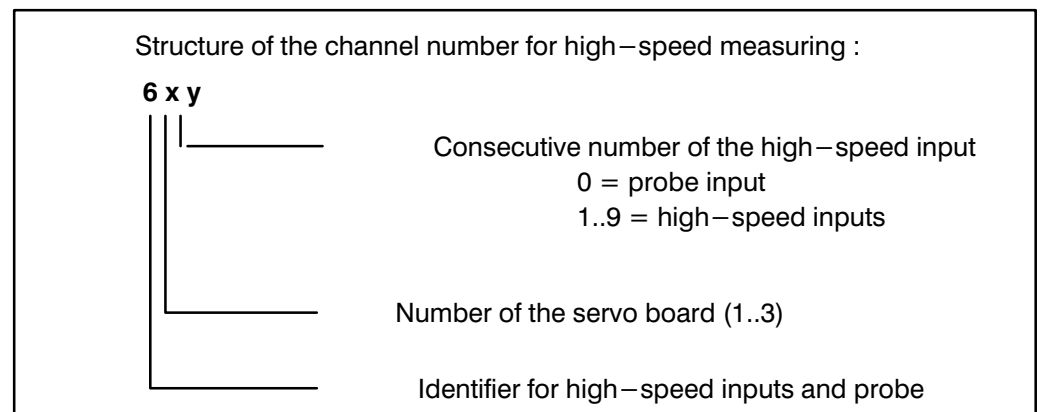
In this case the parameters have the following meaning :

*Variable* = **channel number** of a binary input

*Rel\_Operator* = { = || <> }

*Expression* = { 0 || 1 }

The inputs on the servo boards contain their own channel numbers.



The "**Move until...**" command is executed as normal binary inputs. The movement is interrupted when the status of the selected input changes to the programmed value (0 or 1).

Using a high – speed servo board input offers the advantage of being able to record the measuring position while still in the **position controller cycle**. Supplemental runtimes due to copying interface signals (PIC runtimes) no longer occur.

Using the probe input means that the measuring position is stored with practically no time delay.

In both cases, the next **interpolation cycle** (P2 cycle) is aborted. The measuring position can be determined by reading **@MPOS**.

**@MPOS** is a standard variable which was specifically defined for the application "high – speed measuring".

For high – speed inputs, the abort condition may already be pending before the **MOVE UNTIL** command is active. In this case, no movement is initiated. For probe inputs, a signal change must take place while the corresponding block is active. If no change takes place, movement is executed to the programmed end point.

The position of the most previously executed measuring procedure remains stored in the variable **@MPOS** until the conclusion of the next measuring procedure. If no measuring procedure is executed in the active program, **@MPOS** is not occupied, meaning that when **@MPOS** is read, the runtime error " **point not defined** " is issued.

To avoid the unintentional acceptance of positions from previous measurements, or to avoid aborting the program due to an unoccupied **@MPOS**, it is necessary to specify an error condition in the " **MOVE UNTIL** " block (see example).

Only those positions are accepted which belong to the programmed kinematics and which are on the servo board which has been selected by means of the channel number. If an unoccupied **@MPOS** component is accessed, the program is aborted with the runtime error:

" **point not defined** "

The measuring position is only available in joint coordinates. However, it can be converted to world coordinates with the **BAPS2** standard Function **WC** .

### 2.1.1 Program example

```
;; CONTROL = RHO3
;; KINEMATICS : ( 1 = ROBI1 )
;; ROBI1.JC_NAME = A1, A2
;; ROBI1.WC_NAME = K1, K2

PROGRAMM MTA5T1

WC_POINT: @MESSWERT1, @MESSWERT2

POINT: MESSWERT1, MESSWERT2, ANF_POS, END_POS2, END_POS1

INPUT : 610 = PROBE, ;PROBE INPUT
;ON SERVO BOARD 1
611 = HS_EIN1 ;1. HIGH-SPEED-INPUT
;ON SERVO BOARD 1

BEGIN

;;KINEMATICS = ROBI1

;Measured value via probe

MOVE LINEAR TO ANF_POS

MOVE LINEAR UNTIL PROBE = 1
ERROR JUMP MESS_ERR ;JUMP MESS_ERR
TO END_POS1 ;Prevents reading
;of @MPOS

@MESSWERT1 = @MPOS ;Read measured value

MESSWERT1 = WC (@MPOS) ;Calculation in
;world coordinates

;Measured value via fast input

MOVE LINEAR TO ANF_POS

MOVE LINEAR UNTIL HS_EIN1 = 1
ERROR JUMP MESS_ERR
TO END_POS2

@MESSWERT2 = @MPOS

MESSWERT2 = WC (@MPOS)

JUMP ABSCHLUSS

MESS_ERR: WRITE 'Probe did not respond'

ABSCHLUSS: HALT

PROGRAM_END
```

## 2.2 Hardware requirements

The servo boards used for the rho 3 control have different numbers of probes and high – speed inputs.

**SERVO 3/5/8 wide:**

8 high – speed inputs

1 probe

**Modular servo board (SERVO ABS 5/8, SERVO INC6/POTI):**

8 high – speed inputs

**no** probe

**SERVO 3/5 narrow :**

no high – speed inputs

1 probe

**SERVO 3iS/4iS6iS:**

9 high – speed inputs

1 probe

**SERVO ABS 3i, modular SERVO i:**

9 high – speed inputs

1 probe

**The high – speed inputs are only available if the corresponding module is inserted on the servo board (connector X31).**

The high – speed inputs can be used for all measuring system types.

The probe input is only effective for incremental encoders. For mixed chip operations with non-incremental measuring systems, the position is accepted as the measuring position in the next **position controller cycle** (as with high – speed inputs) when the probe input responds. This requires that at least 1 axis of the programmed kinematics is equipped with an incremental encoder and that this axis participates in the movement.

## 2.3 Machine parameters

### P11 SB (SERVO Board) INPUTS

SB1 No.of Probe Inp.:	0/1	(Servo Board 1) (0 = no probe) (1 = probe available)
SB1 'No.of HS–Inp.:	0..9	(no. of high–speed inputs 0..9)
SB2 No.of Probe Inp.:	0/1	(Servo board 2)
SB2 'No.of HS–Inp.:		
SB3 No.of Probe Inp.:	0/1	(Servo board 3)
SB3 'No.of HS–Inp.:	0..9	

The desired number of high–speed inputs is required in order to perform a comparison in the run-up phase of the actual number of available inputs (depending on the servo board type).

If the number of high–speed inputs specified by machine parameters is greater than the actual number of probes or high–speed inputs, control run-up is aborted with the error message:

**"incorr. highspeed–io"**

If an input is programmed in the user program which was not applied by means of a machine parameter, the following error message appears during runtime:

**"inadmissable input".**

## 2.4 Restrictions

- The "high–speed inputs" and probe inputs work only in conjunction with servo boards. This means that only those axis position values are accepted which are on the same servo board as the programmed input.
- If an access operation is attempted on an unavailable axis position in the @MPOS, the programmed is aborted with the runtime error:

**"point not defined"**

## 2.5 Achievable accuracies

### - Probe input

Response time: 10 $\mu$ s

Accuracy [mm]:

$v$  [mm/s] / 100000 [1/s] + mech. travel of the probe

### - high – speed inputs

Response time: 0..position controller cycle (LRTAKT)

Accuracy [mm]:

$(v$  [mm/s] / 1000) \* LRTAKT [ms] + mech. travel of the probe

$v$  = speed in direction of the probe

LRTAKT = position controller cycle

= interpolation / scaler (machine parameter P5)

### 3 Interruptible axes

#### Drive-on logic

In some applications (for example, in joining operations), it is necessary to remove one or more axes from the position control circuit in order to move these axes by means of, for instance, hydraulic/pneumatic cylinders. The rho 3 is capable of deactivating individual axes by means of **axis-specific** DRIVE ON signals.

#### 3.1 RC input signals

- Drive-on, all axes (PIC250- output no. 66)  
Group signal for all axes
- Drive-on 1st-20th axis (PIC250- output no. 192..211)  
Axis-specific Drive-on signals

The **axis-specific** Drive-on signals only function if the group signal is **not** set.

#### 3.2 Control-internal effect of the DRIVE-ON signals

**DRIVE-ON = 1** (axis activated):

Axis is in the position control circuit. Movement by means of an external setpoint value is not possible. Servo error monitoring is active.

**DRIVE-ON = 0** (axis deactivated):

Axis not in control loop. Axis positions are internally slaved. Servo errors are not triggered if an axis is moved by means of an external setpoint. When the axis is re-activated, the set position is set to the actual position and movement is resumed starting at the current position value.



### 3.3 Applications

Activating or deactivating one or more axes by means of DRIVE-ON signals is possible in manual mode as well as in the automatic mode.

#### 3.3.1 Manual mode

When mode 2 (manual mode) is selected and jogging operations are active for individual axes, the user can deactivate specific axes which are not involved in the movement, thus allowing these disabled axes to be externally manipulated. During external manipulation, the current actual position of the axes is slaved and displayed on the PHG.

Axes which have not been disabled can be moved by means of the PHG in joint coordinates and world coordinates. If movement of a disabled axis is triggered, the operation is aborted with the error message "**DRIVE ON not avail.**".

#### 3.3.2 Automatic mode

It is also possible to disable individual axes in automatic mode. It must be ensured, however, that the DRIVE-ON signals are activated and deactivated in synchronization with the running BAPS program. This requires that the BAPS and PIC program are matched to each other accordingly. A few application examples are provided in the following:

#### 3.3.3 Safety notes

If axes are moved when the DRIVE-ON signal is off, synchronization to the actual position must be performed before the next programmed axis movement (when DRIVE ON has been activated) as otherwise jump operations to the setpoint or (servo) errors can result.

Accepting the actual position can be achieved by means of a dummy MOVE UNTIL or MOVE\_REL UNTIL instruction; for example, MOVE\_REL UNTIL SIG = 1 TO DELTA. Here, note that the condition (SIG = 1) must be fulfilled before reaching the position programmed in DELTA.

**Note:**

**The instruction for accepting the current actual position, MOVE TO @POS, cannot be used here.**

#### 3.3.4 Example 1 for using DRIVE\_ON

4 axis Scara robot (for example, BOSCH SR 60). The 3rd axis is disabled and later enabled by means of the BAPS program.

The disabled axis is moved by means of an external setpoint. Simultaneously, the remaining axes are moved by means of the program in world coordinates.

```

;; CONTROL = RHO3
;; KINEMATICS : ( 1 = SR60 )
;; SR60.JC_NAMES = A_1, A_2, A_3, A_4
;; SR60.WC_NAMES = A_X, A_Y, A_Z, A_A

PROGRAM FSW_A3

OUTPUT:      1 = ANTRE_01,           ;Appl. outputs (1..4) are copied in the PIC program
              2 = ANTRE_02,           ;to input signals DRIVE ON axes 1..4.
              3 = ANTRE_03,           ;(ITEM NO.: 192..195)
              4 = ANTRE_04           ;if user output 17 (AUTO_AE) is set.
                               ; If AUTO_AE is not
                               ;set, the DRIVE-ON signals are set
                               ;(via switch) externally.
                               ;(see respective PIC program)
              17 = AUTO_AE

INPUT:       25 = ANTR1_OK,           ;Check-back signal of the DRIVE-
              26 = ANTR2_OK,           ;ON signal
              27 = ANTR3_OK,
              28 = ANTR4_OK

BEGIN

;;KINEMATICS = SR60
  ANTRE_01 = 1           ;Switch over to automatic
  ANTRE_02 = 1           ;DRIVE-ON control (via
  ANTRE_03 = 1           ;user outputs)
  ANTRE_04 = 1
  AUTO_AE = 1

LOOP:                                     ;Wait until all drives
  WAIT 0.5                               ;are switched on
  IF NOT (ANTR1_OK AND ANTR2_OK AND ANTR3_OK AND ANTR4_OK)
  THEN JUMP LOOP

MOVE LINEAR TO ANF_POS
  ANTRE_03 = 0           ;Disable 3rd axis
  WAIT 5
  MOVE LINEAR TO END_POS ;Movement without participation of 3rd axis
                               ;3rd axis can be simultaneously moved with
                               ;external setpoint.
  ANTRE_03 = 1           ;Re-enable 3rd axis

LOOP2:                                     ;Wait until 3rd axis
  WAIT 0.5                               ;is enabled.
  IF NOT ANTR3_OK
  THEN JUMP LOOP2

MOVE LINEAR UNTIL ANTR3_OK = 1           ;Dummy MOVE- UNTIL for forced acceptance of position
  TO END_POS                             ;as 3rd axis may have been moved by means of external
                               ;set point.

  ;Between the last traverse block and within the MOVE UNTIL block there must be no
  ;change of coordinate system.
  ;Unless the option flag "POS acceptance after change of coordinate system" (P306) is set.

MOVE TO ANF_POS
AUTO_AE = 0
HALT
PROGRAMM_END

```

### 3.3.5 Example 2 for using DRIVE-ON

DRIVE-ON signals are only influenced externally (via switches).  
Axes are only started if the respective DRIVE-ON signals are set.

```
;;CONTROL = RHO 3
;;KINEMATICS : ( 1 = SR60 )
;;SR60.JC_NAMES = A_1, A_2, A_3, A_4
;;SR.60.WC_NAMES = A_X, A_Y, A_Z, A_A

PROGRAM A1234_FB

    OUTPUT:      17 = AUTO_EA

    INPUT:       25 = ANTR1_OK,
                26 = ANTR2_OK,
                27 = ANTR3_OK,
                28 = ANTR4_OK

BEGIN

AUTO_EA = 0
LOOP:                                     ;Wait until all drives
    WAIT 0.5                               ;are enabled
    IF NOT (ANTR1_OK AND ANTR2_OK AND ANTR3_OK AND ANTR4_OK)
    THEN JUMP LOOP

MOVE_REL                                     ;Accept actual position
    UNTIL ANTR1_OK = 1 TO @(10,10,10,10)

    MOVE TO @(-50,-50,-50,-50) ;Approach start positon

    ;Axes move if respective DRIVE-ON signals are set.

WAIT UNTIL ANTR1_OK = 1
    MOVE UNTIL ANTR1_OK = 1 TO @(-50,-50,-50,-50)
    MOVE TO @(50,-50,-50,-50)

WAIT UNTIL ANTR2_OK = 1
    MOVE UNTIL ANTR2_OK = 1 TO @(50,-50,-50,-50)
    MOVE TO @(50,50,-50,-50)

WAIT UNTIL ANTR3_OK = 1
    MOVE UNTIL ANTR3_OK = 1 TO @(50,50,-50,-50)
    MOVE TO @(50,50,50,-50)

WAIT UNTIL ANTR4_OK = 1
    MOVE UNTIL ANTR4_OK = 1 TO @(50,50,50,-50)
    MOVE TO @(50,50,50,50)

HALT

PROGRAMM_END
```

### 3.3.6 PLC program for examples 1 and 2 (segment)

```

;----- DRIVE ON PER AXIS -----
IF NOT AAUS_17_RCA          ;Query user output 17
  THEN JUMP HAND_AE

; DRIVE-ON signals from BAPS program influenced
; by user outputs.

DRIVE_1_RCI = USER_1_RCO   DRIVE_2_RCI = USER_2_RCO
DRIVE_3_RCI = USER_3_RCO   DRIVE_4_RCI = USER_4_RCO

JUMP AE_ENDE

HAND_AE:
;DRIVE-ON logic via switch

DRIVE_1_RCI = USER_9_DI     DRIVE_2_RCI = USER_10_DI
DRIVE_3_RCI = USER_11_DI    DRIVE_4_RCI = USER_12_DI

AE_ENDE:
; Copy DRIVE-ON signals to user inputs
; for querying in BAPS program

USER_25_RCI = DRIVE_1_RCI   USER_26_RCI = DRIVE_2_RCI
USER_27_RCI = DRIVE_3_RCI   USER_28_RCI = DRIVE_4_RCI

```

### 3.4 Monitoring functions

Whenever an axis is to be moved for which no DRIVE-ON signal is set, the running program is aborted with the runtime error

**"DRIVE ON not avail. "**

The axis movement is checked at the joint coordinate level (after coordinate transformation). This makes movement possible in world coordinates, despite disabled axes, as long as none of the disabled axes participate in the movement.

#### Example: SR60

3rd axis is disabled by means of DRIVE-ON.

Axes 1 and 2 can nevertheless be moved in world coordinates (see example 1).

#### **Caution !**

If DRIVE-ON is disabled while a program is active, a MOVE UNTIL block to the POS acceptance must first be executed after reactivating DRIVE-ON. If this condition is not fulfilled, the program will be aborted at the next traverse block with the runtime error

**"DRIVE ON not avail."**

DRIVE-ON is only re-enabled if all axes of the kinematics in question are standing (POS acceptance). If this is not the case, the program is also aborted with the runtime error **"DRIVE ON not allowed"** .

To prevent runtime errors due to prohibited enabling and disabling operations of individual axes, the DRIVE-ON signal change in the PIC program should be blocked accordingly (see example 3).

### 3.5 Example 3 (PLC program)

; Block DRIVE-ON signal change for running axes

.  
.  
.

```
IF INPOS_1_RCO AND INPOS_2_RCO AND  
    INPOS_3_RCO AND INPOS_4_RCO  
    THEN JUMP INPOS_AA
```

; One or more axes move ==> DRIVE-ON  
; must not be enabled. DRIVE-ON must only  
; be disabled if the corresponding axis  
; does not move.

```
IF INPOS_1_RCO THEN  
    IF NOT USER_9_DI THEN DRIVE_1_RCI = 0  
    IF INPOS_2_RCO THEN  
        IF NOT USER_10_DI THEN DRIVE_2_RCI = 0  
    IF INPOS_3_RCO THEN  
        IF NOT USER_11_DI THEN DRIVE_3_RCI = 0  
    IF INPOS_4_RCO THEN  
        IF NOT USER_12_DI THEN DRIVE_4_RCI = 0
```

JUMP AE\_ENDE

INPOS\_AA:

; All axes are IN POSITION ==> DRIVE-ON signals  
; can be set corresponding to switch position  
;

```
DRIVE_1_RCI = USER_9_DI      DRIVE_2_RCI = USER_10_DI  
DRIVE_3_RCI = USER_11_DI     DRIVE_4_RCI = USER_12_DI
```

AE\_ENDE:

.  
.

## 4 Fast, smooth start-off

Fast and smooth start-off if a given condition is fulfilled with the aid of a modified **WAIT UNTIL** command.

For some applications it is necessary to initiate a traverse movement as quickly as possible and with a delay as constant as possible if a given condition is fulfilled. This can be done using the current 'WAIT UNTIL' command, but is unsatisfactory. However, this requirement can be satisfied with the 'modified 'WAIT UNTIL' command described in the following.

### NOTE:

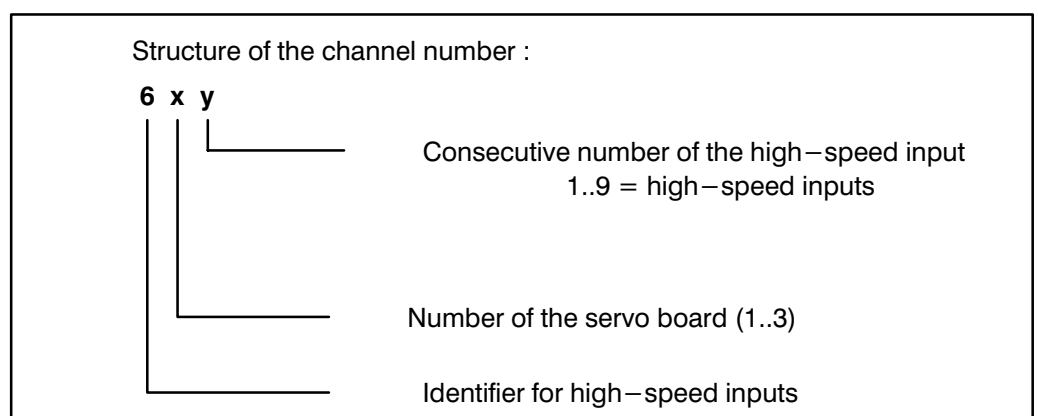
**The 'modified 'WAIT UNTIL' command does not stop block preprocessing. A P2 block (instruction for the interpolator with kinematic reference) is generated. For this reason, the 'modified 'WAIT UNTIL' command must not be used for pure I/O processes as this would otherwise force a P2 synchronization (interpolator cycle synchronization). This would result in an unnecessary kinematics assignment and a sequence delay.**

### NOTE:

The function "Fast, smooth start–off" was improved and expanded in the software versions TO04 and TO05. Please also note the description of the expanded functions in the respective chapters.

### 4.1 Programming

If a specific kinematic is to start off as quickly as possible when a specific condition is satisfied, the condition of the 'WAIT UNTIL' command must be set with the aid of the high–speed inputs of the servo boards which were applied by means of machine parameter P11. This means that the **input channel number 6xy** must be used:



Depending on the hardware configuration, different delays are yielded until movement actually begins:

Hardware configuration	fixed delay	Additional fluctuation width
Servo - i	1 servo board clock position controller cycle	1 P2 - clock (interpolator - cycle)
standard servo in rho 3.2 (CP2.5)	approx. 0.5 interpolator cycle	1 P2 - clock (interpolator - cycle)
standard servo in rho 3.1 (CP/MEM4)	The option in this configuration cannot be used as no HS inputs are available on the servo board.	

A movement is initiated as quickly as possible if the BAPS program contains the following command sequence:

**WAIT UNTIL** *condition*  
**MOVE** *kin1* **LINEAR TO** *P1* .

*kin1*: Name of the kinematics to be moved  
or default kinematics if no name is specified.

*condition* : All the possible logic operations are permissible as a condition of **one** fast input.  
A condition must not consist of a combination of several input channels.

For example, the following are permissible:

HS\_INPUT1 = 1 (high-speed Input)  
HS\_INPUT2 = 0 ( " )

**Note:**

An error output cannot be programmed as a condition (page escape, MAX\_ZEIT).

**Caution !**

If the kinematics which moved before the last 'WAIT UNTIL' differs from the one which is to start off quickly, the control must be informed by means of a dummy traverse block which kinematics are to react quickly:

- (1) MOVE *kin2* LINEAR TO *P0*
- (2) **MOVE\_REL** *kin1* **LINEAR (0,0,0)** ;Dummy MOVE block
- (3) WAIT UNTIL *condition*
- (4) MOVE *kin1* LINEAR TO *P1*
- (5) MOVE *kin2* TO *P2*

If the kinematics change-over (block 2) is forgotten, the 'WAIT UNTIL' command will only be effective for the next traverse block of the kinematics 'kin2', i.e., it will start in block 5. Block 4 would be processed without waiting.

## 4.2 Example:

Steady, fast start-off after high-speed inputs.

```

;;CONTROL = RHO3
;;KINEMATICS : ( 1 = PORTAL , 2 = ZUFUEHRER )
;;PORTAL.JC_NAMES = A1, A2, A3
;;PORTAL.WC_NAMES = X1, Y1, Z1
;;ZUFUEHRER.JC_NAMES = BB
;;ZUFUEHRER.WC_NAMES = XX

PROGRAMM QUICKST
INPUT : 1 = E1 ;NORMAL DIGITAL INPUT
        611 = HSEIN_1, ;1. HIGH-SPEED INPUT ON SERVO BOARD 1
        612 = HSEIN_2, ;2. HIGH-SPEED INPUT ON SERVO BOARD 1
        623 = HSEIN_3, ;3. HIGH-SPEED INPUT ON SERVO BOARD 2
        635 = HSEIN_5 ;5. HIGH-SPEED INPUT ON SERVO BOARD 3

BEGIN
;;KINEMATICS = PORTAL
;;INT = LINEAR
BEG:
MOVE ZUFUEHRER WARTE_POS
MOVE START_POS

WAIT UNTIL HSEIN_1=1
MOVE TO ZIELPOS_1 ;PORTAL (default kin.) moves
; immediately if HSEIN_1=1
MOVE_REL ZUFUEHRER (0) ;so that WAIT command
WAIT UNTIL HSEIN_2=0 ; affects ZUFUEHRER
MOVE ZUFUEHRER TO MAGAZIN ;ZUFUEHRER moves immediately
; if HSEIN_2=0
MOVE_REL (0,0,0) ;so that WAIT command
WAIT UNTIL HSEIN_3=1 ; affects PORTAL
MOVE UNTIL E1=1 TO ZIELPOS_2 ;PORTAL (default kin.) moves
; immediately if HSEIN_3=1
; and E1=0 (in case of E1=1
; no movement)

PARALLEL ;
MOVE_REL ZUFUEHRER (0) ;
WAIT UNTIL HSEIN_5=1 ;
MOVE ZUFUEHRER TO MAGAZIN_3 ;ZUFUEHRER and PORTAL
; both move immediately
; if HSEIN_5=1

ALSO ;
MOVE_REL (0,0,0) ;
WAIT UNTIL HSEIN_5=1 ;
MOVE TO ZIELPOS_3 ;

PARALLEL_END
JUMP BEG
PROGRAMM_END

```



## 5 Setting the belt counter (special function 28)

Previously, the internal belt counters were zeroed by means of the input signals "Reset belt counter". The input signals **Reset belt counter** have been renamed to **Set belt counter**. The bit numbering remains unchanged.

In the future, the internal belt counter will be set with

### Set belt counter

( bit no. 376-383 / PIC address O47.0 - O47.7 / RC input 46.1 - 46.8 )

to the value provided by special function 28. A 'Set belt counter' exists for each belt and special function 28 must be defined as follows (the name of the special function and the variable names can be selected as desired):

#### Declaration of the special function :

**SPC\_FCT: 28 = SET\_BELTCNT** (VALUE INTEGER : *BELT\_NR* VALUE REAL : *RESET\_VALUE* )

<i>BELT_NR</i>	:	Number of the belt for which the value is set
<i>RESET_VALUE</i>	:	Value (in [mm]) to which the internal belt counter is set by means of the corresponding input signal (bit no. and belt no. must agree).
<i>SET_BELTCNT</i>	:	Name of special function can be selected as desired

### 5.1 Example:

```
;;INCLUDE HEADER                                ;Head contains compiler instructions and declarations

PROGRAM SPZ28

SPC_FCT: 28 = SET_BELTCNT (VALUE INT :BELT_NR VALUE REAL :RESET_VALUE)

BEGIN

    SET_BELTCNT (2,-2000)

PROGRAM_END
```

If the signal "Set belt counter" is given for the second belt (RC input 46.2 = "1") **after** the program SPZ28 has run, the internal belt counter is set to the value "-2000".

The reset value remains active until special function 28 is called again (for the same belt).

If the control starts up again, the default reset value is "0".

In case of belt synchronization in a BAPS program, the internal belt counters are also set to the reset value active at the time the program is called; for instance

```
SYNC BELT1, E1=1  
SYNC BELT2 <= 100  
SYNC BELT3 >= 200 .
```

## 5.2 Belt simulation

Using the input signal "Switch on belt simulation", (bit no. 399 / PIC address O49.7 / RC input 48.8), the user is able to simulate a program run while the belt is at a standstill.

If belt simulation is active (input signal = 1), it is not the actual belt speeds (which would be zero with the belts at a standstill) which are considered during program run; rather, the default belt speeds in machine parameter P508 are calculated. The kinematics then move in accordance with these default, fictitious belt speeds.

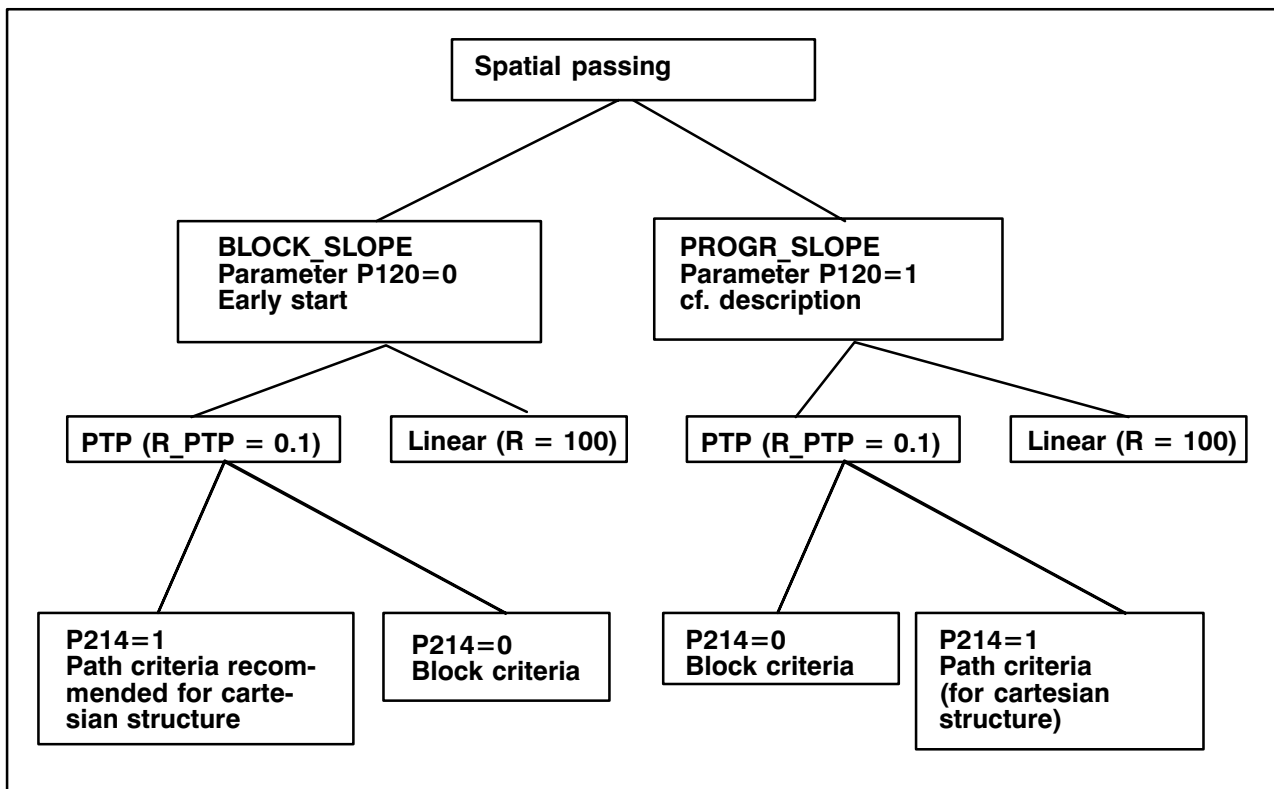
All belts share a common input signal "Switch on belt simulation". A special simulation belt speed in [mm/sec] is applied to each belt by means of P508.

See also rho 3, Description of machine parameters.

## 6 Spatial passing, optimum processing

Programmed target points can be passed over to achieve continuous and harmonious movement for block transitions. This yields an accelerated movement sequence while treating the robot's mechanical components as gently as possible.

Passing, i.e. **intentionally deviating** from the programmed path means guaranteeing a flowing speed profile during block transitions.



### The following generally applies to block slopes:

- Spatial deviation in the passing range depends on speed.
- Individual traverse blocks are started earlier by the amount of the set passing distances.

### The following generally applies to program slopes:

- In a continuous sequence of motions the control tries to keep the speed constant (no braking).

## 6.1 Spatial passing for program slopes

### 6.1.1 PTP passing for program slopes

A distance can be programmed (in degrees or mm) for **each** axis regarding its end position after which a passing operation is initiated. These distances are summed to define a passing range around the target point.

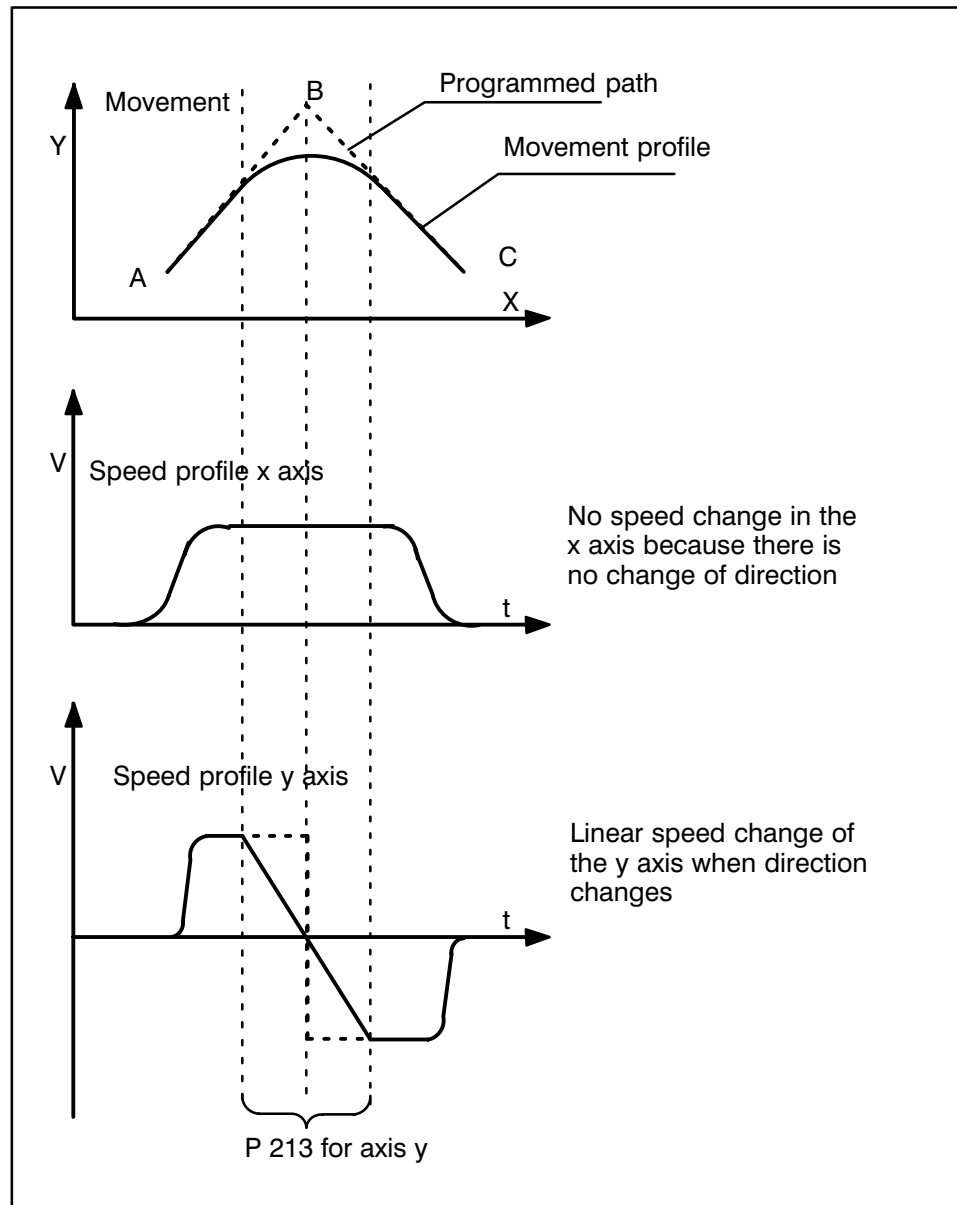
If the operating point enters this passing range during a movement, the robot moves along a path generated by the control which guarantees that the individual axis speeds do not abruptly change. After leaving this passing range, the robot behaviour is identical to movements without programmed passing.

The distances for each axis are stored in the machine parameters. In the user program, these can be changed by means of a factor (**R\_PTP**). The effect is similar to a V\_PTP change. If the factor is programmed, it remains effective until it is written over. All stored distances for individual axes are equally influenced so that the passing range is compressed or stretched. This allows an independent passing range to be defined around each target position which can be optimally adapted to the respective requirements.

The selected distances are decisive in determining the time in which the change in speed is executed at the block transitions. If a relatively small passing range is defined, the speed difference of the consecutive movements must be overcome in a short time. The deviation from the originally programmed path (without passing) is limited to a small range. If large passing ranges are selected, the speed transition is gentler, yet the stretches which deviate from the path are greater.

The transitions between the programmed path and the passing path are tangential.

The following example illustrates the continuous speed change during a block transition. A cartesian system is established with two axes with one in the x direction and one in the y direction. The type of slope is the program slope. The passing range in direction x and direction y is established by means of machine parameter P213. If movement is without passing, speed jumps from +V to -V for the y axis. If passing is active, the speed is changed continuously within the passing range from +V to -V (soft block transitions).

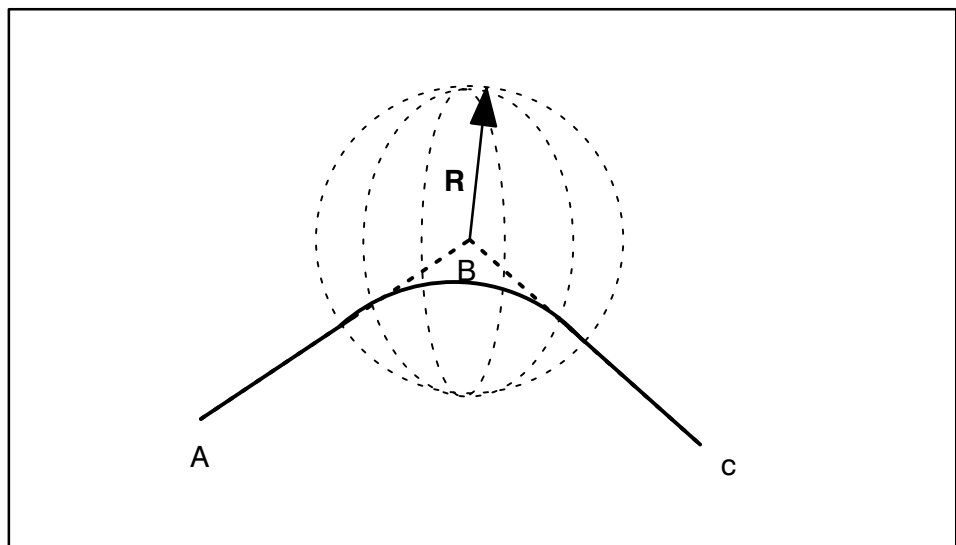


**Note:**

The resulting passing paths depend on the speed.

### 6.1.2 Path passing for program slope

During path passing operations, a three-dimensional sphere is defined around each point which is to be passed. The radius of this sphere is established in the BAPS user program with the standard variable **R**. This makes it possible during path passing to provide each programmed point in space with its own passing range. Between the entry and exit point of this three-dimensional sphere, the original path is abandoned in favor of a harmonious speed profile. Outside of this three-dimensional sphere, the robot takes the path it would have taken in the absence of passing. The same conditions apply for setting the user's optimum passing range as those for PTP passing.

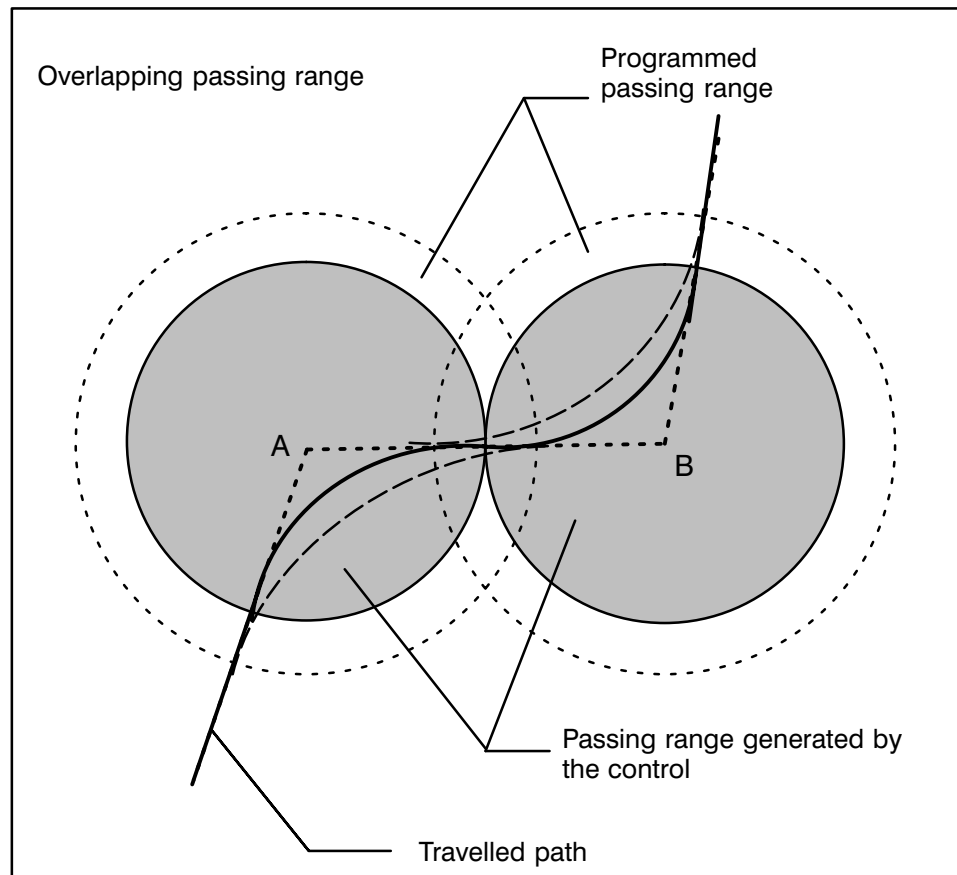


### 6.1.3 Passing when changing the type of interpolation

Passing when the program slope is active is also effective when the type of interpolation is changed. For handling and installation work, it can be advantageous to pass points which are approached in PTP and starting with which linear movements are executed (or vice-versa). The originally programmed path is not deviated from before entering and after exiting the passing sphere. Here, it is not decisive whether linear movements are from the PTP or vice-versa.

### 6.1.4 Special cases

If the distances between the points to be passed are insufficient for maintaining the programmed passing criteria, the control automatically reduces to the maximum possible passing range. In this way, the programmer is freed from adaptation work.



Explanation of the illustration above:

Overlapping passing ranges are automatically reduced by the control to the point that the maximum possible passing range (traced line) is yielded.

**Note:**

Please also note the information on programming program slopes in the BAPS2 programming instructions.

## 6.2 Passing for block slopes

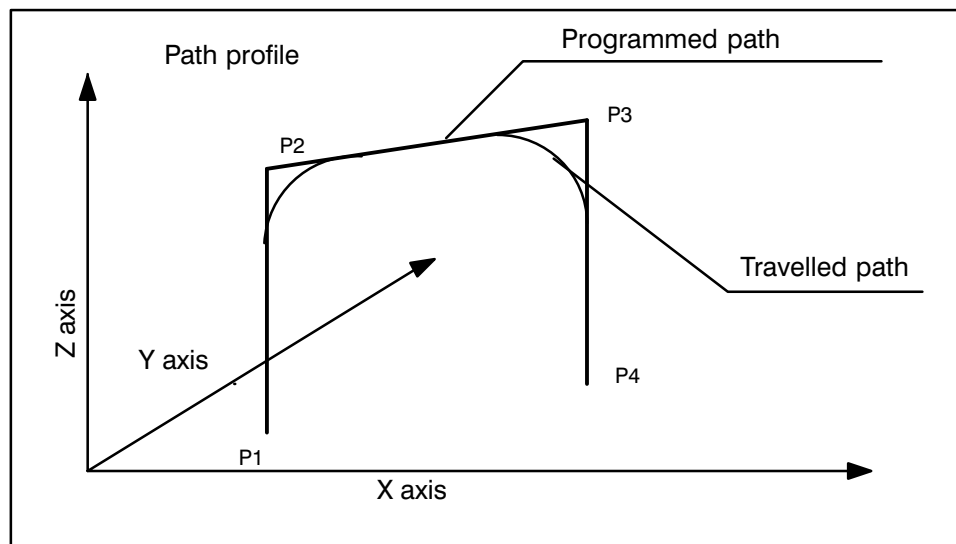
This function is available starting with software version TO03.

### 6.2.1 PTP passing for block slopes

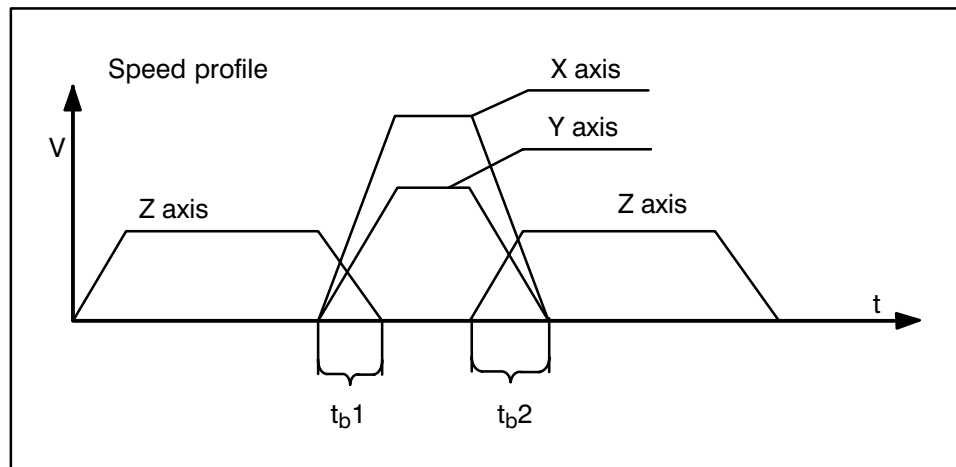
Compared to the slope type **Program slope**, whereby a flowing speed profile is gained by passing while nevertheless maintaining times, in **Block slopes**, the next movement block is started earlier, thus gaining time. A common point in time is determined from the distances specified for each axis regarding their end positions and the following movement block starts at this time. Both block are started parallel to each other and the results are added. This guarantees that the operating point of the robot again reaches the original path travelled along without passing after exiting the passing range. To prevent simultaneous activation of two movement blocks, the control automatically limits the earliest start time to the middle of the previous movement. An initiated movement block is terminated up to the middle of the following block.

The following example serves to clarify movement behavior.

A cartesian system with 3 axes is established whereby the 3rd axis should move up from the working range followed by the movement of axes 1 + 2 in order to then enter the working range together with the 3rd axis. Time gain:  $t_{b1} + t_{b2}$







### 6.2.2 Path passing for block slopes

This function is available with version TO03.  
Path passing with the slope type Block slope affects the path exactly like PTP passing with block slope.

### 6.2.3 Passing when changing the type of interpolation

When block slope is active and the type of interpolation is changed, passing is not executed.

### 6.3 BAPS2 - Language elements

Spatial passing is influenced with the BAPS2 standard variables **R** and **R\_PTP** in a user program.

The language elements (standard variables) act kinematic-specific.

Similar to V programming, an **R** for radius is used for linear interpolation. Absolute values are programmed for **R**.

The unit **R** is established by the measuring system evaluation.

**R\_PTP** acts as a factor in the case of PTP interpolation the same as with V\_PTP.

This factor (R\_PTP) can be specified as a decimal or percent value. It is calculated with the machine parameter P213. The value range for R\_PTP is between 0.01 and 100.

Just as with V or V\_PTP, it is also possible to program the following within the movement command:

'WITH R = ...' or

'WITH R\_PTP = ..'

The **R** or **R\_PTP** then relates only to the block.

#### 6.3.1 Programming

**;; INT = LINEAR** ; Presetting interpolation type

**R = 50** ; Acts beyond block

**MOVE LINEAR WITH R = 20 VIA P2** ; Block related

**;; INT = PTP** ;Presetting interpolation type

**R\_PTP = 1.4** ; Acts beyond block

; or

**R\_PTP = 140%**

**MOVE PTP WITH R\_PTP = 1.4 VIA P2** ; Block related

If R = 0 or R\_PTP = 0 is programmed, spatial passing movement is deactivated.

## 6.4 Machine parameters

The function "Spatial passing" is activated and deactivated by means of machine parameter options. This function is preset (default) when the control is delivered and after an EEPROM backup operation.

At the start of the program, the passing distances and factors set in parameter P212 are taken as default. The default values in the user program can be written over with the BAPS2 standard variables described above. The passing function is deactivated if 0 is the default.

Parameter P212 can be used to define a passing range without having to explicitly program it in the BAPS2 program.

### NOTE:

**We recommend presetting parameter P212 with 0.**

Depending on parameter P214, the axis-specific passing distances or spatial distances are specified in parameter P213. The R\_PTP factor is based on these distances.

The spatial criterium is suitable for cartesian kinematics as the value specified here for PTP interpolation represents a spatial distance. This criterium acts for program slopes as well as for block slopes.

### Presetting the the passing distances and factors :

<b>P212 -</b>	in world coordinates [degrees or mm] - one value per kinematic in joint coordinates [as factor] - one value per kinematic
<b>P213 -</b>	passing distances in joint coordinates [degrees or mm] Axis number values or passing distances in joint coordinates [mm or degrees] - one value per kinematic
<b>P214-</b>	Type of passing movement 0 = Axis criteria 1 = Spatial criteria

## 6.5 Restrictions

Points which are approached in circular interpolation or after which movement is executed with circular interpolation are **not** spatially passed because in this case, due to interpolation itself, the path section is changed in the interpolation cycle. This applies to both program slopes and block slopes.

## 7. Protocol 3964/R

The READ/WRITE device function packs and unpacks the data of a BAPS2 program into and out of the protocol 3964R which is stored in blocks.

Use of this function is optional. Protocol 3964R is set as protocol 8, the same as with the other 7, under MODE 9.1 or MODE 7.8.3 in the interface-dependent subpoint 3, 4, 5 or 6.

The operating system then implements the data link layer for the data intended for transmission in protocol 3964R, with establishment of connections, block saving, time monitoring, block repeating, etc.

As opposed to the previous protocols, data is transmitted with this protocol via the interface in both directions. Additionally, the data are not converted, but are transmitted in binary form. For this reason, the points listed here require attention.

### 7.1. Implementing the protocol driver

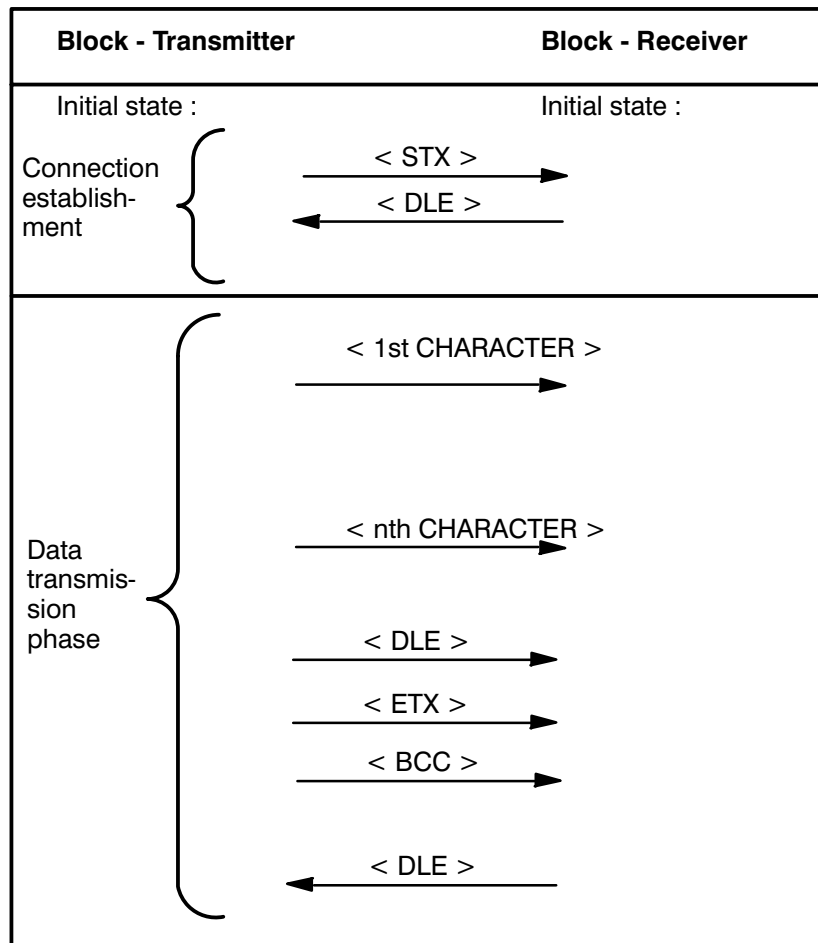
During control start, a driver task is activated for each BAPS interface for handling the protocol 3964R (establishing connections, data linking, time monitoring, and when required block repeating).

This driver task receives and acknowledges a maximum of one data block, even when READ itself is not yet active. In addition, the driver task synchronizes the transmission and reception of data blocks.

#### **NOTE:**

As opposed to READ/WRITE with the previous protocols, which only temporarily occupied a respective interface, using protocol 3964R means that the driver task **permanently occupies the interface** so that it cannot be used for other functions.

In the absence of faults the protocol 3964R appears as follows:



A connection must be established for each data block because a connection is considered cleared with the positive acknowledgement of the block check sum **<BCC>**.

If the block receiver responds to a connection attempt with a character other than **<DLE>**, or doesn't respond within the acknowledgement delay time of 550ms, the block transmitter repeats the character **<STX>** a maximum of 5 times (for a total of 6 attempts) to establish connection.

If a connection is established, the block transmitter transmits the data whereby a **DLE** contained in the data is transmitted twice. The data block may be up to 138 characters long (without duplicate characters). The data are followed by the characters **<DLE> <ETX>** as end-of-block identifiers and end with the block check sum **<BCC>** as an XOR link operation linking all the characters of the message except the start control character **<STX>**.

While data are transmitted, the block receiver waits for additional characters during the character delay time of 220ms until **<BCC>** is received; otherwise, it aborts reception, transmits the character **<NAK>**, and returns to initial state.

During transmission of <**BCC**>, the block transmitter waits for a response from the block receiver within the acknowledgement delay time. If the block receiver responds with a character other than <**DLE**>, in particular <**NAK**>, or does not respond within the acknowledgement delay time, the block transmitter repeats the block a maximum of 5 times (for a total of 6 attempts).

If both ends attempt to establish connection simultaneously, the end without priority must wait. The side with priority then begins data transmission as described. As the timeout times for this protocol are permanently established (see delay times), the priority for conflict cases is set with the output timeout time. If output timeout time value is -1, the driver will function without priority. The transmission priority is a part of the interface parameter.

## 7.2. Interfacing with the BAPS program

The following **BAPS data types** can be read and written:

**BINARY,**  
**CHARACTER,**  
**INTEGER,**  
**REAL, POINT,**  
**JC\_POINT,**  
**TEXT**

In addition, **one-dimensional fields** can also be read and written.

Using the type **TEXT** for transmitting binary data can pose problems for some **BAPS** operations because of the special position of the **NIL** character as the end character, particularly because, as a rule, the user protocol for a **3964R** connection begins with two **NIL** characters. In addition, a **3964R** user data block can be up to 128 characters (>80) long (plus a maximum of 10 bytes for the message header of the application layer).

Because only the data link layer of protocol **3964R** is implemented, the application layer of the protocol must be implemented in the **BAPS** program. We recommend writing subroutines for converting the **BAPS**-specific representation of data to user-specific representation of data and vice versa (for example, for the conversion of an **INTEGER** value to a **WORD** representation common to the **PLC** and vice versa).

Type	Number [byte]	Representation, value range
BINARY	1	0 or 1
CHARACTER	1	0 .. 255
INTEGER	4	Byte sequence set via options normal : lower value byte first
REAL	4	IEEE format, byte sequence as with INTEGER
POINT	4 * Coord. number	Each coordinate as a REAL, without separator
JC_POINT	4 * Axis number	Each axis as a REAL, without separator
TEXT	80	Each byte as a CHARACTER
Fields of type	Field size	Each element as a data type [BINARY, CHARACTER, INTEGER, REAL, POINT, JC_POINT]

For data of types POINT and JC\_POINT, belt coordinates are included in the byte count.

### 7.3. WRITE device

For WRITE, data are copied in binary form to the output block without conversion. The number of bytes depends on the BAPS data type; see the table above.

If several programmed expressions are separated with commas in a WRITE command, the data are copied consecutively, as indicated above, to an output block and the entire block, saved by protocol 3964R, is then transmitted.

### 7.4. READ device

During a READ operation, the same number of bytes are read from the received block as are copied during a WRITE operation. If all the bytes of the received block are not read in one READ operation, data are continuously read in each following READ operation. Data are only read out of a new received block once the preceding received block has been completely read. However, a block limit must not be within a data type which is to be read. Otherwise, this would lead to the runtime error "Protocol error READ". Refer to "READ/WRITE device with protocol 3964R" V1.2. Because the data are transmitted in binary form, data values can only be verified to be either 0 or 1 for variables which are BINARY. For DEC type variables, or for POINT or JC\_POINT components, only an FPU-Trap system error can be intercepted and converted to a runtime error "Protocol error READ". Whether the read data values are of a practical magnitude must therefore be checked in the BAPS program.

## 7.5. Special cases and restrictions

The total execution time for reading a data module of a given PLC is mostly determined (provided transmission at 9600 baud is successful) by the execution time required for generating and in particular interpreting the 3964R application layer in the BAPS program.

If a program is aborted but has already read at least 1 byte, the rest of the block is deleted in order to make a restart possible.

If the physical interface is not available at start up or already occupied, for example with the function **"COUPLING TO THE PROGRAMMING UNIT"**, the driver task is not initialized.



## 8. Multiple use of the serial interfaces

The operating system RBS of the rho 3 control provides more functions which copy or read data at serial interfaces than are provided on interfaces at the hardware end.

These functions are:

**Coupling** to the programming unit (also online functions)

**Printing** files or machine parameters

**WRITE/READ**V24\_1 .. V24\_4

On the CP/MEM4 hardware version, a combined V24/20mA interface and a pure V24 interface are available.

On the hardware version CP2.5 (with integrated interface board), three combined V24/20mA interfaces and one RS422 interface are available.

Allocation between function and the serial interface is made in Mode 9.1 or Mode 7.8.3.

If the same serial interface is selected for two or more functions in these modes, a conflict may arise if these functions attempt to use the interface simultaneously.

Note that certain functions will constantly occupy an interface. These are the functions which couple to the programming unit, and the functions WRITE/READ V24\_x if protocol 3964R (8) is set.

### 8.1. Automatic reactions to a conflict

If a function (B) wants to use an interface which is already occupied by another function (A), the RBS has function (B) wait until function (A) is concluded.

In WRITE V24\_X, a function is considered concluded if a contiguous WRITE instruction has been issued, and for READ if exactly one variable has been read.

If several processes want to use the same function, the above behavior also applies to their access operations. If several processes are queued for allocation, the priority of a given process determines its place in the allocation.

Special cases:

The functions COUPLING and WRITE/READ V24\_x with protocol 3964R do not clear the interfaces so that a function (B) would wait indefinitely for the conclusion of a function (A).

WRITE/READ PHG or V24\_x with PHG protocol (6) stops the RBS user interface, uses the interface (for the PHG), and ensures at its end that the RBS user interface is continued.

## 8.2. Controlled avoidance of conflicts

To effectively avoid conflict situations, all 6 functions are provided with disable input signals at the internal interface.

13.1 : DISCOUPL\_RCI

13.2 : DISPRINT\_RCI

13.3 : DISSER\_1\_RCI

13.4 : DISSER\_2\_RCI

13.5 : DISSER\_3\_RCI

13.6 : DISSER\_4\_RCI

If these signals change from 0 to 1 while the function is active, the function is aborted (CONDITION = interface error).

If these signals are high when a function is active, the function is not executed. (CONDITION = interface occupied)

If these signals change from 1 to 0, functions COUPLING and READ V24\_x can be directly continued with protocol 3964R, and the others only after a new function call. If the transmission rate of the interface changes, it is re-initialized. If there are still characters in the reading FIFO of the interface they are deleted.

If these signals are low, operations are executed in accordance with the method described above.

Normal signal state is 0. The blocking signals are only necessary if function conflicts arise and a conflict strategy is desired other than the automatic one, or if the conflict situation cannot be automatically solved.

## 8.3. Replugging or reswitching the interface

As a rule, when the interface is replugged or reswitched, an invalid character is read which, for the function READ device and in accordance with the protocol, can lead to the runtime error "Protocol error READ" or to the STATE

"Framing error".

This can be prevented by:

- Disabling the function (A)
- Replugging or reswitching
- Enabling the function (B)

#### 8.4. Status display

If the coupling function is deactivated by means of its input signal, the following message appears in the **Info function**:

**Disable coupling PG '**

#### 8.5. Example

The functions COUPLING and READ/WRITE V24\_1 with protocol 3964R are to be used at the same interface. For changing over, a digital input COUPL\_ON\_DI exists; for example, a signal provided by an interface changeover switch.

The PIC program would then appear as follows:

```
DISCOUPL_RCI = NOT COUPL_ON_DI
```

```
DISSER_1_RCI = COUPL_ON_DI
```

For some parts of the BAPS program, it may be helpful to have the changeover signal available; for example:

**PIC program:**

```
USER_1_RCI = NOT COUPL_ON_DI
```

**BAPS program:**

```
INPUT BINARY : 1 = V24_1_FREI
....
WAIT UNTIL V24_1_FREI=1
OR
ZUST_V24_1 = CONDITION (V24_1)
IF (ZUST_V24_1 = -7) ; interface error
OR (ZUST_V24_1 = -2) ; interface busy
THEN WAIT UNTIL V24_1_FREI=1
```

Variant:

The BAPS program switches the interface as required by means of an OUTPUT.

**BAPS program:**

```
OUTPUT BINARY : 1 = JETZT_V24_1
```

```
INTEGER : ZUST_V24_1
```

```
...
```

```
JETZT_V24_1 = 1
```

```
READ V24_1, Variables
```

```
...
```

```
WRITE V24_1, Variables
```

```
ZUST_V24_1 = CONDITION (V24_1) ; for synchronization
```

```
JETZT_V24_1 = 0
```

**PIC program:**

```
DISCOUPL_RCI = USER_1_RCO
```

```
DISSER_1_RCI = NOT USER_1_RCO
```

For a controllable interface changeover switch, this signal will then be output on a digital output.

## 8.6. Notes on implementation

If an automatic changeover occurs by means of the PLC, the difficulty arises that the PLC is not able to recognize whether data are being transferred at the interface at that moment, e.g., by the coupling. For READ device, synchronization by means of user signals is possible, and for WRITE device, the transmission must be allowed to end with the command CONDITION (device).

There is no check-back signal indicating that the changeover procedure is concluded.

Assigning the functions COUPLING and READ/WRITE device to the physical interface is an indirect machine parameter. The setting cannot be read by the BAPS program or by the PLC program so that conflict handling must be provided for as a fixed component of the programs.

If the COUPLING function is deactivated, ONLINE status functions, ONLINE test and file transfer operations are not possible. Calling a coupling operation at the programming unit end (ROPS3) leads to a "Time out error".

## 9 **CONDITION inquiry of interfaces**

In BAPS it is currently not possible to check the status of an interface. If a copy operation is performed to an interface that is not connected, for example, a Timeout error results and the running program is aborted.

The new compiler instruction `SER_IO_STOP` can prevent the user program from being aborted if an I/O error occurs.

The standard function **CONDITION** allows the status of an interface to be determined.

The possible conditions are encoded and provided to the program for evaluation by means of the return value of the function (data type `INTEGER`). The function value can be evaluated by the BAPS programmer and the corresponding reaction can be initiated.

The following interfaces are supported :

- V24\_1 to V24\_4
- PHG
- TTY

### 9.1 **Compiler instruction SER\_IO\_STOP**

This compiler instruction can be used to prevent the user program from being aborted if an interface error occurs.

**Syntax :**

**;; SER\_IO\_STOP [+/-]**

The instruction is only permissible at the beginning of the program, is valid for the complete program, and may occur only once. If the compiler instruction is not used, an interface error leads to a program abort.

<b>Syntax</b>	<b>Effect</b>
<code>;; SER_IO_STOP -</code>	No program abort for error
<code>;; SER_IO_STOP +</code>	Program abort for error
<code>;; SER_IO_STOP</code>	Program abort for error

## 9.2 CONDITION function and interface

The desired interface (for example, V24\_1) is provided to the standard function **CONDITION**. The return value of type INTEGER can be queried.

### Example

**IF CONDITION ( V24\_1 ) < 0 THEN HALT**

The following values are provided by the standard function CONDITION :	
Function value	Meaning
0	No error
-1	No interface
-2	Interface occupied
-3	Time out
-4	Parity error
-5	Overrun error
-6	Framing error
-7	General interface error
-8	Incorrect character string length
-9	Protocol error
-10	Illegal DEC value
-11	Point not defined

### Example :

```

;; SER_IO_STOP - ; No program abort for
;faulty communication

PROGRAM IO_TEST
  INTEGER : Index, Anzahl

  BEGIN

  Anzahl = 0
  READ V24_1, Index
  IF CONDITION ( V24_1 ) < 0 THEN
    BEGIN
      WRITE 'general READ error V24_1'
      HALT
    END
  ELSE
    REPEAT Index TIMES
      Anzahl = Anzahl + 1
    REPEAT_END

PROGRAM_END

```

### 9.3 CONDITION function and file

Files are permissible as arguments for the function CONDITION.

**Example :**

; Declaration of the file

**FILE :** DATA stand for any given file name

; Query whether file available

**IF CONDITION (DATA) < 0 THEN JUMP K\_DAT**  
**READ\_BEGIN (DATA)**

....

...

K\_DAT:

The following values are provided by the standard function CONDITION :	
Function value	Meaning
0	No error
-1	No file

## 10 Standard procedures

### 10.1 Standard procedure INT\_ASC

applies as of  
Version  
TO02A

The standard procedure **INT\_ASC** converts an integer number to an array of characters. If an error is detected during conversion process, the 'character array' remains unchanged and the corresponding error code is returned.

**INTEGER:** int\_number  
**INTEGER:** index\_ca  
**INTEGER:** length\_char  
**INTEGER:** error\_rue  
**ARRAY[1..10]**  
    **CHAR:** ascii\_array

int\_number = -1234  
index\_ca = 1  
length\_char = 5

**INT\_ASC**  
(int\_number,  
  ascii\_array,  
  index\_ca,  
  length\_char,  
  error\_rue)

```
IF
  error_rue <> 0
THEN
  WRITE 'error in
  INTEGER-ASCII conver-
  sion'
ELSE
  WRITE ascii_array
```

----- PRESET -----  
int\_number = number to be converted  
            ( only type INTEGER permitted  
            | int\_number | <= 2147483647)

---- RESULT ACKNOWLEDGMENT ----  
ascii\_array = Result array of  
            Type ARRAY [ ] char  
            (The destination area is initialized  
            with blanks before conversion)

----- PRESET -----  
index\_ca = Start index in the character array

----- PRESET -----  
length\_char = Maximum number of characters  
            reserved for the number to be converted

---- RESULT ACKNOWLEDGMENT ----  
error\_rue = Error number output

0	No errors
-1	Start index outside array limits
-2	End index (start index + length) outside array limits
-3	Reserved length too small
-4	Range transgression (value too high)
-5	Array length < 0
-6	Array length = 0



**10.2 Standard procedure ASC\_INT**

```

INTEGER: int_number
INTEGER: index_ca
INTEGER: length_char
INTEGER: error_rue
ARRAY [1..10]
CHAR: ascii_array

ascii_array = 'AaBc'
index_ca = 1
length_char = 5

ASC_INT
(ascii_array,
int_number,
index_ca,
length_char,
error_rue)

IF
error_rue <> 0
THEN
WRITE 'error in
ASCII-INTEGER conver
sion'
ELSE
WRITE int_number

```

applies as of  
version  
TO02A

The standard procedure **ASC\_INT** converts an array of characters to an integer. The procedure reads in characters as of the start position until:

- a character which is not a digit is detected
- the maximum number of characters has been read
- the end of the character array is reached

----- PRESET -----  
ascii\_array = array of type ARRAY [ ] char  
to be converted

---- RESULT ACKNOWLEDGMENT ----  
integer\_number = converted number  
(only type INTEGER permitted)

----- PRESET -----  
index\_ca = Position in the array as of which  
the number is to be read

----- PRESET -----  
length\_char = Maximum number of  
characters to be read

---- RESULT ACKNOWLEDGMENT ----  
error\_rue = Error number output

0	No errors
-1	Start index outside the array limits
-2	End index (start index + length) lies outside the array limits
-3	-----
-4	Range transgression (value too high)
-5	Array length < 0
-6	Array length = 0
-7	Character string does not start with a number or sign

## 11 PUBLIC variable

### Example program

```

;Exporting program
PROGRAM exp_var

;PUBLIC data
PUBLIC DEF POINT: start_pos
PUBLIC INTEGER: index
PUBLIC
SEMAPHORE: writeprotect

REAL: mvalue

BEGIN
;BAPS statements
EXCLUSIVE writeprotect
start_pos =POS
index=5
EXCLUSIVE_END
;...
;... further statements
;...
PROGRAM_END

;Importing program
PROGRAM imp_var

;PUBLIC data
EXTERNAL exp_var: start_pos
EXTERNAL exp_var: index
EXTERNAL exp_var: writeprotect

;local data
POINT: end_pos

BEGIN
;BAPS statements
EXCLUSIVE writeprotect
RPT index TIMES

MOVE TO start_pos
MOVE TO end_pos
RPT_END
EXCLUSIVE_END
;
PROGRAM_END

```

PUBLIC variables allow the simple exchange of data between several independent BAPS2 user programs (processes). The fundamental principle is to group programs which work with the same PUBLIC variables into program groups. Within this group, one program may export data while others only import these data. The control can contain as many of these program groups as desired (limited only by available memory).

#### Declaration part in the **exporting program**:

PUBLIC variables are declared by adopting the reserved word PUBLIC in the type declaration of these variables. The data range for these variables is reserved in the IRD file. Teach points (for example identified by the key word DEF) are stored in the PNT file.

#### Declaration part in the **importing program**:

The variables can be accessed by other BAPS programs if the variables are declared with EXTERNAL and the name of the other exporting program.

#### Restrictions

Consider the following restrictions when using PUBLIC data :

1. The exporting program of a program group must have been compiled before the importing program of this group (due to type check by the compiler). This means that the user must ensure that importing programs are more recent than the exporting program. If this is not the case, an error message or a warning is issued at the start of the program.
2. In a program, it is not possible to import and export data simultaneously. Only data from one program may be imported.
- 3.1 PUBLIC data may be exported or imported only in the declaration part of the main program, and not in the sub-routines.
- 3.2 Access to the variables must be prevented by use of the EXCLUSIVE statement before interruption in order to guarantee data consistency.  
The consistency of simple standard types such as BINARY, INTEGER, REAL, CHAR is provided by the operating system.

## 12 Kinematic-related automatic - manual mode

Starting with operating system version TO02F, it is possible in rho3 operations to operate with several kinematics in the **AUTOMATIC MODE** and **simultaneously** operate with several in the **MANUAL MODE**. The rho3 offers this capability in addition to the previous global AUTOMATIC/MANUAL mode.

There is no longer a strict separation between AUTOMATIC MODE and MANUAL MODE.

When taking advantage of this option, the user must take safety measures to guarantee that movements in AUTOMATIC MODE do not endanger an operator who is working at the same time with other kinematics in the MANUAL MODE.

### 12.1 New Signals and their meanings

<b>RC inputs</b>				
BAPS name	Bit no.	PIC addr.	RC input	Signal description
KIN_A_MN_RCI	54	O 6.6	5.7	Automatic/Manual per kinematic
K01_A_MN_RCI	38	O 4.6	3.7	Auto/Manual, not kin. 1
K02_A_MN_RCI	39	O 4.7	3.8	Auto/Manual, not kin. 2
K03_A_MN_RCI	40	O 5.0	4.1	Auto/Manual, not kin. 3
K04_A_MN_RCI	41	O 5.1	4.2	Auto/Manual, not kin. 4
K05_A_MN_RCI	42	O 5.2	4.3	Auto/Manual, not kin. 5
K06_A_MN_RCI	43	O 5.3	4.4	Auto/Manual, not kin. 6
K07_A_MN_RCI	44	O 5.4	4.5	Auto/Manual, not kin. 7
K08_A_MN_RCI	45	O 5.5	4.6	Auto/Manual, not kin. 8
K09_A_MN_RCI	46	O 5.6	4.7	Auto/Manual, not kin. 9
K10_A_MN_RCI	47	O 5.7	4.8	Auto/Manual, not kin. 10

#### **KIN\_A\_MN\_RCI :**

If this signal is set, the AUTOMATIC/MANUAL mode per kinematic is active and kinematic-related signals are used to distinguish automatic and manual mode:

K01\_A\_MN\_RCI through K10\_A\_MN\_RCI.

The signal AUTO\_MN\_RCI, PIC address O.7 has no meaning in this case.

If this signal is not set, the signal AUTO\_MN\_RCI, PIC address O2.7 acts globally for all kinematics and the option AUTOMATIC/MANUAL mode is inactive. The signals K01\_A\_MN\_RCI through K10\_A\_MN\_RCI have no meaning in this case.

#### **Kxx\_A\_MN\_RCI :**

xx=01..10 designates the kinematic (for example, xx=03 means kinematic 3). The signal has the same meaning as the global signal AUTO\_MN\_RCI related to kinematics xx. These signals switch the kinematics xx into AUTOMATIC or MANUAL mode.

### 12.1.1 Signal meanings

If KIN\_A\_MN\_RCI is set, some signals receive a somewhat different meaning. If the signal is not set, the meanings remain the same.

**POWERRED\_RCO, PIC address I8.4 (power reduction) :**

This signal is only set to "0" if all kinematics are in automatic mode and no "Emergency mode without RC" is pending.

Otherwise the signal is "1".

**TEST\_MN\_RCO, PIC address I12.1 (TEST/MANUAL, not(copy)) :**

This signal is set to the status of TEST\_MN\_RCI, PIC address O2.6 (Test/Manual, not) when at least one kinematic is in manual mode.

Otherwise the signal is "0".

**PHG\_ACTV\_RCO, PIC address I9.0 (PHG operation and manual are active) :**

This signal is set to "1" if at least one kinematic is in manual mode.

Otherwise operation is as before.

**AUTO\_MN\_RCO, PIC address I12.0 (Automatic/Manual, not) :**

This signal is set to "1" if all kinematics are in automatic mode.

Otherwise the signal is "0".

### 12.2 Automatic mode

In principle, the same things apply to kinematic-related automatic mode as to the previous global automatic mode.

The reference points of all kinematics must be approached before programs with traversing movements can be started.

It is not necessary to deactivate the PHG by means of Mode 8 before switching from the kinematic-specific manual mode to the kinematic-specific automatic mode.

User programs can be started regardless of the signal status of the individual kinematic-specific Kxx\_A\_MN\_RCI. If a block is to be prepared that is responsible for kinematics which are not in automatic mode, the runtime error "Inadmis. in manual op." is issued.

If the signal Kxx\_A\_MN\_RCI is set from "1" to "0" during execution of a kinematic-specific block (switching the kinematic xx from automatic to manual), the runtime error "AUTO/MANUAL switching" is issued. When this happens, no program with traversing movements which uses changed over kinematics can be executed before the program in question has been deselected or is reset.

### 12.3 Referencing and Set-up

With referencing, the same applies as before with the exception that only those axes whose kinematics are in manual mode can be started for referencing.

## 12.4 Teach in

Teach in is possible for those points only whose respective kinematics are in manual mode. Otherwise operation is the same as before for **global AUTOMATIC/MANUAL**.

## 12.5 Testing

If kinematic-specific blocks are to be executed during testing (for example, MOVE...), a check is made whether the corresponding kinematics are in manual mode. If this is not the case an error message is issued. Otherwise operation is the same as before for global AUTOMATIC/MANUAL.

## 12.6 Changing from global AUTOMATIC/MANUAL mode to AUTOMATIC/MANUAL MODE per kinematics

When switching over, the following signal sequences must be maintained. The signals Kxx\_A\_MN\_RCI must be brought to the same state as the signal AUTO\_MN\_RCI, PIC address O2.7 (Automatic/Manual, not). The signal KIN\_A\_MN\_RCI must then be changed.

## 12.7 Notes on proper use of options

With this option, partially or temporarily independent kinematics can be operated with one control in different operating modes.

In order to guarantee trouble-free operations, program selection and deselection and changing between the operating modes Automatic and Manual should be implemented by means of the interfacing control (PIC) program.

In the PIC, the corresponding program should be deselected before switching from Automatic to Manual. In addition, before selecting a program it should be ensured that the corresponding kinematic is in automatic mode. When using this option, we recommend operating only one kinematics in a given program. This ensures that the kinematics in the BAPS program are separate.

## 12.8 Diagnosis on the PHG

**Automatic** is shown in the basic menu of the PHG display only if all kinematics are in automatic mode. If some kinematics are not in automatic mode, the message "Manual" is issued.

Under Mode 7.3.3 (DIAGNOSIS.SYSTEM CONDITIONS.ACT. OPERATING MODE), the current operating mode of each kinematic can be displayed.

## 13 External program/process deselection

This option allows the user to stop the program/processes he is running, even by means of the PLC/PIC, similar to **external program selection**.

Just as with external program selection, using an 8-bit data channel, a parity bit and a strobe signal, the process to be stopped can be selected from the

### **EXPROG.DAT file.**

The same signals are used as for external program selection.

Either the message "Process aborted" is used as acknowledgement, or, if this is not possible, the message "Error with external process abort" as a strobe signal to the interface.

If a process abort is successful, the process is stopped directly, meaning for example that an active movement is not completed.

When a main process is stopped, all sub-processes it started are simultaneously stopped as well.

Permanent processes are stopped just like normal processes, regardless of the signal "permanent processes shall remain active" (PIC address O7.5).

### 13.1 RC input signals

To select a process to be stopped, the eight data bits of the external program selection, the **abort signal** (strobe), as well as a parity bit are required.

<b>RC inputs</b>				
BAPS name	Bit no.	PIC addr.	RC input.	Signal meaning
STBEXPS_RCI	80	O10.0	9.1	Strobe. External process abort
PTY_EXTP_RCI	83	O10.3	9.4	Parity, external prog. Selection/Abort
EXTPRG_0_RCI	88	O11.0	10.1	External prog.
EXTPRG_1_RCI	89	O11.1	10.2	Selection/Abort
EXTPRG_2_RCI	90	O11.2	10.3	Bit 1 - 8
EXTPRG_3_RCI	91	O11.3	10.4	
EXTPRG_4_RCI	92	O11.4	10.5	
EXTPRG_5_RCI	93	O11.5	10.6	
EXTPRG_6_RCI	94	O11.6	10.7	
EXTPRG_7_RCI	95	O11.7	10.8	

### 13.2 RC output signals

When a selected process is successfully aborted, the acknowledgement signal "Process has been aborted", no. 116, PIC address I14.4, is set.

If it was not possible to abort the process, e.g., because no process with a corresponding name exists, or if the parity is incorrect, the signal

"error with external process abort", no. 117, PIC address I14.5 is set.

Strobe times for these signals are set with **machine parameter P9**.

### 13.3 Restrictions

Selecting a process by means of external program selection and simultaneously stopping another process or the same process by means of "external program abort" is prohibited. This means that both strobes may **not** be pending simultaneously.

A corresponding interlocking capability must be implemented by the user in his interface program (PLC/PIC).

### 13.4 EXPROG.DAT structure

In the file **EXPROG.DAT**, the coordination is established between the code value and the program which is to be aborted when the code value and strobe are pending. The code value must be entered in hexadecimal form.

Structure of the file <b>EXPROG.DAT</b>	
<b>Code value program name (name of the IRD file)</b>	
<b>; Example of an EXPROG.DAT</b>	
<b>00 = PROG1</b>	<b>; Comment separated by semicolon</b>
<b>02 = PROGXY</b>	
<b>03 = PROGAB</b>	
<b>:</b>	
<b>:</b>	
<b>FF = PROGFF</b>	<b>;</b>

## 14 Setting potentiometer values

The function "Optional setting/non-setting of the global and kinematic-related potentiometer values to 100%" (=1.0) is used for Control Reset, AUTO==>MAN.==>AUTO and Process selection/start.

The global potentiometer values or **global factors** are those values, or factors, which are set by means of the PHG (Mode 11, sub-modes 4, 5, 6) as well as interfaces. They cannot be addressed in the BAPS2 program.

The **kinematic-dependent factors** can only be changed by the BAPS2 program. They are addressed with the following:

"kinematicname.VFACTOR"

"kinematicname.AFACTOR" or

"kinematicname.DFACTOR"

The option described below was implemented to flexibly meet the requirements of the user without changing the operating system. This function is available as of version TO02F.

### 14.1 Description of function

The different reset variants are established by means of an option byte machine parameter.

Each bit of this byte controls behaviour for the following functions:

" **Control reset**"

" **Process selection/Start**" ,

" **Automatic- / Manual switching**" and

" **Manual- / Automaticswitching**".

The respective option address can be taken from the option list. If the option byte = 0 (which corresponds to the power-up state after EEPROM backup), the control behaves as previously before this function was introduced.



**14.1.1 Option byte=0**

The following table shows how the individual factors are influenced by the following operations.

Operation	Global factor	Kin.-dep. factor.
CONTROL RESET	1.0	unchanged
AUTO ==> MANUAL	unchanged	unchanged
MANUAL ==> AUTO	unchanged	unchanged
PROCESS SELECTION	unchanged	1.0

During control Start-up, in each case the global and kinematic-dependent factors are set to 100% (1.0).

If one of the bits in the options byte is set, the result is as described in the following: Note that all the bits corresponding to the desired (previous) behaviour may be set to "1" or "0".

**14.1.2 Option byte<>0**

The following table shows how the individual factors are influenced by the following operations.

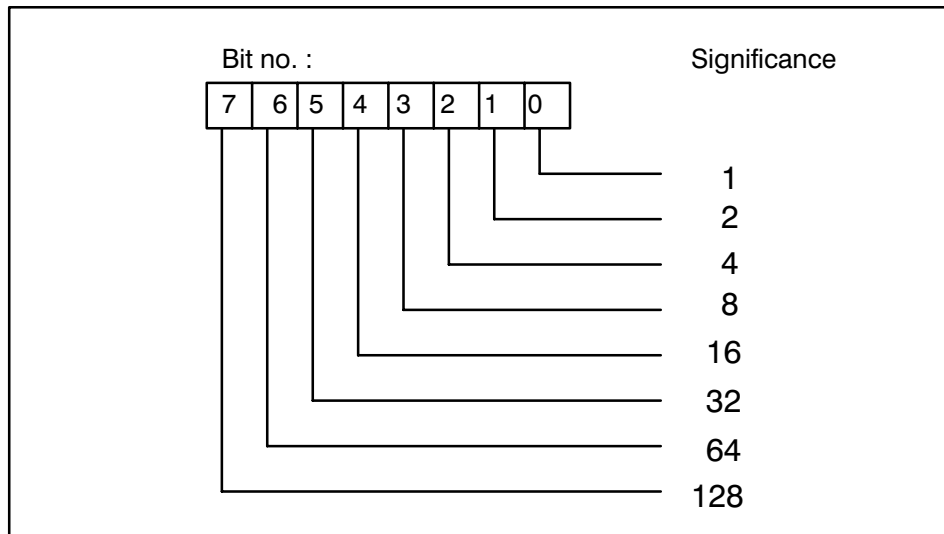
Bit no.	Significance	Kin.-dep. factor for	becomes/remains
0	1	CONTROL RESET	1.0
1	2	AUTO ==> MANUAL	1.0
2	4	MANUAL ==> AUTO	1.0
3	8	Process selection/Start"	unchanged

Bit no.	Significance	Global factor for	becomes/remains
4	16	CONTROL RESET	unchanged
5	32	AUTO ==>MANUAL	1.0
6	64	MANUAL ==>> AUTO	1.0
7	128	PROCESS SELECTION/START	1.0 (*)

(\*) Note: Acts on all kinematics, including kinematics already moved in another process.

Because input is bit-by-bit, the significance of the individual bits is provided in the following table for simpler handling. The value to be entered is equal to the sum of all the individual values.

**Option byte bit number with respective significance**



**Example:**

Desired here is that the kinematic-dependent factors for CONTROL RESET and the global factors for AUTO ==> MANUAL and MANUAL ==> AUTO are set to 1.0.

Option byte = 1 + 32 + 64 = 97

## 15 INFO function on the PHG

The INFO function is used for providing information to the operator during rho 3 operations in case a selected function in a menu branch has not been executed.

Assume the user wants to approach the reference points. He has selected Mode 1 REFERENCE POINTS and attempts to move the axes. The control does not react because, for example, there is no control enable signal, or referencing has been deactivated by means of machine parameter **P 402**.

Selecting the INFO function shows the operator information which indicates why the selected function was not executed. If this is not due to a malfunction, the message "No info available" appears.

When the INFO function is exited, operations can be resumed at the point where INFO was called.

### 15.1 Availability of the INFO function

The INFO function can be called up in each PHG menu branch.

Currently, the following information can be provided in the modes listed below:

- Mode 1 Reference points
- Mode 2 Manual
- Mode 3 Progr. BAPS/PIC
- Mode 3.1 Programming BAPS
- Mode 3.2 Programming PIC
- Mode 4 Define/Teach in
- Mode 4.1 Define
- Mode 4.2 Teach in
- Mode 5 Test BAPS progr.
- Mode 6 PIC monitor
- Mode 6.1 PIC monitor on
- Mode 6.4 Timer and counter (change)
- Mode 7 Diagnosis
- Mode 7.1 Axis display
- Mode 7.8 Mach. parameter
- Mode 9 Device/File I/O
- Mode 9.1 Default
- Mode 10 Select program
- Mode 10.1 Select processes
- Mode 10.2 Stop processes
- Mode 11 Help functions

The remaining modes issue the following message when the INFO function is called:

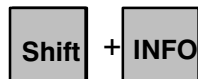
' **no INFO messages** '

## 15.2 Restrictions

The INFO function does not provide messages in case of compiler errors, for instance, which are reported directly by the compiler anyway. In addition, the INFO function does not provide information on pending system errors.

## 15.3 Operation

The INFO function is selected by simultaneously pressing the following keys:



Exit the INFO function or acknowledge error messages by pressing:



Scroll back with the following keys:

+



Scroll forward with the following keys:



Exit the INFO function by simultaneously pressing the following keys:



**15.4 Possible error states, Info texts**

MANUAL and AUTOMATIC

<b>PHG text:</b>	<b>Comment:</b>
'Manual disabled '	Manual disabled
'Dis.M4.1,9.1,11.4..6'	Modes 4.1,9.1,11.4,11.5,11.6 disabled
'Progr. BAPS disabled'	BAPS program disabled
'Progr. PIC disabled '	PIC program disabled
'Device/file IO disab'	Device/file I/O disabled
'Progr.start disabled '	Program start disabled
'Restart of RC disabl'	RC restart via PHG disabled
'PHG keys disabled '	Block PHG keys for RC
'Del/mod files disabl'	Program modify and clear functions disabled
'Def/teach in disable'	Define/Teach in disabled
'Refence points disab '	Approach ref. point function disabled
'Disable system mess.'	Disable: approach ref. point message
'Disable online funct'	Disable: Online functions
'Disable coupling PG '	Disable: interfacing to the PG
'Disable printer '	Disable: printer output
'READ/WR. SER 1 disab'	Disable: Read/Write Ser. 1
'READ/WR. SER 2 disab'	Disable: Read/Write Ser. 2
'READ/WR. SER 3 disab.'	Disable: Read/Write Ser. 3
'READ/WR. SER 4 disab'	Disable: Read/Write Ser. 4

**- Mode 1**

**Reference points**

<b>PHG text:</b>	<b>Comment:</b>
'no reference points'	Reference points not yet approached
'No referenc.(P402=0)'	Approach ref. point function inactive (MP 402 = 0)
'emerg.op. without RC'	EMERGENCY MODE EMERGENCY STOP (via error display)
'notINPOS of all axis'	No INPOS
'FEED HOLD '	STOP FEED
'FEED ALLOW n.avail.'	No FEED ENABLE
'CONTRL.ALLOW n.avail'	No control ENABLE
'DRIVE ON not avail.'	No DRIVE ON
'TRAVEL ALLOW n.avail'	No TRAVERSE ENABLE
'kinematics: '	Kinematic(s): 1; 2; 3;...
'1;2;3; '	

<b>- Mode 2</b>	<b>Manual</b>	
	(1)	
	'Kinem.: ROBI_1.....'	Selected kinematics: ROBI_1
	'no reference points'	Reference points not yet approached
	'Coord. not selected'	Coordinate system not yet selected
	'emerg.op. without RC'	EMERGENCY MODE
		EMERGENCY STOP (via error display)
	'notINPOS of all axis.'	Axes not in position
	'FEED HOLD '	STOP FEED
	'FEED ALLOW n.avail.'	No FEED ENABLE
	'CONTRL.ALLOW n.avail'	No control ENABLE
	'DRIVE ON not avail.'	No DRIVE ON
	'TRAVEL ALLOW n.avail'	No TRAVERSE ENABLE
	'kinematics: ' ,	Kinematic(s): 1; 2; 3;...
	'1;2;3; ' ,	

**Note:**

(1) The selected kinematics are displayed here because in the axis display it is not always possible to positively recognize which kinematics are selected at that moment.

<b>- Mode 3</b>	<b>Progr. BAPS/PIC</b>	
	'Prog. BAPS disabled'	BAPS program disabled
	'Progr. PIC disabled '	PIC program disabled
<b>- Mode 3.1</b>	<b>Progr. BAPS-Prog.</b>	
	'Prog. BAPS disabled'	BAPS program disabled
<b>- Mode 3.2</b>	<b>PIC program</b>	
	'Progr. PIC disab. '	PIC program disabled
	'Stop of PIC250 '	PIC 250 stopped
	'PIC checksum error'	PIC checksum error
	'EEPROM write protect.'	PIC write protection active
<b>- Mode 4</b>	<b>Define/Teach in</b>	
	'Def/Teach in disable'	Define/Teach disabled (no PNT file -> already displayed)
<b>- Mode 4.1</b>	<b>Define</b>	
<b>- Mode 4.2</b>	<b>Teach in</b>	
	'No referenc.(P402=0)'	Approach ref. point function inactive (MP 402 = 0)
	'Coord. not selected'	Coordinate system not yet selected
	'emerg.op. without RC'	EMERGENCY MODE
		EMERGENCY STOP (via error display)
	'notINPOS of all axis'	No INPOS
	'FEED HOLD '	STOP FEED
	'FEED ALLOW n.avail.'	No FEED ENABLE
	'CONTRL.ALLOW n.avail'	No CONTROL ENABLE
	'DRIVE ON not avail.'	No DRIVE ON
	'TRAVEL ALLOW n.avail'	No TRAVERSE ENABLE
	'kinematics: ' ,	Kinematic(s): 1; 2; 3;...
	'1;2;3; ' ,	

- Mode 5	<b>Test BAPS progr.</b> 'TEST bit not select.'	Operating mode TEST not set via interface
- Mode 6	<b>PIC monitor</b> 'Stop of PIC250' 'PIC checksum error' 'EEPROM write protect.'	PIC 250 stopped PIC checksum error PIC write protection active
- Mode 6.1	<b>PIC monitor on</b> 'Stop of PIC250'	PIC 250 stopped
- Mode 6.4	<b>Timers and counters</b> 'Del/mod files disab'	Function for changing timers and counters disabled
- Mode 7	<b>Diagnosis</b>	
- Mode 7.1	<b>Axis displays</b> 'notINPOS of all axis'	No INPOS Traverse range limit reached (with fault display)
- Mode 7.3	<b>System conditions</b>	
- Mode 7.3.2	<b>Process conditions</b> 'error in USER-TASK' 'errors and warnings'	Process error Errors/Warnings (similar to Mode 7.2)
- Mode 7.8	<b>Mach. parameters</b>	
- Mode 7.8.2	<b>Set machine parameters</b> 'Del/mod files disabl'	Function for changing mach. parameters disabled
- Mode 9	<b>Device/File I/O</b>	
- Mode 9.1	<b>Default</b> 'Interface: n.avail.' 'Interface: u.double'	No physical interface Double assignment of interface, for ex.: 1
- Mode 10	<b>Select program</b>	
- Mode 10.1	<b>Select processes</b> 'Progr.start disabled'	Program start disabled
- Mode 10.2	<b>Stop processes</b>	
- Mode 11	<b>Help functions</b> 'Restart of RC disabl' 'V/A/D Factor disabled'	RC restart via PHG disabled VFACTOR/AFACTOR/DFACTOR disabled

## 15.5 Output on the PHG

The info texts with the highest priority overwrite other lower-priority texts in the PHG display.

Output priority is as follows:

Priority

- |   |   |
|---|---|
| 1 | Messages from the test system (only if the test system is active) |
| 2 | WRITE PHG from BAPS   |
| 3 | Coded text output   |
| 4 | INFO texts  |
| 5 | Standard operation on the PHG                                     |

**NOTE:**

Because the INFO function has a lower priority than a pending BAPS READ operation, the BAPS READ operation must first be concluded in order to activate the INFO function.



## 16 Coded error output

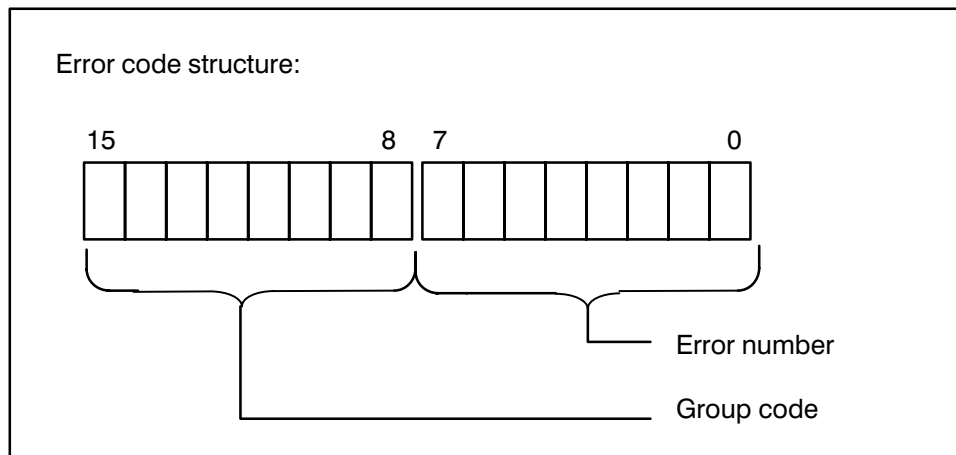
The function "**Coded error output**" issues all errors recognized by the rho 3 operating system to the rho 3 control's digital interface in binary form.

In the following, an error code's structure, the interface assignment and type of errors codes at the interface are described.

An exact description of error codes, the error texts on the PHG, possible causes as well as corrective measures are provided in the Appendix of the rho 3 signal description.

### 16.1 Structuring the errors in groups

Errors which occur are divided into groups of 0 to 14, and up to 256 errors can be coded within each group.



**The groups are divided as follows:**

#### **Group 0 : Process runtime errors**

These error messages are directly related to active BAPS2 processes.

Error code : 1 through 255

#### **Group 1 : CAN errors**

These messages are reserved specifically for control configurations with CAN drive interfaces.

Error code : 256 through 511

#### **Group 2 : Rough interpolation cycle (P2) error, measuring system errors**

This group consists basically of measuring system errors, movement range limits, and EMERGENCY STOP.

Error code : 512 through 767

**Group 3 : Servo error, In Position errors**

This group contains servo errors, interpolator-stop errors, drive-on errors and other errors which can occur when operating the axes.

Error code : 768 through 1023

**Group 4 : Miscellaneous errors**

This group consists of PIC or PLC error messages and other error messages which do not fit in groups 0 through 3.

Error code : 1024 through 1279

**Group 5 : Warnings**

Group 5 consists of coded warnings.

Error code : 1280 through 1535

**Group 6 through 12 :**

These groups are not currently used.

Error code : 1536 through 3327

**Group 13 : Rough interpolator (P2) - runtime errors for rho 3.2**

For rho 3.2 controls with CP2.5 modules, system messages from the control loop end are interpreted as runtime errors.

Error code : 3328 through 3583

**Group 14 : System errors**

This group consists of rho 3 system error messages.

Error code : 3584 through 3839

**Group 15 : System errors reserved**

Error code : 3840 through 4095

## 16.2 Address assignment of the error code signals

Error codes are issued as RC outputs to the addresses (**Error code channel**) specified in the following table.

Symbol.Name	Item. no.	PIC addr.	CL300 address	Byte.Bit	Signal description
STB_COER_RCO	118	I14.6	I12.6	7.7	Strobe coded error output
PTY_COER_RCO	119	I14.7	I12.7	7.8	Parity coded error output
ERROR_0_RCO	152	I19.0	I17.0	12.1	Coded error output bit 0
.					
.					
ERROR_15_RCO	167	I20.7	I18.7	13.7	Coded error output bit 15

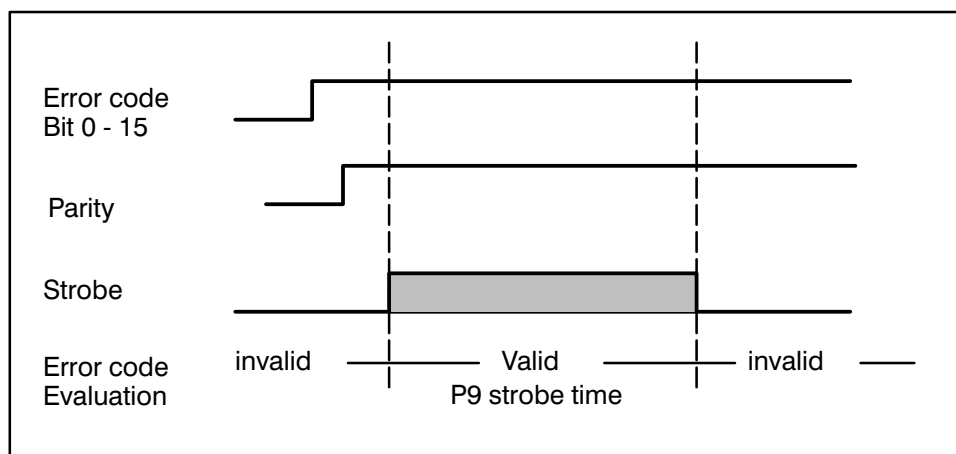
## 16.3 Coded error output of runtime errors

Runtime errors are those errors which are acknowledged after being cleared and reset by Control reset; for example, point variable is undefined. Errors of groups 0 - 13 are runtime errors.

The rho 3 control operating system can store up to 64 runtime errors. These messages, described in the following, are issued via the 16 bit error code channel.

### 16.3.1 Principle of the output of error messages

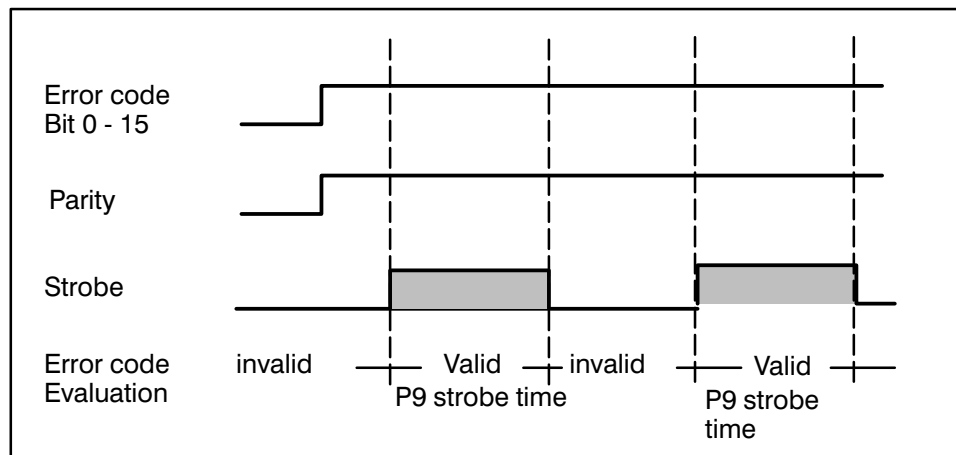
If a runtime error occurs, it is output in coded form at the interface via error code bit 0 through 15 with a strobe signal; this means that the error code can only be evaluated when a strobe signal is pending.



The duration of the strobe signal depends on the strobe time of the system outputs which is set in machine parameter P9.

### 16.3.2 Output of several error codes

If several error messages occur they are output one after the other. The strobe signal is set to '0' after the set strobe time has elapsed. '0' remains then for this strobe time. This is followed by the next error code. The strobe signal is reset to '1' (see illustration).



This sequence continues until the last error code is issued.

### 16.3.3 Parity

The parity bit is generated for even parity. It is generated for all 16 error code bits and is only valid when a strobe signal is pending.

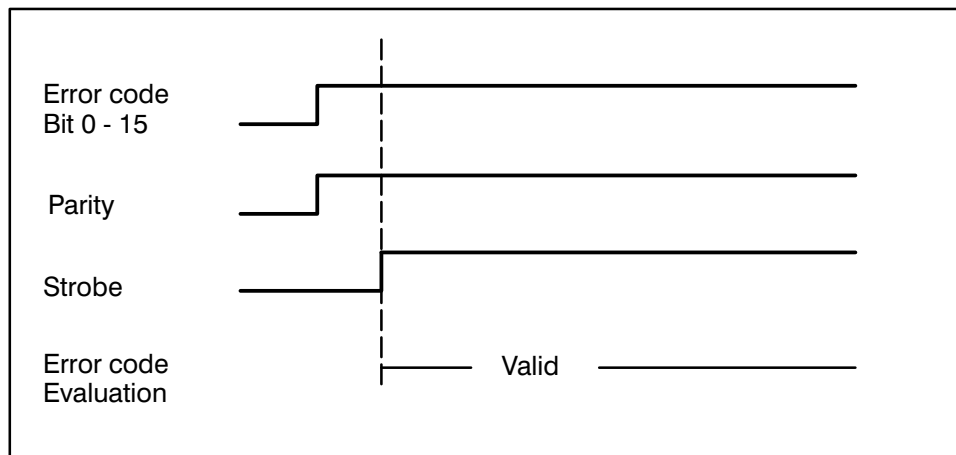
### 16.3.4 Response for successive errors

If an error occurs, for example EMERGENCY STOP, and it is followed by one or more errors, for example Interpolator Stop Warning Axis 1, Servo error Axis 1, then all error codes still pending are re-issued when new errors occur. After their causes have been corrected, errors can be cleared with Control reset at the interface (RC input no. 19, PIC address O2.3), or with Control reset on the PHG (Mode 11.1).

## 16.4 Coded error output of system errors

Starting with software version TO02F, system errors are also issued in coded form at the interface of the rho 3. System errors can only be acknowledged with RC re-start, provided the cause of the error has been corrected.

Unlike coded runtime errors, where the strobe signal is only present with system strobe time (in accordance with the value set in machine parameter **P9**), the strobe signal remains 's t a t i c' after the system error code is issued. This means that it is only cleared after a key on the PHG is pressed.



The parity bit is generated for e v e n parity for all 16 error code bits. It is valid only when a strobe signal is present.

## 17. ROPS3/IQPRO Version W1J

### 17.1. Compatibility between ROPS 3 – rho 3

ROPS 3 /IQpro	Software version		
	control rho 3	PIC250 Program file .QLS for BAPS–PIC .P2O for PROFI	CL300 Program file .P2O for PROFI
W1C	TO01E		
W1F	TO01I	R3_1I_D.QLS	RHO3_1I.P30
W1J	TO02F	R3_2D_D.QLS R3_2D.P2O	RHO3_2D.P30

New versions may be incompatible due to additional functions.

For example:

In version W1F, the function "ORD" was added to the BAPS2 language scope. BAPS2 programs containing this function cannot be compiled or run in older control versions than TO01I. However, alle BAPS2 programs generated with W1C can also be compiled in TO01I or W1F. Please also refer to "Software changes of the current rho 3 software version".

#### 1. Different file format

With the Profi software version 3.0, the P2O file format was changed. The new format is recognized by the code converter (P2O –> P2X) and converted accordingly.

#### 2. Batch files for DOS

As of version W1J, subfunctions of ROPS3 are also available in DOS.

For easier operation in DOS, these functions are included in BAT files. These batch file processing programs are created during installation and stored in the Rops directory. All programs have their own info feature which is invoked with 'file name'/?.

The following functions have been implemented:

All funcnctions for file management of rho3 (coupling),

the BAPS2 compiler,

the BAPS–PIC compiler and

the Rops editor.

Below, please find a list of all BAT files available and their description.

**Info.BAT**

displays a list of all BAT files with a short description.

**RLis.BAT**

displays a list of the rho3 files. The filename and extension may be specified in the call. If no name is entered, the default is \*.\*. The rho3 name must not contain a drive or path name. The function can be cancelled with <Esc>.

Call: RLis rho3Name <RETURN>

**RRen.BAT**

changes the name of a rho3 file.

The batch file needs two parameters: the first one is the old rho3 name, the second is the new file name. The rho3 names must not contain a drive or path name. The file extensions must be identical.

The function can be cancelled with <Esc>.

Call: RRen OldName NewName <RETURN>

**RDel.BAT**

deletes a file in rho3.

The call must include the file name and the extension. The rho3 name must not contain a drive or path name. The function can be cancelled with <Esc>.

Call: RDel rho3Name <RETURN>

**RLoad.BAT**

uploads a file from the PG to the rho3.

The batch file needs two parameters, the first one is the file on the PG, the second is the file of the rho3. If the second parameter is not entered, the rho3 file will have the PG name. The rho3 name must not contain a drive or path name. The file extensions must be identical. If a file with the same name is already available in rho3, the upload process is aborted. The function can be cancelled with <ESC>.

Call: RLoad PGName rho3Name <RETURN>

**RLoadO.BAT**

uploads a file from the PG to the rho3 (replacing an existing file).

The batch file needs two parameters, the first one is the file on the PG, the second is the file of the rho3. If the second parameter is not entered, the rho3 file will have the PG name. The rho3 name must not contain a drive or path name. The file extensions must be identical.

If a file with the same name is already available in rho3, it will be replaced. The function can be cancelled with <ESC>.

Call: RLoadO PGName rho3Name <RETURN>

#### **RStore.BAT**

copies a file from the rho3 to the PG.

The batch file needs two parameters, the first one is the file in the rho3, the second is the file of the PG. If the second parameter is not entered, the PG file will have the rho3 name. The file extensions must be identical. If a file with the same name is already available on the PG, the copying process is aborted. The function can be cancelled with <ESC>.

Call: RStore rho3Name PGName <RETURN>

#### **RStorO.BAT**

copies a file from the rho3 to the PG (replacing an existing file).

The batch file needs two parameters, the first one is the file in the rho3, the second is the file of the PG. If the second parameter is not entered, the PG file will have the rho3 name. The file extensions must be identical. If a file with the same name is already available on the PG, it will be replaced. The function can be cancelled with <ESC>.

Call: RStorO rho3Name PGName <RETURN>

#### **Comp.BAT**

compiles a QLL file.

The call must include the file name of a QLL file, without drive, path or extension.

Call: Comp QLLfile <RETURN>

#### **PComp.BAT**

compiles a QLS file and generates the Intel hex format. The call must include the file name of a QLS file, without drive, path or extension.

Call: PComp QLSfile <RETURN>

#### **RSet.BAT**

offers the possibility to set the interface parameters of the coupling. The operation is menu-guided and supported by an info function. The function is terminated with <ESC>.

Call: RSet <RETURN>

#### **Edi.BAT**

calls up the ROPS3 editor.



The call may include up to two file names. If two names are entered, the files are edited in different windows.

Call: Edi File1 [File2] <RETURN>

### 3. Coupling and 386 mode

The WINDOWS 386 mode may produce problems with the coupling functions (rho3/IQ140):

The active coupling is displayed as an icon, magnification to a full-size screen display may cause a "timeout".

If the coupling is called up several times, a system error may occur. In order to avoid this, you should start Windows in standard mode with "WIN /S".

### 4. Keyboard operation of the DOS modules

The following modules can only be accessed with the keyboard:

– Setup coupling/printer/colors

– Editor of the group "ROPS 3"

<ESC> Exit

<F10> Help

<TAB> Select next group

<CURSOR-UP/DOWN> Select next option

### 5. Mouse connected to serial port

A serial mouse should be connected to COM1, and the coupling with the control to COM2. A configuration the other way round may cause problems.

Error message: "Key not found 152"

### 6. 386 mode

If several programs are started simultaneously in the programming system in the 386 mode of Windows (e.g. Editor of the "ROPS 3" group and BAPS compiler), the following error message will be displayed:

"Sharing violation on drive ..." or

"Error 2 opening ..."

**Remedy:**

– Close one program before calling up the next one, or

– Start Windows in the standard mode with "WIN /S".

### 7. "CAPS LOCK" key

In several older programming units (PG4), pressing the "CAPS LOCK" key several times may initiate a warm start of the unit.

### 8. Floppy disk drive

If Windows accesses the floppy disk drive, although no disk was inserted in the drive, and if "Cancel" does not work, further operation is possible by inserting a floppy into this drive.

9. File manager – Formatting

When the error occurs:

<Formatting error.

The current drive cannot be formatted>

select another drive and repeat the formatting process.

10. Window in 386 mode

For the following functions, the window for the user interface may be too small in 386 mode:

- ROPS 3 – Editor
- Setup functions
- Coupling – copying with wildcards

**Remedy:**

- Change the window size accordingly,  
or select "Full Screen".

11. Changing floppies

A floppy change is not automatically recognized if a dialog box is open (e.g. after "clicking on" Editor). In order to update the contents of the file list, "click" again on the drive symbol. (The same applies to the PC data window in coupling).

12. Multiple selection of "Print/Write"

Calling this function while another instance is running will cause a device conflict.

**Remedy:**

Please wait until the function has been completed.

13. Copying with wildcards

If no file is found that meets the wildcard specification when copying with wildcards (PC -> control/control -> PC), no message is displayed that no file was found.

14. PROFI software support

14.1. Loading and saving with the PROFI software

Two batch files were created for loading or saving the PIC250 program with the PROFI software.

The batch files must be installed as applications in the PROFI user interface. The batch files with the names "LOAD\_PIC.BAT" and "SAVE\_PIC.BAT" are stored in the [ROPS3] directory. Below, please find a description of how to install "LOAD\_PIC.BAT". The installation of "SAVE\_PIC.BAT" is performed analogously.

As of the Profi software version 3.0, the P20 file format was changed, this new format is recognized by "LOAD\_PIC.BAT" and converted accordingly.

Using the PROFI software, help can be requested at any time with the <F10> key.

The batch file is installed as follows:

- Call up PROFI

- In the main menu, press <F7> to select the "Application" submenu.
- Press <F2> to select the function "Change".
- Up to nine programs can be installed with the submenu. A free program location can be selected with the corresponding function key.
- Then you are expected to enter a title, e.g. "Load rho3".

Finish entry with <Enter>.

- Then enter a program name.

For batch files:

C:\COMMAND.COM      <Enter>

- The next entry refers to the parameter.

Entry: /c C:\ROPS3\LOAD\_PIC.BAT %1 %2 <Enter>

- Last entry: Activate pause, operation with <A> key.  
Finish your entry with <ENTER>.  
The installation is complete.

Operation, starting with the main menu:

- Select Application <F7>
- Select Run <F1>
- Function "Load rho3"
- "Parameter %1" DOS file name (without extension) <Enter>

For DOS file names a path may be entered. If no directory is entered, the current path is the default. Specifying an extension will abort the batch file.

- "Parameter %2" rho file name (without extension) <Enter>  
Path names are not allowed on rho!

The batch procedure is invoked and executed, the result is displayed on the screen. Return to PROFIL with <Esc>.

**Notes:**

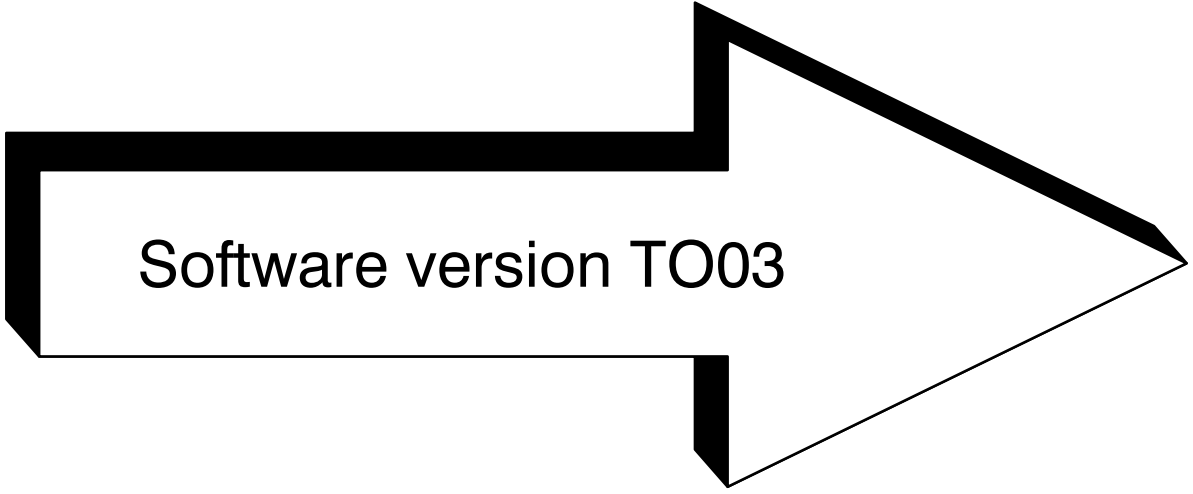
**For the function "Save rho3", the parameters %1 (DOS file name) and %2 (rho file name) are exchanged according to the direction of the transfer, i.e. "Parameter %1" is the rho file name, "Parameter %2" is the DOS file name.**

14.2. Standard PIC program for PROFIL software

With the new ROPS3 software, a standard PLC program is available for PIC250 as an instruction list.

The program fits to rho3 as of version TO02F. The program with the name R3\_2D.P2O is stored in the [ROPS3] directory. The related symbol file has the name R3\_2D.S2S.





Software version TO03

## Software version TO03

<b>1</b>	<b>General notes on software version TO03</b>	
1.1	System process priority .....	3 – 1
1.2	Bit coupler with CAN drive interface .....	3 – 1
1.3	25MHz CPU module .....	3 – 1
1.4	Intelligent modular servo board .....	3 – 1
<b>2</b>	<b>Direct function selection with the PHG .....</b>	<b>3 – 2</b>
2.2	Restrictions, handling errors .....	3 – 5
<b>3</b>	<b>Displaying BAPS2 source texts in the TEST mode</b>	
3.1	Writing a program line (Mode = 5.11) .....	3 – 6
3.2	Displaying BAPS2 source text in single-step mode (Mode = 5.9) .....	3 – 7
<b>4</b>	<b>New or modified machine parameters</b>	
4.1	Modified machine parameters .....	3 – 8
4.1.1	Previous meanings of the machine parameters .....	3 – 8
4.1.2	Modified meanings of the machine parameters .....	3 – 8
4.2	New machine parameters .....	3 – 8
4.3	Compatibility of machine parameters .....	3 – 9
4.4	Parameter input for singleturn absolute encoder .....	3 – 9
<b>5</b>	<b>Servodyn-G(C) software limit switch</b>	
5.1	Function .....	3 – 10
5.2	Setting rule .....	3 – 10
<b>6</b>	<b>Referencing with Servodyn-G(C) drive systems</b>	
6.1	Setting referencing modes .....	3 – 12
6.2	REF- MODE 0 ("Normal") .....	3 – 13
6.2.1	Function .....	3 – 13
6.2.2	Setting the reference point switch .....	3 – 13
6.3	REF- MODE 1 ("Correct orientation") .....	3 – 14
6.3.1	Function .....	3 – 14
6.3.2	Setting the reference point switch .....	3 – 15
6.4	REF- MODE 2 ("Resolver suitable") .....	3 – 16
6.4.1	Function .....	3 – 16
6.4.2	Setting the reference point switch .....	3 – 16
6.5	REF- MODE 3 (Combination of 1 and 2) .....	3 – 17
6.5.1	Function .....	3 – 17
6.5.2	Setting the reference point switch .....	3 – 17
<b>7</b>	<b>Reversing the direction of rotation for the Servodyn-G(C) .</b>	<b>3 – 18</b>
<b>8</b>	<b>Configuration recognition for the PLC program</b>	
8.1	Description .....	3 – 19

<b>9</b>	<b>Interface signal error acknowledgement</b>	
9.1	Description of function .....	3 – 21
9.2	RC signals .....	3 – 21
9.3	Restrictions, handling errors .....	3 – 21
<b>10</b>	<b>Counter and PIC timer expansion .....</b>	<b>3 – 22</b>
10.1	Flag assignment of PIC250 timers and counters .....	3 – 23
<b>11</b>	<b>Special function 3, Setting machine position</b>	
11.1	Declaration .....	3 – 24
11.2	How the function for setting machine position works .....	3 – 25
<b>12</b>	<b>Expansion of special functions 1 (IOL) and 2 (PPO) .....</b>	<b>3 – 27</b>
12.1	Response to interface signals and system states .....	3 – 28
<b>13</b>	<b>BAPS2 - Language expansion</b>	
13.1	CASE – instruction .....	3 – 29
13.2	STANDARD - CONSTANTS .....	3 – 31
13.2.1	Standard - Constant CLS (clear screen) .....	3 – 31
13.2.2	Standard - Constant VERSION .....	3 – 31
13.3	Constant declarations .....	3 – 31
<b>14.</b>	<b>ROPS3/IQPRO Version W1L</b>	
14.1.	Compatibility with ROPS3 .....	3 – 32

## 1 General notes on software version TO03

The following functions and options are available for the rho 3 control configurations with the new software version.

This new software version is compatible with version TO02. When converting from a TO01 software version to TO03, please also read the Notes in part 2 of this documentation.

### 1.1 System process priority

In the new software version, the priority of the BAPS2 compiler in the control is set to 140. If user processes with higher priority (priority value <140) are active in the control, this may cause extended compilation time.

### 1.2 Bit coupler with CAN drive interface

As of the software version TO03, it is possible to use a bit coupler with a CAN drive interface to the Bosch Servodyn-G for 6 axes.

### 1.3 25MHz CPU module

This new software version supports use of the 25MHz CPU module. The software automatically recognizes the processor module being used.

This module can only be used in rho 3.1 configurations on a CP/MEM4 module, as well as on the following intelligent servo boards:

SERVO 3i,  
SERVO 4i and SERVO 6i.

The 25MHz CPU module increases performance by approx. 20% compared to the standard 15MHz CPU module.

**Note:**

**The 25MHz CPU module cannot be used on the SERVO m.i. and not in the rho 3.2 configuration on the CP 2.5 CPU board.**

### 1.4 Intelligent modular servo board

The new intelligent modular servo board (SERVO m.i.), which can be modularly fitted with absolute, incremental, and potentiometer measuring systems, is supported starting with this software version. The interface conditions of the new board are described in the manual **Bosch rho 3 interface conditions, 1070 073 024**.



## 2 Direct function selection with the PHG

Previously, selecting a specific sub-mode in the PHG operator interface required scrolling through the various levels by selecting the corresponding modes one by one. This was also necessary to return to the basic level from a sub-level which was done by pressing CURSOR LEFT several times.

### Example: previous selection of a function

**MODE**

**3**

Enter the menu level

**Enter**

After < ENTER >, the control shows the selected level

**4**

Enter the sub-level and confirm with <ENTER>

**Enter**

Press until the desired sub-level is reached








To make it easier for the operator to select sub-levels and to return to the basic level, the operating system of the rho 3 has been improved with a function which allows direct selection of sub-levels as well as returning directly to the basic level.

**2.1 Direct selection**






If the operator wants to go from one operating level to a deeper one, he does so by entering all the intermediate levels, including the destination level. The individual levels or sub-levels are separated by periods (".").

**Example:**

If an operator wants to look at the "Process conditions", he can reach the corresponding PHG function by entering "**Mode=7.3.2**".

	
	Enter the menu level
	Separate sub-levels with < . >
	Enter the sub-level and
	Separate sub-levels with < . >
	Enter the sub-level and
	Confirm with <ENTER>

The procedure is the same for going from Mode 3.2 "Programming PIC programs" to the sub-mode "Displaying timers and counters" of the same hierarchical branch. In this case, enter "**Mode=2.1**".

	
	Enter the sub-level and
	Separate sub-levels with < . >
	Enter the sub-level and
	Confirm with <ENTER>

### 2.1.1 Direct return

Enter "Mode=0" to return to the basic level from any sub-mode. From there, branching is possible to any sub-menu.

**MODE**

**0**

Enter 0 for the basic level and

**Enter**

Confirm with <ENTER>

### 2.1.2 Directly changing operating levels

Enter a leading "0" before the desired function to change, for instance, from "Diagnosis" to "Auxiliary functions".

**Example:**

"Help functions" of the sub-mode "VFACTOR" is selected via the basic level by entering "MODE=0.11.4" at any level.

**MODE**

**0**

Enter the menu basic level

**▪**

Separate sub-levels with < . >

**11**

Enter the main level and

**▪**

Separate with < . >

**4**

Enter the sub-level and

**Enter**

Confirm with <ENTER>

## 2.2 Restrictions, handling errors

Jumping "crosswise" across the hierarchy branch of the PHG tree can only be done via the basic level.

Checking for a permissible sub-mode only happens after all the previous levels have been gone through. The only exception is the check for permissible ASCII characters ("0".."9" and "."). In case of an error, the direct selection function gets "stuck" at the level in which an invalid sub-mode was selected. This means that if "Mode=9.4.10" is entered, the direct selection function is aborted in Mode 9.4 as Mode 9.4.10 does not exist. In this case, there is no error message. The user can then select one of the valid sub-modes, in this case 1, 2, or 3.

If it is necessary to select a file name while proceeding to a sub-mode, the direct selection function continues to run until and including the selection of the file.

Selecting individual functions on the PHG operation tree remains possible by entering the corresponding mode. It is also possible to return from level to level with CURSOR LEFT.

### 3 Displaying BAPS2 source texts in the TEST mode

In the new software version, BAPS2 source text lines which are longer than the PHG display can be completely read by scrolling the line.

#### 3.1 Writing a program line (Mode = 5.11)

This command can be used to display QLL program lines on the PHG. After the command is selected, the test system prompts the operator to enter the line number he wants to see.

**Display on the PHG:**

```
dana xxxxx  
Write  
  
Line :#
```

xxxxx : Next program line

# : Cursor for the expected input

**Operation possibilities:**

Program line number entry (max. 32767; xxxxx is the default)

 Enter

Command execution

 Shift + 

Selection of new test system  
There is no write operation


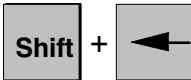





After entering the desired line number, the contents of the program line are displayed on the PHG.

**Display on the PHG:**

```
dana xxxxx  
Write  
Program line
```

xxxxx : Next program line

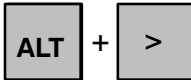

**Operation possibilities:**

		Conclude display, enter new test system command
		
		Scrolling within a line
		Scrolls the program line horizontally to the right
		Scrolls the program line horizontally to the left
		
		Scrolls within the QLL program Repeat function by holding the keys down

When program lines are displayed, the spaces between line numbers and the first character are masked out.

### 3.2 Displaying BAPS2 source text in single-step mode (Mode = 5.9)

It is possible to scroll horizontally within a program line with the following key combinations.

		Scrolls the program line horizontally to the right
		Scrolls the program line horizontally to the left

Thus, program lines longer than the PHG display can still be completely read.

**Note:**

Further information on the BAPS2 test function is provided in the document "rho 3 PHG operation", P. No. 4264 edition 7/93.

## 4 New or modified machine parameters

In addition to the introduction of new machine parameters, the new software version has changed the effects of certain machine parameters.

### 4.1 Modified machine parameters

#### 4.1.1 Previous meanings of the machine parameters

Up to and including the software version TO02, the machine parameters

P104 – Slope acceleration PTP and JOG in joint coordinates,

P105 – Slope point PTP and JOG in joint coordinates, and

P122 – Acceleration/delay variation times PTP and JOG in joint coordinates

apply in Automatic mode for PTP interpolation as well as in Manual mode in the JOG mode in joint coordinates.

The machine parameter P104 defines the Up slope ramp and the Down slope ramp for JOG in joint coordinates.

Parameter P117 – slope acceleration JOG in world coordinates - applies to the Up slope ramp as well as to the Down slope ramp.

#### 4.1.2 Modified meanings of the machine parameters

The following machine parameters act for the corresponding kinematics only in automatic mode (PTP):

P104 – Slope acceleration PTP

P105 – Slope point PTP

P122 – Acceleration/delay change times PTP

Otherwise their meanings are as before.

Machine parameter P117 (Up and Down slope acceleration JOG in world coordinates) has been expanded by the Down slope value. The value for the Down slope is requested at parameter input.

### 4.2 New machine parameters

The following machine parameters must be re-entered beginning with software version TO03:

P128 – Up and Down slope acceleration JOG in joint coordinates

P129 – Slope point JOG in joint coordinates (applies to Up and Down slope ramp)

P130 – Acceleration/deceleration alteration time for JOG in JC

The same machine parameter description applies to the parameters

**P128,**

**P129,**

**P130** (JOG in joint coordinates)

as that for parameters P104, P105, and P122 (PTP).

### 4.3 Compatibility of machine parameters

The machine parameters of versions TO01 and TO02 can also be loaded in version TO03.

If the new parameters P128, P129 and P130 are not occupied, it is assumed that the same values apply for the slope values of these parameters as for parameters P104, P105 and P122. The same applies to the expanded parameter P117.

The desired values for the new parameter should be set.

**Note:**

A detailed description of all possible parameters is provided in the document "rho 3 Machine parameter descriptions", P. No. 4262 edition 5/93.

### 4.4 Parameter input for singleturn absolute encoder

in the multiturn format (T + R encoder)

For absolute encoders, the number of rotations and the pulse/rotation ratio must be entered by means of P401. When 1 is entered for the number of rotations, the data format is automatically switched from 24 bits to 12 bits (corresponds to Stegmann encoder).

For encoders from other manufacturers (for example, T + R), 24 bits are transferred even if the encoder is singleturn. To make it possible to connect these encoders to the rho 3, parameter P401 has been expanded as follows:

Entry	:	1	Singleturn encoder in "Stegmann format" (Number of data bits = 12)
Entry	:	-1	Singleturn encoder in multiturn format (T+R) (Number of data bits = 24)



## 5 Servodyn-G(C) software limit switch

### 5.1 Function

Limit switch monitoring in the SM..-GC (Servodyn-GC) is activated by means of a control bit (ENABLE LIMIT SWITCH) transferred via the digital CAN interface. This simultaneously establishes the zero position in the SM..-GC.

The ENABLE LIMIT SWITCH bit is transferred by the rho 3 at the conclusion of the referencing cycle (including reference point offset). The SM..-GC establishes the resolver's next zero crossing as the zero position. This means that the set limit values are based on this position.

If limit switch monitoring has been activated on the SM..-GC (RMC), it remains active until the SM..-GC (RMC) is switched off. Switching the rho 3 on or off or rho 3 start-up do not reset the SM..-GC (RMC) limit switch logic because only the first 0/1 change of the ENABLE LIMIT SWITCH bit is evaluated by the SM..-GC (RMC). On the other hand, monitoring must be reactivated after switching the SM..-GC (RMC) on or off by referencing again.

### 5.2 Setting rule

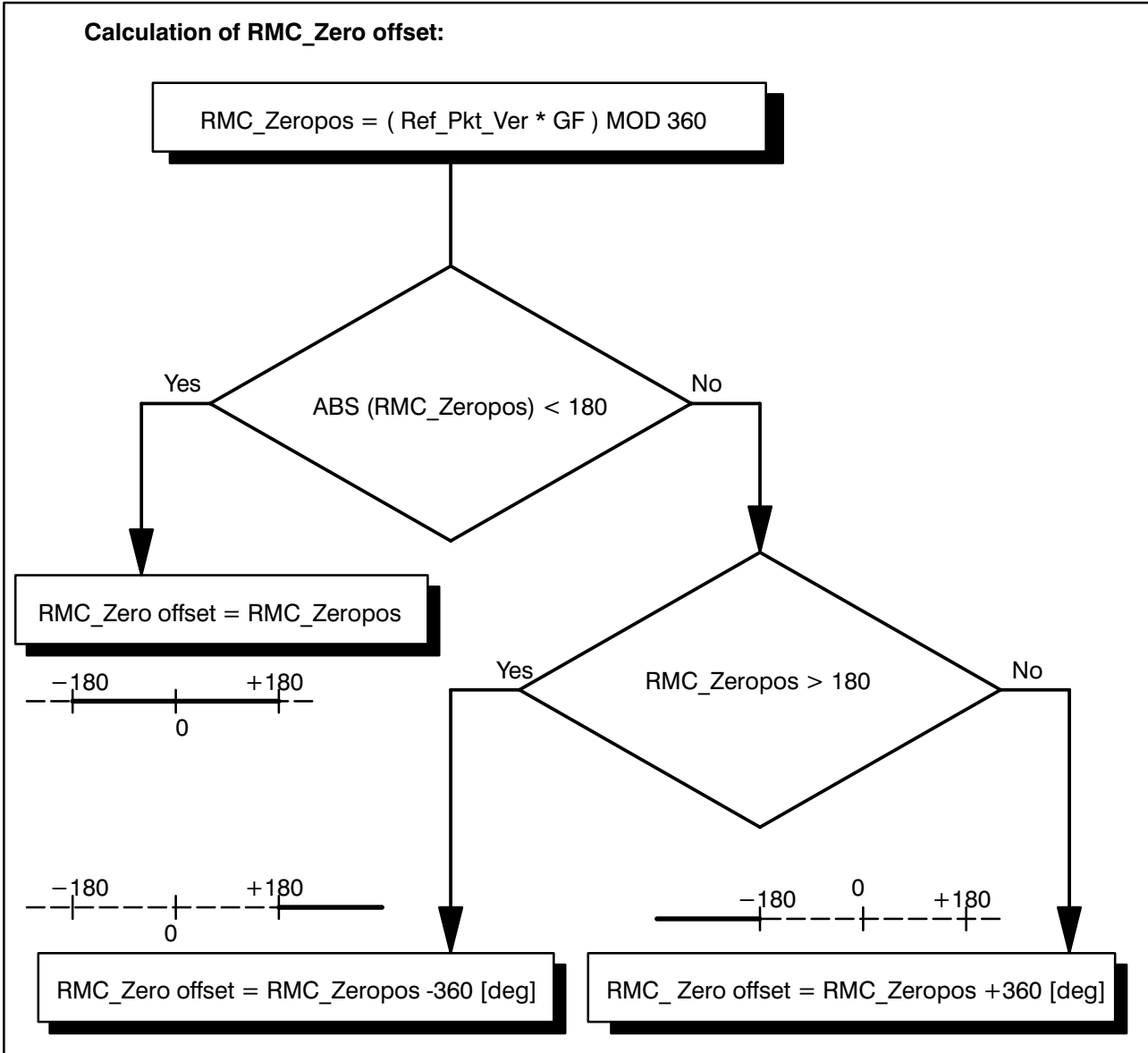
The software limit switches RMC-limit (pos/neg) must be set on the drive booster for each axis in accordance with the following computational rule.

$$\begin{aligned}
 p\_RMC\_L &= ((POS\_SW\_ENDS - Ref\_Pkt\_pos) * GF + RMC\_Zero\ offset) * CPS/360 \\
 n\_RMC\_L &= ((NEG\_SW\_ENDS + Ref\_Pkt\_pos) * GF - RMC\_Zero\ offset) * CPS/360 \\
 Ref\_Pkt\_pos &= Ref\_Pkt\_act (P207) + Ref\_Pkt\_offset (P208)
 \end{aligned}$$

#### Explanation to the abbreviations :

<b>n_RMC_L</b>	:	Negative software limit switch in [inc]
<b>p_RMC_L</b>	:	Positive software limit switch in [inc]
<b>Ref_Pkt_pos</b>	:	Axis position at the conclusion of the referencing operation
<b>Ref_Pkt_act</b>	:	Reference point actual value, corresponds to machine parameter P207
<b>Ref_Pkt_offset:</b>	:	Reference point offset, corresponds to machine parameter P208
<b>CPS</b>	:	<b>CAN Position Scaling</b>
<b>RMC_Zero offset [grd]</b>	:	Distance between ref.point end position and the next resolver zero crossing in [degrees] at resolver

**GF** : Gear factor for rotary axes without dimension, for linear axes [mm/degrees]



The RMC\_Zero offset should be between -90 [degrees] and +90 [degrees]. If this is not the case, the reference point offset must be changed accordingly. In order not to change the end position after referencing, the set home position Offset can be corrected by 90 [degrees]; see also the appropriate documentation on the drive booster.

## 6 Referencing with Servodyn-G(C) drive systems

When using the rho 3 control in conjunction with the Bosch **Servodyn-GC** and the digital CAN drive interface, the referencing modes described in the following can be set.

### 6.1 Setting referencing modes

The desired type of referencing is selected with machine parameter P401.

**Example:**

Axis 1 with CAN interface

```
ROBI_1           A_1
SERVO BOARD NO.: 1
CONNECTOR NO.:   X51           ;CAN module on I/O board
MODULE NO.:      1
INPUT ON MODULE: 1
REF. MODE:       0           ;"Normal" referencing
                  1           ;Referencing
                  2           ;"Correct orientation"
                  3           ;Referencing without approach
                              ching the zero crossing
                              ;Referencing
                              ;"Correct orientation"
                              ;without approaching the zero
                              ;crossing

PULSE/ROTATION:   CPS
MS EVALUATION (NOMINAL VALUE): Degrees or mm/motor rotation * CPS
SET VALUE OUTPUT NO.: 1
```

## 6.2 REF.- MODE 0 ("Normal")

### 6.2.1 Function

Referencing is initiated with the PHG in Manual mode or in the BAPS program (program INIT). The axis moves toward the reference point until the reference point switch is recognized. When the switch is recognized, the axis continues to move in the same direction until the next encoder zero crossing. When this position is reached, the reference point actual value (P207) is used as the actual axis position. If a reference point offset (P208) was entered, this point is moved to and the referencing cycle is thus concluded.

### 6.2.2 Setting the reference point switch

The reference point switch must be adjusted during "normal" referencing so that it is as far away as possible from the zero crossing.

**Procedure:**

- Move the axis precisely to the edge of the reference point cam.
- Start the rho 3.
- Read the axis position on the PHG.
- Set home position offset on the RMC.

Home position offset =  $(180 \text{ [degrees]} + (\text{CDR} * \text{VZ}[\text{MBewert}] * \text{PHG-Wert} * \text{GF})) \text{ MOD } 360$

GF : Gear factor

VZ (MBewert) : Sign of the measuring system evaluation (P401)

CDR : **CAN DIRECTION** of **R**otation (MOOG Parameter OD)  
+1 if positive  
– 1 if negative

## 6.3 REF- MODE 1 ("Correct orientation")

This reference mode must be used for rotary axes capable of several rotations and for which there is no measuring gear box between the axis and the reference point cam. This prevents overrotation of the axes and thus damage to the supply lines.

### 6.3.1 Function

Referencing is initiated with the PHG in Manual mode or in the BAPS program (program INIT) with the instruction REF\_PNT. The axis moves toward the reference point until the reference point switch is recognized. When the switch is recognized, the zero position of the axis is calculated based on the position value which has been read and the gear factor. This position is approached in the next step. When this position is reached, the reference point actual value (P207) is used as the actual axis position.

If a reference point offset (P208) was entered, this point is moved to and the referencing cycle is thus concluded.

For correct orientation referencing, axis rotation is derived from the absolute position of the resolver at the reference point. Here, resolver rotation is divided into  $n$  segments, whereby  $n = 1/\text{after-decimal value of the gear factor}$ .

#### **Example:**

For the 4th axis of the SR60 with a gear ratio of 4.1:1, the value  $n = 10$  is obtained. This means that one resolver rotation is divided into 10 36-degree segments. The rho 3 internal position value for a segment is between -18 degrees and +18 degrees so that the zero point is in the middle of a segment. At a rotation movement of 36 degrees on the resolver the gear ratio of 4.1:1 yields an axis movement of only 8.78 degrees. If the edge of the reference point cam is near a segment transition, different reference point positions can be calculated because there is no guarantee that the cam is always recognized in the same segment. For this reason, it is essential to set the resolver zero point carefully.

### 6.3.2 Setting the reference point switch

The reference point switch must be adjusted during "correct orientation" referencing so that it is as near as possible to the zero crossing.

**Procedure:**

- Manually move the axis to the approximate middle position (cable must be slack) so that it is precisely at the edge of the reference point cam.
- Start the rho 3.
- Read the axis position on the PHG.
- Set the home position offset on the Servodyn-GC booster.

Home position offset = (360 [degrees] + (VZ[MBewert] \* PHG-Wert \* GF)) MOD 360

GF : Gear factor

VZ (MBewert) : Sign of the measuring system evaluation (P401)

This setting ensures that the absolute zero point of the axis is near the middle position. At the same time, it also ensures that the edge of the reference point cam is not near a resolver segment transition.

## 6.4 REF.- MODE 2 ("Resolver suitable")

This referencing mode must be used whenever the traversing range of the machine after recognition of the reference point switch is not sufficient for reaching the next encoder zero point.

### 6.4.1 Function

Referencing is initiated with the PHG in Manual mode or in the BAPS program (program INIT). The axis moves toward the reference point until the reference point switch is recognized. Then the encoder position is read and the axis is stopped with the set delay ramp. The read encoder position is calculated together with the reference point actual value (P207) and then used as the actual axis position. If a reference point offset (P208) was entered, this point is moved to and the referencing cycle is thus concluded.

In this referencing mode, the zero point of encoder is not approached.

The parameters P207 (reference point actual value) and P208 (reference point offset) refer, as before, to the next encoder zero point.

The parameters P108 and P109 (referencing speed) retain their meanings. Ensure that P109 (1st reduced speed) is set so that axis braking between the reference point switch and the machine limit switch is possible.

P110 (2nd reduced speed) has no meaning for referencing without approaching the encoder zero point.

### 6.4.2 Setting the reference point switch

The reference point switch must be adjusted during "resolver suitable" referencing so that it is as near as possible to the zero crossing.

**Procedure:**

- Move the axis precisely to the edge of the reference point cam.
- Start the rho 3.
- Read the axis position on the PHG.
- Set the home position offset on the Servodyn-GC booster.

Home position offset =  $(360 \text{ [degrees]} + (\text{CDR} * \text{VZ}[\text{MBewert}] * \text{PHG-Wert} * \text{GF})) \text{ MOD } 360$

GF : Gear factor

VZ (MBewert) : Sign of the measuring system evaluation (P401)

CDR : **CAN DIRECTION** of **R**otation (MOOG Parameter OD)

+1 : if positive  
– 1 : if negative

## **6.5 REF.- MODE 3 (Combination of 1 and 2)**

### **6.5.1 Function**

Referencing is initiated with the PHG in Manual mode or in the BAPS program (program INIT). The axis moves toward the reference point until the reference point switch is recognized. When the switch is recognized the zero position is calculated as described under point 6.3.1 and the axis is stopped at the set delay ramp. The calculated zero position is calculated together with the reference point actual value (P207) and then used as the actual axis position. If a reference point offset (P208) was entered, this point is moved to and the referencing cycle is thus concluded.

### **6.5.2 Setting the reference point switch**

The reference point switch is set as described under point 6.3.2.



## 7 Reversing the direction of rotation for the Servodyn-G(C)

To reverse the direction of rotation of a Servodyn-G(C) (MOOG–RMC) by means of machine parameters, the measuring system evaluation (P401.7) as well as the setpoint output (P401.8) of the corresponding axis must be entered as negative values.

**Example:**

The direction of rotation of the 2nd axis is to be reversed.

o l d machine parameter P401 (axis 2):

A_2 Servo-B.	:	1
A_2 CAN plug no.	:	X51
A_2 CAN module no.	:	1
A_2 CAN input	:	2
A_2 Ref. mode	:	0
A_2 Pulse/rotation	:	16384
A_2 MBewert	:	16384.0
A_2 Setpoint output	:	2

n e w machine parameter P401 (axis 2):

A_2 Servo-B.	:	1	
A_2 CAN plug no.	:	X51	
A_2 CAN module no.	:	1	
A_2 CAN input	:	2	
A_2 Ref. mode	:	0	
A_2 Pulse/rotation	:	16384	
A_2 MBewert	:	<b>-16384.0</b>	<-- new value
A_2 Setpoint output	:	<b>-2</b>	<-- new value

**Note:**

**If only one of the two values is "reversed", the axis will not reach the position during movements.**

## 8 Configuration recognition for the PLC program

The configuration recognition feature allows the user to use the same PLC program for different machine configurations.

### 8.1 Description

Starting with version TO03B, the configuration recognition "Machine configuration" is set with machine parameter P2.

A value between 0..15 can be entered for this parameter; the default value is 0.

The value in P2 is output as a 4-bit code at the RC internal interface and can be used in the PLC program for program flow control.

#### RC output signal assignment:

RC outputs:						
BAPS name	Bit no.	PIC addr.	PC600 addr.	CL300 addr.	RC output	Signal description
MA_TYP_0_RCO	124	I15.4	I15.4	I13.4	8.5	Bit 0 significance 1
MA_TYP_1_RCO	125	I15.5	I15.5	I13.5	8.6	Bit 1 significance 2
MA_TYP_2_RCO	126	I15.6	I15.6	I13.6	8.7	Bit 2 significance 4
MA_TYP_3_RCO	127	I15.7	I15.7	I13.7	8.8	Bit 3 significance 8

#### Example:

A PLC program is to be created which can be used for two rho 3 controls with different machine configurations.

Control 1: Machine configuration : one kinematic with 3 axes

Control 2: Machine configuration : two kinematics with 4 axes each

"0" (default) is preset in parameter P2 of the 1st control for the machine configuration; "2" in the P2 of the 2nd control.

Excerpts from the corresponding sections of the PLC program:

```

AN      -MA_TYP_0           ;Decoding of the machine configuration
AN      -MA_TYP_1           ;output by the control.
AN      -MA_TYP_2
AN      -MA_TYP_3
=       -AEIN_42           ;AEIN_42 = Control 1 (P2 = 0)

AN      -MA_TYP_0
A       -MA_TYP_1
AN      -MA_TYP_2
AN      -MA_TYP_3
=       -AEIN_43           ;AEIN_43 = Control 2 (P2 = 2)

AN      -M1_END4_P         ;Limit switch 4th axis + control 2, kin. 1
A       -AEIN_43
=       -AEIN_7

AN      -M1_END4_N         ;Limit switch 4th axis + control 2, kin. 1
A       -AEIN_43
=       -AEIN_8

AN      -M2_END4_P         ;Limit switch 4th axis + control 2, kin. 2
A       -AEIN_43
=       -AEIN_15

AN      -M2_END4_N         ;Limit switch 4th axis + control 2, kin. 2
A       -AEIN_43
=       -AEIN_16

A       -AEIN_42           ;Jump to jog key control 1
JC      -JOG_ST1

;       Jog key on the PHG control 2

A       -ANGRUP_1
A       -JOG1MPHG
AN      -JOG1PPHG
=       -JOG_1_M           ;Jog, 1st axis neg. at RC

.
.
.
A       -ANGRUP_2
A       -JOG4PPHG
AN      -JOG4MPHG
=       -JOG_8_P           ;Jog, 8th axis pos. at RC

.
.
.
A       -LOGISCH1
JC      -BREMSFR

;       Jog key on the PHG control 1

        -JOG_ST1

A       -ANGRUP_1
A       -JOG1MPHG
AN      -JOG1PPHG
=       -JOG_1_M           ;Jog, 1st axis neg. at RC

.
.
.
A       -ANGRUP_1
A       -JOG3PPHG
AN      -JOG3MPHG
=       -JOG_3_P           ;Jog, 3rd axis pos. at RC

;       Time for brake enable M1

        -BREMSFR

A       -SVE_BETN
AN      -M1_ANTRB
R       -FAHRVER1

.
.

```

## 9 Interface signal error acknowledgement

The new interface signal allows errors to be acknowledged without the signal "Control reset". In this way, errors can now be reset without having to stop active processes.

### 9.1 Description of function

However, errors can continue to be cleared with "Control reset", provided their cause has been corrected. In addition, error messages can be cleared by means of the RC input "Error acknowledgement". Active processes are not stopped when these functions are used.

### 9.2 RC signals

Clearing or acknowledging error messages can be initiated with the RC input

**"Error acknowledgement"                      BAPS-PIC No. 62                      PIC addr. 07.6.**

The signal is evaluated as a strobe signal which means that a signal change from "0" to "1" resets the internal error buffer. The displayed runtime errors, warnings, and other errors are cleared

### 9.3 Restrictions, handling errors

For safety reasons, EMERGENCY STOP cannot be cleared after the conditions which lead to emergency status have been rectified. This still requires the following:

**"Control reset",                                      RC input 19,                                      PIC address. 02.3**

In addition, it is not possible to reset P2 errors, for example servo errors, with the signal "Error acknowledgement".

## 10 Counter and PIC timer expansion

In the new rho3 software version TO03 , the number of PIC timers and counters available in the stages 8, 16, 24 or 32 can be set with an option byte.

Timers and counters are selected by means of predefined flag areas in the PIC250; the PIC program must be modified accordingly to use the timers and counters.

Timer and counter values are assigned with machine parameters **P13** and **P14**.

The timers and counters are implemented by software functions of the rho 3 operating system.

The smallest unit of timer time depends on the hardware configuration used.

For a rho 3.1, the smallest unit of time is the transformation cycle (clock cycle) set by machine parameter P5.

For a rho 3.2, the smallest unit of time is the PIC cycle time which as a rule is 10 ms.

**10.1 Flag assignment of PIC250 timers and counters**

Flag assignment of timers and counters				
Option		Bit no.	PIC addr.	Signal description
0	8 counters	0..7	M0.0..M0.7	Decrements down-counter 0..7
		8..15	M1.0..M1.7	Down-counter 0..7 ready
		40..47	M5.0..M5.7	Down-counter 0..7 reset
	8 timers	16..32	M2.0..M2.7	Start timer 0..7
		24..31	M3.0..M3.7	Timer 0..7 is ready
		32..39	M4.0..M4.7	Timer 0..7 is active
1	16 counters	48..55	M6.0..M6.7	Decrements down-counter 8..15
		56..63	M7.0..M7.7	Down-counter 8..15 is ready
		88..95	M11.0..M11.7	Down-counter 8..15 reset
	16 timers	64..71	M8.0..M8.7	Start timer 8..15
		72..79	M9.0..M9.7	Timer 8..15 is ready
		80..87	M10.0..M10.7	Timer 8..15 is active
2	24 counters	96..103	M12.0..M12.7	Decrements down-counter 16..23
		104..111	M13.0..M13.7	Down-counter 16..23 is ready
		136..143	M17.0..M17.7	Down-counter 16..23 reset
	24 timers	112..119	M14.0..M14.7	Start timer 16..23
		120..127	M15.0..M15.7	Timer 16..23 is ready
		128..135	M16.0..M16.7	Timer 16..23 is active
3	32 counters	144..151	M18.0..M18.7	Decrements down-counter 24..31
		152..159	M19.0..M19.7	Down-counter 24..31 is ready
		136..143	M17.0..M17.7	Down-counter 16..23 reset
	32 timers	160..167	M20.0..M20.7	Start timer 24..31
		168..175	M21.0..M21.7	Timer 24..31 is ready
		176..183	M22.0..M22.7	Timer 24..31 is active

**Note:**

The option byte can be reset after consulting with Bosch.

## 11 Special function 3, Setting machine position

The special function "Set machine position" allows you to change internal machine positions from BAPS2 user programs. This can be particularly useful for incremental encoders in order, for example, to replace referencing.

The current machine position of each individual axis (joint coordinate) is yielded inside the control from the read operation of the measuring system, and for incremental encoders in conjunction with the referencing function.

### 11.1 Declaration

This option is implemented as a special function which must be declared in the declaration part of a program as described below.

```
SPC_FCT : 3 = MACH_POS (VALUE INTEGER : KIN_NO
                       KIN_NAME.JC_POINT : @P_NAME )
```

<i>MACH_POS</i>	Name of special function can be selected as desired.
<i>KIN_NO</i>	Number of the kinematics for which the machine positions are manipulated.
<i>KIN_NAME</i>	Name of the kinematic (or default kinematics if no name is specified) for which the machine position is manipulated.
<i>@P_NAME</i>	Name of the point issued when the special function is called. It contains the values (in [mm] or [degrees] ) at which the machine positions of the corresponding kinematics are set.

#### NOTE:

- If the machine positions of several kinematics are to manipulated in one program, the special function must be individually defined for each kinematic. The names in this case must be different (compare example 1 on page 3 – 25).
- @MACH\_POS may also contain belt values. However, these values cannot be manipulated.
- **To ensure that the control accepts the new values in joint coordinates as well as in world coordinates, a MOVE block which refers to the same kinematics must be programmed in the BAPS program after calling the special function. This can also be a dummy MOVE block which does not initiate or cause movement, such as**  
**MOVE\_REL KIN\_X (0,...,0).**  
If a MOVE block is not programmed, it may happen that the old positions are displayed under POS or Status (Mode 7, Mode 1 on the PHG) as world coordinates.

- The special function can be called from anywhere in a BAPS program. However, we recommend calling the special function directly following control start, without any previous movements. When this is done, the following program example replaces referencing.

**Example 1 :**

```

::CONTROL = RHO 3
::KINEMATICS : (1=SCARA , 2=PORTAL)
::SCARA.JC_NAMES =SC1,SC2,SC3,SC4
::SCARA.WC_NAMES = XSC,YSC,ZSC,OSC
::PORTAL.JC_NAMES = PO1,PO2,PO3
::PORTAL.WC_NAMES = XPO,YPO,ZPO

PROGRAM SPZ3

SPC_FCT : 3 = SC_MACHPOS (VALUE INTEGER : KIN_NR
                                          JC_POINT : @MACHPOS) ; Default kinem.
SPC_FCT : 3 = PO_MACHPOS (VALUE INTEGER : KIN_NO
                                          PORTAL.JC_POINT : @MACHPOS)

DEF SCARA.JC_POINT: @SC_ANFPOS
DEF PORTAL.JC_POINT : @PO_ANFPOS

BEGIN
      @SC_ANFPOS = @POS
      @SC_ANFPOS.SC4 = @SC_ANFPOS.SC4 - 360
      SC_MACHPOS (1,@SC_ANFPOS) ; Manipulate SCARA machine position
                                  ; of the 4th axis by 360 degrees
      MOVE_REL SCARA @(0,0,0,0) ; Dummy MOVE block to accept
                                  ; the new values

      @PO_ANFPOS = @(100,100,500)
      PO_MACHPOS (2,@PO_ANFPOS) ; Set PORTAL machine position
      MOVE_REL PORTAL (0,0,0) ; Dummy MOVE block to accept
                                  ; The new values

PROGRAM_END

```

**11.2 How the function for setting machine position works**

- Special function 3 sets the control-internal machine position to the joint coordinate point specified in the transfer parameter. The control no longer "knows" the machine's previous position. The method of operation corresponds to referencing with a new reference point actual value. The new joint coordinate position is displayed under @POS. The new machine positions are also obtained with @MPOS.
- At the same time, the current world coordinate position **POS** is changed. It is yielded from the forward transformation of the new machine position.
- As seen in the example above, special function 3 can be used for setting the machine position for all the axes simultaneously as well as for manipulating individual axes.
- The traversing range limits set with machine parameters P202, P203, P204, P205 are always based on the current machine position changed by special function 3.
- The machine position set in the program remains active until the special function is called again. "Control reset" does not change the machine position.



- If the control is started again, manipulation is not active which means that the values provided by the measuring system apply.
- If the reference points for the manipulated axes are reapproached after calling special function 3, the reference point actual values set in machine parameter P207 are used again, meaning that subsequent referencing resets the values set with special function 3.

## Caution!



- (1) **Note that if wrong machine positions are activated due to incorrect programming, the control will no longer "know" where the machine actually is.**  
**This may result in the following:**
- **Unexpected and apparently incorrect movements may occur, particularly in conjunction with a coordinate transformation.**
  - **The set traversing range limits may react incorrectly (too soon, too late, or not at all).**
- (2) **If a control is operating with special function 3 as well as with the referencing function, it must be ensured that the referencing speed (machine parameter P108) is identical to the 1st reduced referencing speed (machine parameter P109) as otherwise there is danger that an axis may approach the reference point switch at excessive speed during repeated referencing and then possibly finds an incorrect marker.**

## 12 Expansion of special functions 1 (IOL) and 2 (PPO)

Starting with software version TO02H or TO03C, active IO or PPO cycles can be reset by means of the program. This can be particularly useful when a movement program is stopped by a monitoring process with " **STOP** Proz\_x" while an IO or PPO function is active.

A reset requires the same declaration of special functions 1 and 2, meaning that the parameter list is identical.

### Declaration of the special function :

<b>SPC_FCT : 1 = I/O LOGIC(</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>:I/O LOGIC_NR</b>	;Function no.
	<b>VALUE</b>	<b>INTEGER</b>	<b>:KIN_NO</b>	
	<b>VALUE</b>	<b>INTEGER</b>	<b>:COORD_NO</b>	
	<b>VALUE</b>	<b>REAL</b>	<b>:IO_POSITION</b>	
	<b>VALUE</b>	<b>REAL</b>	<b>:PARAM_VALUE</b>	
	<b>VALUE</b>	<b>INTEGER</b>	<b>:TIME_OFFS</b>	
<b>SPC_FCT : 2 = PPO(</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>: PPO_NO</b>	;Function no.
	<b>VALUE</b>	<b>INTEGER</b>	<b>:KIN_NO</b>	
	<b>VALUE</b>	<b>INTEGER</b>	<b>:COORD_NO</b>	
	<b>VALUE</b>	<b>REAL</b>	<b>:IO_POSITION</b>	
	<b>VALUE</b>	<b>REAL</b>	<b>:PARAM_VALUE</b>	
	<b>VALUE</b>	<b>INTEGER</b>	<b>:TIME_OFFS</b>	

To reset an active IO/PPO function, special functions 1 and 2 are called with the corresponding **negative** function number.

In this case, the coordinate number, IO/PPO position and the time offset are ignored. The programmed parameter value is issued immediately and the corresponding IO/PPO function is enabled. In this way, the aborted program can be restarted without the error messages 'PPO:Fkt. still in use' or 'IOL:Fkt. still in use' being issued.

### Example:

- 1) I/O LOGIC ( -2,2,1,1, 0,0)
- 2) PPO ( -3,1,1,1, 50,0)
- 3) PPO (-204,3,1,1,100,0)

regarding 1) The I/O function no. 2 of the 2nd kinematics is reset. The corresponding binary output is deleted.

regarding 2) The PPO function no. 3 of the 1st kinematics is reset. The corresponding INTEGER output is set to the value 50.

regarding 3) The PPO function no. 204 of the 3rd kinematics is reset.  $100/V_{\text{rated}} * V_{\text{max}}$  [Volt] are output at the corresponding analog output.  $V_{\text{rated}}$  is the default rated value in machine parameter P405, and  $V_{\text{max}}$  is the maximum voltage of the servo board (13.3V or 10V) which can be output.

## 12.1 Response to interface signals and system states

If one of the following conditions occurs during a traverse block in which an IO or PPO function is active, the IO function is aborted and reset (meaning the output is cleared) and the PPO function is stopped (meaning no PPO change follows):

- FEED HOLD is triggered
- Feed allow is deactivated
- Travel allow is deactivated
- Reset issued via PHG/interface
- Emergency mode pending
- Interpolator stop
- Runtime error during program execution
- Movement stop of the active IO coordinate (even if the Stop is purposely programmed)

The following error message appears:

**"PPO/IOL : IPO - STOP "**

At the conclusion of the program, the last active PPO output value is not changed.

**Example:** for a monitoring process

```
.  
.
IF ANZ_FEHLER <> 0 THEN
BEGIN
      STOP          PROZ_I/O      ;Process with IO logic
      STOP          PROZ_PPO     ;Process with PPO logic
      START:       I/O_RUE      ;Reset IO logic
      START:       PPO_RUE     ;Reset PPO logic
END
```

## 13 BAPS2 - Language expansion

The new rho 3 software version expands the the BAPS2 language by the following language elements.

When using offline programming, these language elements are available with the ROPS3 software version W1K.

### 13.1 CASE – instruction

Starting with rho 3 software version TO03, BAPS2 commands are augmented by the branch instruction. This instruction makes selecting from several alternatives simple. This avoids nested "IF-THEN" queries. Program execution time is shortened as compared to nested "IF-THEN" instructions.

**SYNTAX:**

<b>CASE</b> <i>Selection expression</i>	<b>EQUAL</b> <i>List of constants</i>	<b>:</b> <i>INSTRUCTION</i>
	{ <b>EQUAL</b> <i>List of constants</i>	<b>:</b> <i>INSTRUCTION</i> }
	<b>DEFAULT</b> <i>INSTRUCTION</i> ]	
<b>CASE_END</b>		

Permissible type for *selection expression* : BINARY, INTEGER, CHAR, and TEXT.

The constants in the *list of constants* must be compatible with the selection expression. A constant may be used only once in a case instruction.

Specifying the DEFAULT part is optional. If the DEFAULT part is missing in the case instruction, there will be no runtime errors during program execution, even if no CASE condition is met.

Example for using the CASE instruction :

```
PROGRAM FALLS_BSP
:
:
:           Declaration
:
INTEGER : BETRAG,OFFSET
:
:
:           Program part
BEGIN
BETRAG = 0
OFFSET = 0

READ BETRAG                               ;
CASE      BETRAG
EQUAL    9,10      : OFFSET = 0
EQUAL    11,12     : OFFSET = 1
EQUAL    13        : OFFSET = 2
EQUAL    14        : OFFSET = 3
DEFAULT  OFFSET = 7
PROGRAM_END
```

## 13.2 STANDARD - CONSTANTS

The BAPS2 vocabulary has been expanded by the standard constants

**CLS** and  
**VERSION.**

### 13.2.1 Standard - Constant CLS (clear screen)

```
WRITE PHG,CLS
```

valid as of  
version TO02F

CLEAR SCREEN.  
The PHG display is cleared with "CLS".

### 13.2.2 Standard - Constant VERSION

```
IF VERSION < 2.31  
THEN WRITE  
'Old compiler version'
```

valid as of  
version TO02F

The version number of the compiler can be queried  
in the BAPS program.  
The constant is type REAL.

## 13.3 Constant declarations

Starting with software version TO03, it is possible to declare constants. A constant's value can also be expressed as a computational rule. Constants must be established with constant declarations before the variables of a program.

Example:

**CONST :**

```
RED = 1,YELLOW = 2 * RED,  
GREEN = 2 * YELLOW ,  
LINES = 4,COLUMNS = 20,  
LIMIT_VALUE = 2.5 * 7.25
```

*; Variable declarations*

```
INTEGER      :          COLOR, Z_INDEX, S_INDEX
```

```
REAL         : MESS-WERT
```

```
FIELD [ LINES .. COLUMNS ] CHAR : PHG_DISPLAY
```

```
;
```

```
.
```

```
.
```

*;Program part*

```
;
```

```
IF MESS_WERT >LIMIT VALUE THEN .....
```

## 14. ROPS3/IQPRO Version W1L

### 14.1. Compatibility with ROPS3

Software version						
ROPS3 / IQpro	Control rho 3/IQ140	BAPS2– compiler version	Control IQ120	PROFI SOFTWARE	PIC250 Program file .QLS for BAPS–PIC .P20 for PROFi	CL300 Program file .P30 for PROFi
W1C W1F	TO01E TO01I			2.43	R3_1I_D.QLS	RHO3_1I.P30
W1J	TO02F	2.3	1.10	3.0	R3_2D_D.QLS R3_2D.P20	RHO3_2D.P30
W1K	TO03F	2.31	2.00	3.0	R3_2D_D.QLS R3_2D.P20	RHO3_2D.P30
W1L	TO03F	2.32	2.00	3.12	R3_4A_D.QLS R3_4A.P20	RHO3_4A.P30

The following should be noted for the installation and use of the ROPS3 software version W1L:

1. Installation problems with active MS–DOS driver SHARE.EXE

If the driver SHARE.EXE is active, the error message:

```
< Not able to read drive x      >
< Cancel/Repeat                >
```

may appear during installation of the second floppy of the ROPS3 software.

**Remedy:**

Deactivate the driver SHARE.EXE in your system, e.g. by changing the AUTO-EXEC.BAT or CONFIG.SYS files.

Start your computer again and install the ROPS3 software.

After installation of ROPS 3 you may activate the SHARE.EXE driver again.

2. New functions in ROPS3

As of software version W1K the BAPS short description is available as online help.

3. **Notes**

Please also note the information on ROPS3 in part 1 of the present document.

**Index****A**

absolute measuring system, 5 – 24  
ASC\_ARRAY, 2 – 41  
ASSIGN, 4 – 31

**B**

BATCH–Dateien, 2 – 62  
Edi.BAT, 2 – 64  
Info.BAT, 2 – 63  
PUeb.BAT, 2 – 64  
RLad.BAT, 2 – 63  
RLis.BAT, 2 – 63  
RLoe.BAT, 2 – 63  
RNeN.BAT, 2 – 63  
RSet.BAT, 2 – 64  
RSpe.BAT, 2 – 64  
RSpeUe.BAT, 2 – 64  
RUeb.BAT, 2 – 64

**C**

CAN, 5 – 24  
Module input, 5 – 24  
Pulses per rotation, 5 – 25  
CLS, 3 – 31

**E**

EEPROM, User memory in, 4 – 7  
EXCLUSIVE, 2 – 42  
EXCLUSIVE\_END, 2 – 42  
EXTERNAL, 2 – 42  
external, program selection, 2 – 47

**G**

go block, limitation, 5 – 44

**I**

incremental measuring system, Queries in P401,  
5 – 23  
Inputs  
asynchronous, 5 – 45  
fast, 4 – 12  
INT\_ASC, 2 – 40  
Interpolator, Stop, 4 – 27

**M**

Module number, 5 – 23

**P**

Plug number, 5 – 23  
Potentiometer measuring system, 5 – 25  
PUBLIC, 2 – 42

**R**

Resolver, orientationally correct referencing, 5 – 24,  
5 – 25

**S****SAFETY NOTES**

A/D/V factor, 4 – 15  
Max. axis speed, 4 – 28  
Memory test, 4 – 16  
Options, 4 – 17

Schnelles, konstantes Losfahren, Erweiterung,  
4 – 52

SEMAPHORE, 2 – 42

servo card, 5 – 23

**SPECIAL FUNCTION**

Spec. 29, 4 – 18  
Spec. 30, 4 – 21  
Spec. 31, 4 – 22  
Special function 4; Call system function, 4 – 35  
Special function 15, 4 – 47  
Special function 21, 4 – 42

Special function, 5 – 35

1,2 = set and reset outputsignals, 3 – 27  
3 = setting machine position, 3 – 24  
41 = asynchronous speed change, 5 – 5  
42 = asynchronous target stipulation, 5 – 5  
45 = MOVE\_FILE, 5 – 14  
46 = start block limitation, 5 – 44

**STANDARD PROCEDURE**

ASC\_INT, 2 – 41  
INT\_ASC, 2 – 40

Strobe, INTEGER inputs, 4 – 16

SYNCHRONOUS BELT, without parallel belt axis,  
4 – 49

Synchronous belt, Synchronization type 3, 4 – 54

**T**

Transformation, Forwards,, 3 – 25

**V**

VERSION, 3 – 31

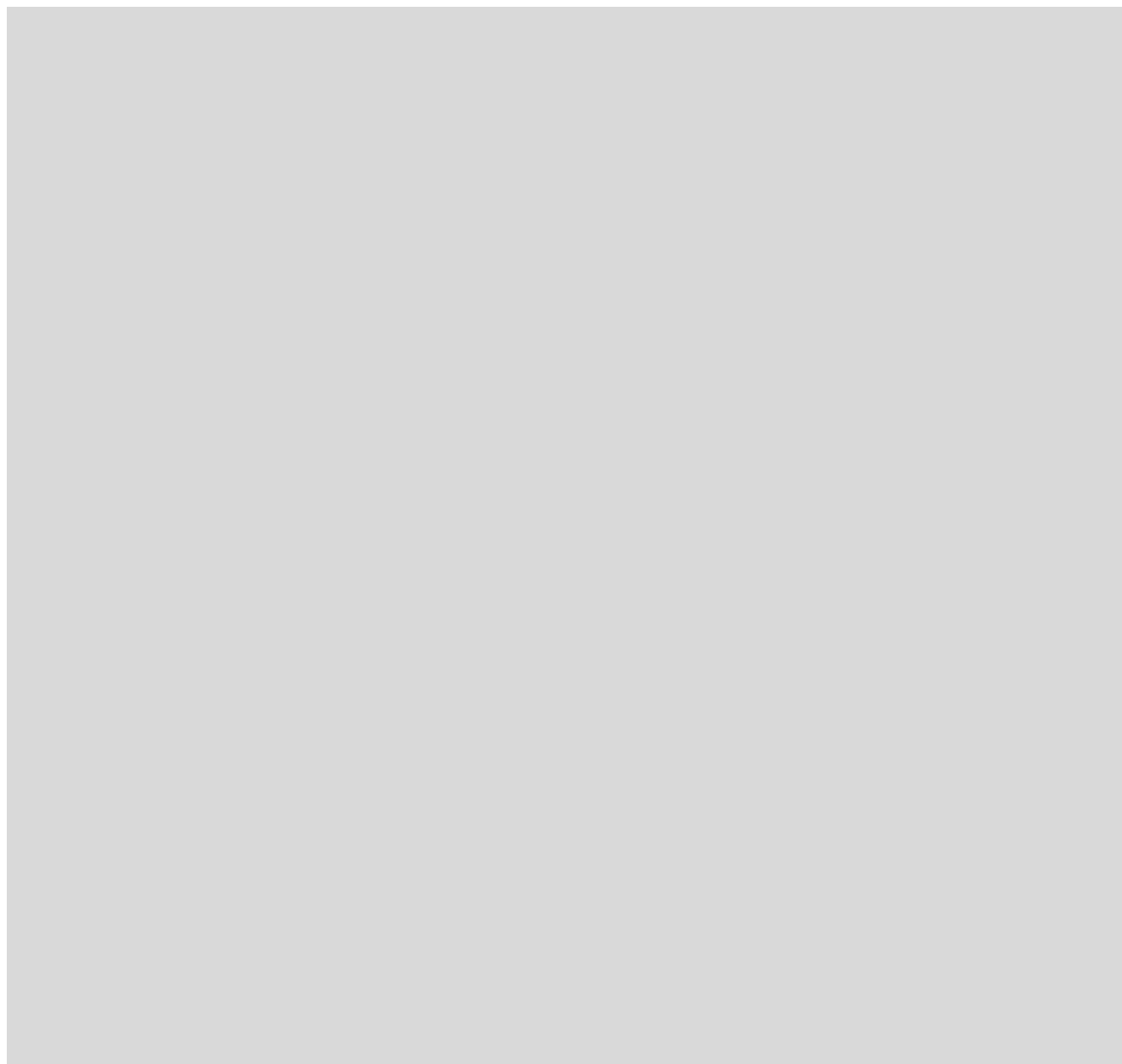
**W**

WARTE BIS, MAXZEIT, 4 – 52



rho 3

# Releases Part 2 Versions TO04 and TO05



Version

# 107



*rho 3*

# Releases Part 2 Versions T004 and T005

1070 073 078-107 (97.01) GB



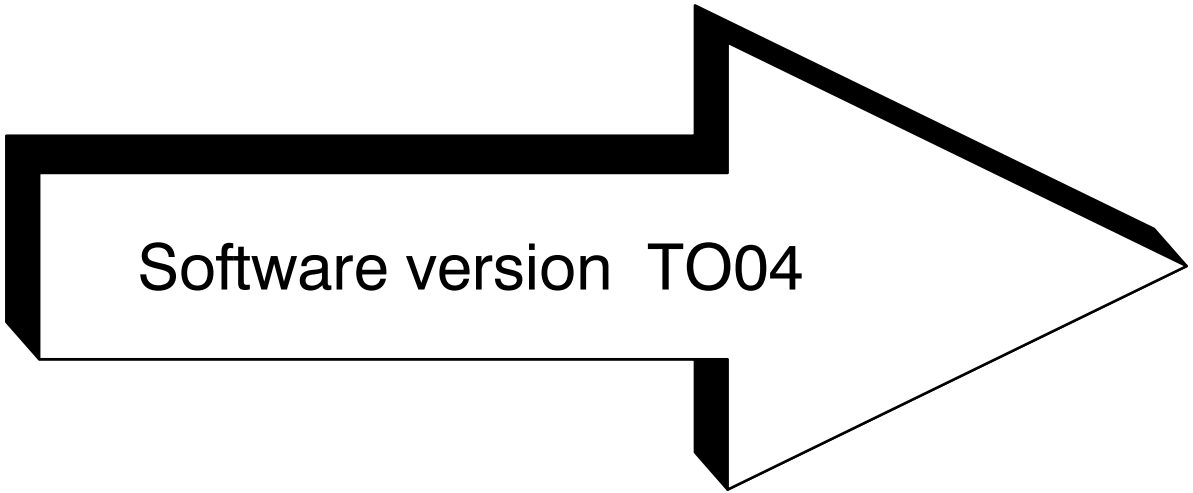
Reg. Nr. 16149-03

© 1995 - 1997

by Robert Bosch GmbH,

All rights reserved, including applications for protective rights.  
Reproduction or handing over to third parties are subject to our written permission.

Discretionary charge 50.– DM



Software version T004

## Contents Part 2

### Software version T004

<b>1</b>	<b>General notes on software version T004</b>	
1.1	CP/MEM5 module .....	4 – 1
1.2	30MHz CPU module .....	4 – 1
1.3	25 MHz CPU module .....	4 – 1
<b>2</b>	<b>Expansion of the function "Fast, smooth start-off" .....</b>	<b>4 – 3</b>
<b>3</b>	<b>Directly approaching points in Teach IN</b>	
3.1	Operation .....	4 – 4
3.2	Preventing points which have been taught in from being overwritten .....	4 – 5
3.3	Restrictions .....	4 – 6
<b>4</b>	<b>Saving the user memory in the EEPROM</b>	
4.1	Requirements .....	4 – 7
4.2	Retrieving the user memory from the EEPROM .....	4 – 8
4.3	Marginal conditions .....	4 – 8
<b>5</b>	<b>Selecting PHG functions using interface signals</b>	
5.1	Description of function .....	4 – 9
5.1.1	Selection .....	4 – 9
5.2	Allocation file .....	4 – 9
5.2.1	Allocation file structure .....	4 – 10
5.3	Deselection, return to the basic level .....	4 – 10
5.4	Interface assignment .....	4 – 11
5.5	Restrictions, handling errors .....	4 – 11
<b>6.</b>	<b>Displaying "high-speed inputs" on the PHG</b>	
6.1.	Selecting the display on the PHG .....	4 – 12
6.1.1	Structure and contents of the display .....	4 – 13
<b>7</b>	<b>Changes to the machine parameter input process</b>	
7.1	Entering a password .....	4 – 14
7.2	Scrolling when selecting a group .....	4 – 14
<b>8</b>	<b>New parameters starting with version T004A .....</b>	<b>4 – 15</b>
8.1	Displaying set options .....	4 – 17
<b>9</b>	<b>Recording the reference path (movement history)</b>	
9.1	Activating the reference path recording function, special function 29 .....	4 – 18
9.2	Storing the position values in the transformation cycle or rough interpolation cycle (clock pattern) .....	4 – 20
9.3	Deactivating the reference path recording function, special function 30 .....	4 – 21
9.4	Reading the reference path value, special function 31 .....	4 – 22
9.5	Program examples (partial) .....	4 – 24

<b>10</b>	<b>Limiting axis speeds in linear mode</b> .....	<b>4 – 27</b>
10.1	Machine parameters for limiting axis speed .....	4 – 28
10.2	Previous axis speed limits .....	4 – 28
<b>11</b>	<b>Manual axes as endless axes</b> .....	<b>4 – 29</b>
<b>12</b>	<b>The BAPS2 instruction ASSIGN</b>	
12.1	Syntax ASSIGN instruction .....	4 – 31
12.2	ASSIGN instruction .....	4 – 32
12.3	Compatibility .....	4 – 32
12.4	Using ASSIGN on standard channels .....	4 – 32
12.5	Runtime conditions .....	4 – 33
12.6	Compiler instruction FILE_ERROR .....	4 – 33
12.7	Restrictions on selecting file names .....	4 – 34
12.8	Application example for ASSIGN .....	4 – 34
<b>13</b>	<b>Calling operating system functions from the BAPS2 user programs</b>	
13.1	Operating system commands .....	4 – 35
13.2	Declaration part .....	4 – 35
13.3	Commands .....	4 – 37
13.3.1	Compiling BAPS programs .....	4 – 37
13.3.2	Copying files .....	4 – 37
13.3.3	Deleting files .....	4 – 38
13.3.4	Starting processes .....	4 – 38
13.3.5	Stopping processes .....	4 – 38
13.4	Example .....	4 – 39
13.5	Restrictions, error treatment .....	4 – 41
<b>14</b>	<b>Belt type selection, special function 21</b>	
14.1	Declaring special function 21 .....	4 – 42
14.2	Using special function 21 .....	4 – 43
14.3	Notes and error messages regarding special function 21 .....	4 – 44
<b>15</b>	<b>Belt synchronization without the robot being able to move parallel to the belt</b> .....	<b>4 – 45</b>
15.1	Operating conditions for belt synchronization type 2 .....	4 – 46
15.2	Special function for parameterisation of the belt characteristics .....	4 – 47
15.3	Belt stop/belt start for belt synchronization type 2 .....	4 – 49
15.3.1	Meaning of the threshold value for the belt status "Belt stopped" .....	4 – 49
15.3.2	Meaning of the threshold value for the belt status "Belt running" .....	4 – 50
15.4	Bypassing the error message "Ax velocity exceeded" .....	4 – 50
15.5	General notes on synchronization type 2 .....	4 – 50
15.6	Restrictions due to this type of belt synchronization in conjunction with other options .....	4 – 51
15.7	Program example .....	4 – 52
<b>16</b>	<b>Belt synchronization type 3 (cam interpolation)</b>	
16.1	Operating conditions for belt synchronization type 3 .....	4 – 54
16.2	Description of the parameterized curve .....	4 – 56
16.3	Programming the sections .....	4 – 56
16.4	Special cases .....	4 – 57
16.4.1	Comparison with normal traverse blocks .....	4 – 57
16.5	Program example .....	4 – 58

**17**      **Several axes on one setpoint output** ..... **4 – 60**  
17.1      Setting machine parameters ..... 4 – 61

**18**      **ROPS3/IQPRO**  
18.1      Compatibility list: ..... 4 – 62

## 1 General notes on software version TO04

The following functions and options are available for the rho 3 control configuration with the new software version.

This new software version is compatible with versions TO02 and TO03.

**When converting from a TO01 software version to TO04, please also read the Notes in part 2 of this documentation.**

### 1.1 CP/MEM5 module

After October 1993, the CP/MEM4 module was replaced with the CP/MEM5 module. The CP/MEM5 is compatible with the the CP/MEM4 and can be used in place of the CP/MEM4. For those CP/MEM4's which are currently equipped with a 25MHz processor module, a 30MHz module must be inserted instead, with at least the rho 3 software version TO04.

### 1.2 30MHz CPU module

This new software version supports use of the 30MHz CPU module. The software automatically recognizes the inserted processor module.

This module can only be used in rho 3.1 configurations on a CP/MEM5 module.

### 1.3 25 MHz CPU module

On the intelligent servo boards:

SERVO 3i,

SERVO 4i and SERVO 6i

the 25MHz processor module can still be used.

**Note:**

**The 25MHz CPU module cannot be used on the SERVO m.i. and not in the rho 3.2 configuration on the CP 2.5 CPU board.**





## 2 Expansion of the function "Fast, smooth start-off"

Please also refer to the description of the function "Fast, smooth start-off" in part 2 of this document.

### ATTENTION !

This function has been specially developed for quickly starting off.

Up to operating system version TO03, a movement block **must** be programmed in the BAPS program after the command "WAIT UNTIL high speed input" (for example, MOVE, MOVE UNTIL, MOVE\_REL, CIRCULAR).

**If the "WAIT UNTIL" command in the program is followed by blocks which do not initiate movement (for example, Set output, Read input, WAIT time), they may be executed too soon, i.e., before the "WAIT UNTIL" condition occurs.**

This restriction no longer applies as of operating system version TO04, i.e. the command "WAIT UNTIL high speed input" can be followed by any BAPS2 command. Programmed, sequential processing is always guaranteed. However, subsequent movement blocks can only be started with an additional delay of two transformation cycles (P2 clocks).

Example:

```
1      WAIT UNTIL HS_31=1
2      OFF1 = 1
3      OFF2 = 0
4      WRITE PHG, 'HS_31 is set'
5      MOVE ZIELPOS
```

The movement of the block (5) is started with a delay of (2 X P2—clocks + delay time).

The command "WAIT UNTIL HS input" always ensures synchronization with the transformation cycle (rough interpolation cycle). This means that the command **cannot** be programmed in a pure I/O process.

### 3 Directly approaching points in Teach IN

The function for directly approaching points in Teach in mode makes it easy to change points for which teach-in has already been performed. For example, when a packing system is commissioned the removal positions and deposit positions have already been defined by teach-in. Assume the user would like to make slight corrections to these positions. Because the positions are far apart, it is time-consuming to manually move from the removal position to the deposit position. Using a traverse command, the user can move directly to the deposit position which results in considerable time savings.

#### 3.1 Operation

Select Teach in.

Select Teach in mode on the PHG with one of the following:

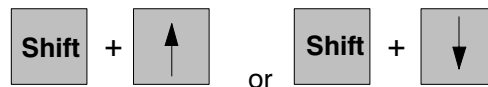
- Mode 3.1.3 (Programming, BAPS program, Teach in),
- Mode 4.2 (Define, Teach)

or

- Mode 5.14 (BAPS program Test, Teach in).

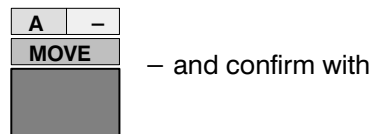
The desired point is selected after the Teach in mode is activated.

Scroll with the following keys to go to the point where the selection is to be made:



Or, enter the name of the point.

After selecting the point, press



Movement to the desired point is started.

The robot approaches the coordinate values (already taught in) which are set in the point.

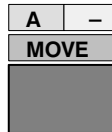
The movement is immediately aborted as soon as the confirmation key is released.

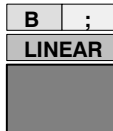
The movement can only be started again by repeating the selection procedure above.

**PHG display:**

**MOVE  
@POSITION**

**Execute with : ENTER**

If the movement is to be linear, in addition to the  key, the

 key on the PHG must also be pressed.



**PHG display:**

**MOVE LINEAR  
POSITION2**

**Execute with : ENTER**

Speed and acceleration are derived from the machine parameters for JOG speeds **P100** group.

The speed override for VFACTOR, AFACTOR and DFACTOR can be set with PHG Mode 11.4,5,6 "Help functions", as well as with the interface (PIC, PLC).

Using the key combination  +  the operator can switch to the position display during the movement operation.

When the robot reaches the point, a new value can be taught in for the selected point.

### **3.2 Preventing points which have been taught in from being overwritten**

When the user is teaching points and he presses the ENTER key, the message  
" **Pkt overwrite:ENTER**"

appears to inform him that the currently selected point will be overwritten if he presses the ENTER key again. If any other key is pressed, the old point value is retained.

### 3.3 Restrictions

Directly approaching a teach point should be linear. If the robot (for example, SR450) is changed from right-handedness to left-handedness (i.e., change of sign on axis 1), the rho 3 issues the message:

**"Programmed point not reachable with actual interpolation type"** .

In this case, PTP must be activated to approach the desired point.

## 4 Saving the user memory in the EEPROM

To save the user program beyond the buffer duration of the battery buffer, it is possible to save the contents of the user memory in the EEPROM (electrically erasable programmable read-only memory). If required, it is then possible by means of PHG commands to automatically or manually load the contents of the EEPROM into the RAM from which the user program was executed.

Storing the user program to the EEPROM can be initiated using the PHG; Mode 9.6, Memory --> EEPROM .

### 4.1 Requirements

This function is only available in conjunction with  
rho 3.1 (96 kBytes user memory / 128 kBytes EEPROM) or  
rho 3.0 (128 kBytes user memory / 256 kBytes FLASH–EEPROM)  
and when the following conditions are met:

- There is enough EEPROM
- EEPROM write protection is not active
- No processes are active
- All files are closed
- EMERGENCY STOP is pending
- At least 512 bytes are free in the user memory
- The option is activated

Error messages on the PHG :

- |                                  |  |
|----------------------------------|--|
| – <b>'EEPROM not enough'</b>     | Required memory is larger than the EEPROM                                      |
| – <b>'EEPROM write protect.'</b> | Write protection is active   |
| – <b>'Prog. is active'</b>       | Can only be saved if no program is active                                      |
| – <b>'File(s) not closed'</b>    | All files must be closed   |
| – <b>'no EMERGENCY STOP'</b>     | Save operation only possible in EMERGENCY STOP status                          |
| – <b>'Memory too full'</b>       | EEPROM capacity (96 Kbyte) insufficient for saving the user programs and files |
| – <b>'Option not found'</b>      | Option must be activated   |

To avoid error messages, we recommend saving the user memory only directly following control start-up. control start-up is automatic after the save operation.

## 4.2 Retrieving the user memory from the EEPROM

Retrieving the user memory from the EEPROM and into the RAM is initiated with the PHG (Mode 9.7, EEPROM --> Memory), or by means of interface signals.

This is only possible if:

- No processes are active
- EMERGENCY STOP is pending
- EEPROM memory and user memory have same size
- User memory contents were saved in the EEPROM beforehand
- The option is activated

Error messages on the PHG as described under 4.1:

- 'Prog. ist active'
- 'no EMERGENCY STOP'
- 'Memory <> EEPROM'
- 'EEPROM data n. valid'
- 'Option not found'

Every time an EPROM backup operation occurs, the contents of the EEPROM are saved to the RAM if the above conditions are met.

Any user programs in the user memory are no longer available after retrieval from the EEPROM.

If the function has been completed successfully, the control starts up automatically.

## 4.3 Marginal conditions

This option is only available for the rho 3.1 and rho 3.0 with corresponding hardware (cf. point 4.1). The user memory must be exactly 96 kBytes (rho 3.1) or 128 kBytes (rho 3.0). If another user memory is applied, an error message is issued for PHG commands MODE 9.6 and MODE 9.7, and the commands are not executed.

The function can be disabled or enabled as required.

## **5 Selecting PHG functions using interface signals**

Previously, PHG operator interface functions could only be activated by means of PHG operations. Exceptions are manual mode and the referencing function.

These can also be selected using interface signals. In many cases, the ability to make a selection by simply pressing a key, or from within a user process, is desired.

In addition, it often makes sense to reduce the number of operation possibilities. The option for selecting PHG functions using the interface was developed in order to provide the user with a tool which he can use to design his own operator interface.

### **5.1 Description of function**

Coded selection of PHG functions using the interface is optional. It can only be used if the relevant option bit (option 41) is set in the machine parameters.

#### **5.1.1 Selection**

Due to the existing interface structure (RC inputs and outputs), only coded selections are possible.

Encoding is in hex form with 4 bits. In this way it is possible to select 16 different functions.

Activation of coded PHG functions is initiated with the positive edge of the RC input 49, PIC address O6.1, "Strobe, PHG function using the interface".

Code=0 with falling edge of the strobe signal is for selecting the basic level.

A coded selection can only be executed if the basic level of the PHG operator interface is active. This status is indicated with "1" by means of the RC output, PIC address I13.0, active basic menu.

### **5.2 Allocation file**

The allocation of code numbers to the desired PHG function is done by means of a file.

The file name for the German version is "**PHGKODE.DAT**", and for the English version "**PHGCODE.DAT**".

### 5.2.1 Allocation file structure

In the allocation file, each code number is assigned a function of the PHG menu tree. The format of the PHG function corresponds to the "Direct selection" of PHG functions.

Example:

Structure of the <b>PHGCODE.DAT</b> file			
<b>Code value</b>	<b>=</b>	<b>assigned PHG mode</b>	<b>; Comment</b>
0	=	0	;Code=0 with negative edge initiates ;return to basic level
1	=	3.2.2.1	;Display PIC timers and counters
2	=	2	;Manual
3	=	4.2	;Teach
4	=	7.1	;Axis positions
A	=	7.3.2	;Process conditions
F	=	11.1	;Initial position on the PHG

The PHG function "Display PIC timers and counters" is to be activated with code=1. For conventional inputs, it would be necessary to enter Mode 3, Mode 2, Mode 2 and Mode 1.

With direct selection, this is Mode=3.2.2.1.

Thus, the line "1=3.2.2.1" must be written in the allocation file.

The allocation file can contain blanks, empty lines and comments. Comments are initiated with a semi-colon ";," and apply to the rest of the respective line.

The code number must be written as a one-place hexadecimal number (1..F).

It is not necessary to occupy all codes. Allocating a PHG function to several different codes is permissible.

### 5.3 Deselection, return to the basic level

Selecting a PHG function and returning to the basic level can be done with code=0 and the negative edge of the strobe signal.

This return operation can be executed at any time, even when dynamic displays are active (for example, lag) or during an input operation (for example, VFACTOR).

For deselection, it is not essential to know which PHG function was assigned to the code "0" in **PHGCODE.DAT**. In this way, the code "0" can be used for selection. However, note that as soon as there is no selection signal (for code=0), the system immediately returns to the basic level.



## 5.4 Interface assignment

Starting with version TO04H, the signals for the coded selection of PHG functions for RC inputs is available via interface.

RC inputs:					
BAPS name	Bit no.	PIC addr.	CL300 addr.	RC input	Signal description
	49	O 6.1	O5.2		Strobe, coded PHG function selection
	56	O7.0	O6.0	6.1	Coded PHG function selection, significance 1
	57	O7.1	O6.1	6.2	Coded PHG function selection, significance 2
	58	O7.2	O6.2	6.3	Coded PHG function selection, significance 4
	59	O7.3	O6.3	6.4	Coded PHG function selection, significance 8

RC outputs:					
BAPS name	Bit no.	PIC addr.	CL300 addr.	RC input	Signal description
	104	I13.0	I11.0	6.1	Basic menu level active

## 5.5 Restrictions, handling errors

- There must be no blanks within the PHG function designation in short form (for example, 7. 3 .2 is illegal).
- The character following the last Mode no. of a line must be either a blank " ", a semi-colon ";", or <CR><LF>. A comment must be initiated by a semi-colon ";". The rest of the line is then skipped.
- Selections via interface are not possible if the basic level of the PHG operator interface is not active.
- If a code is activated for which no allocation exists in **PHGCODE.DAT** the error message **"Error in PHGCODE.DAT"** is displayed.
- Errors in the allocation file are only recognized during runtime, i.e., while selecting a PHG function. Such errors lead to the error message

**" Error in PHGCODE.DAT.**

In addition, an error code is output at the interface.

## 6. Displaying "high – speed inputs" on the PHG

"High – speed inputs" are those inputs that are physically available on the servo boards. Various servo card types can be inserted in an rho 3 control. Depending on the type, the boards 8 and 9 have no high – speed inputs.

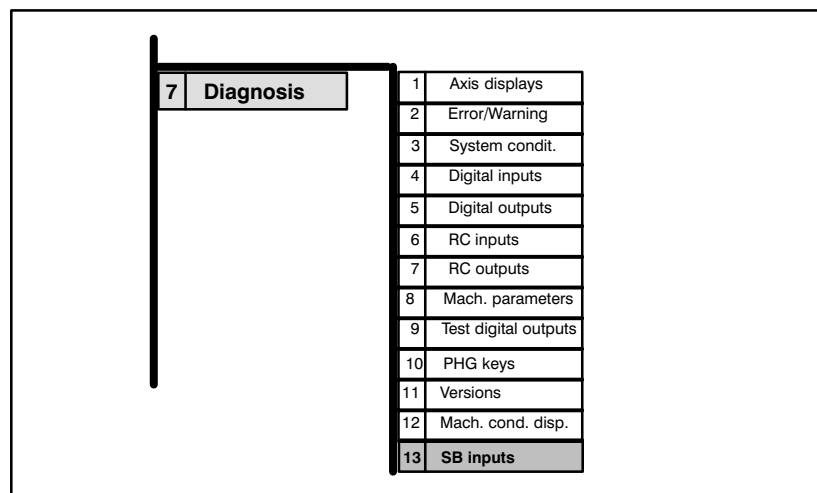
### Overview of the possible servo boards

Servo board type	Connector	Number of possible inputs
SERVO 3i S	X31	9
SERVO 4i S	X31	9
SERVO 6i S	X31	9
SERVO 8 S	X31	8
SERVO m.i.S	X31	9 (all versions)
SERVO 3		none
SERVO 5		none

The number of the high – speed inputs in use must be indicated with machine parameter **P11** for each inserted servo board.

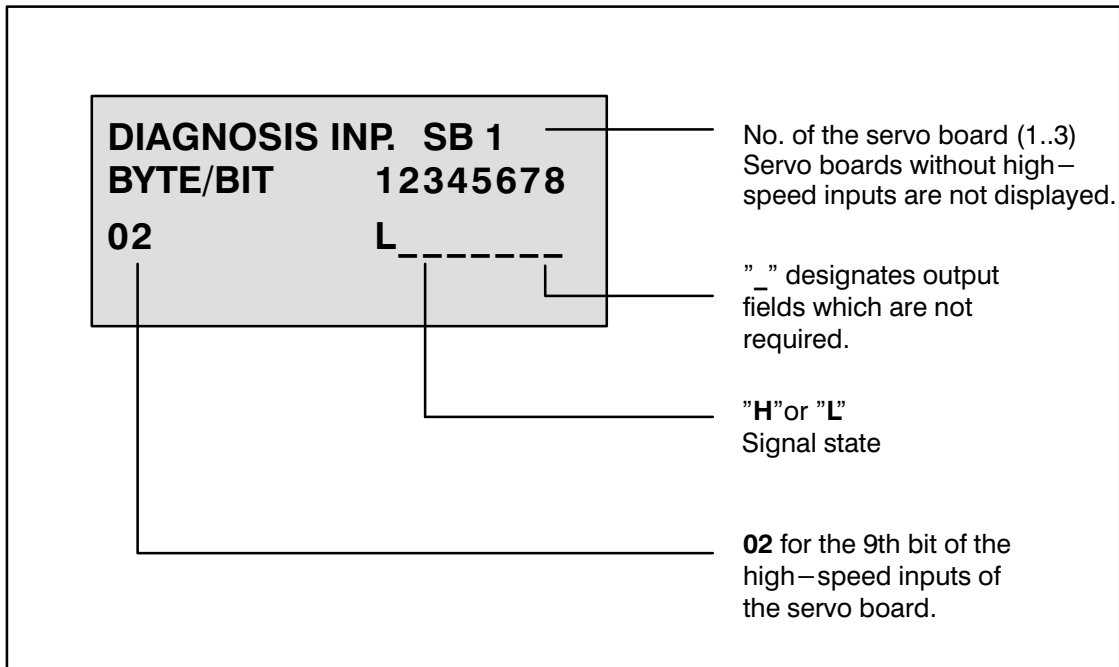
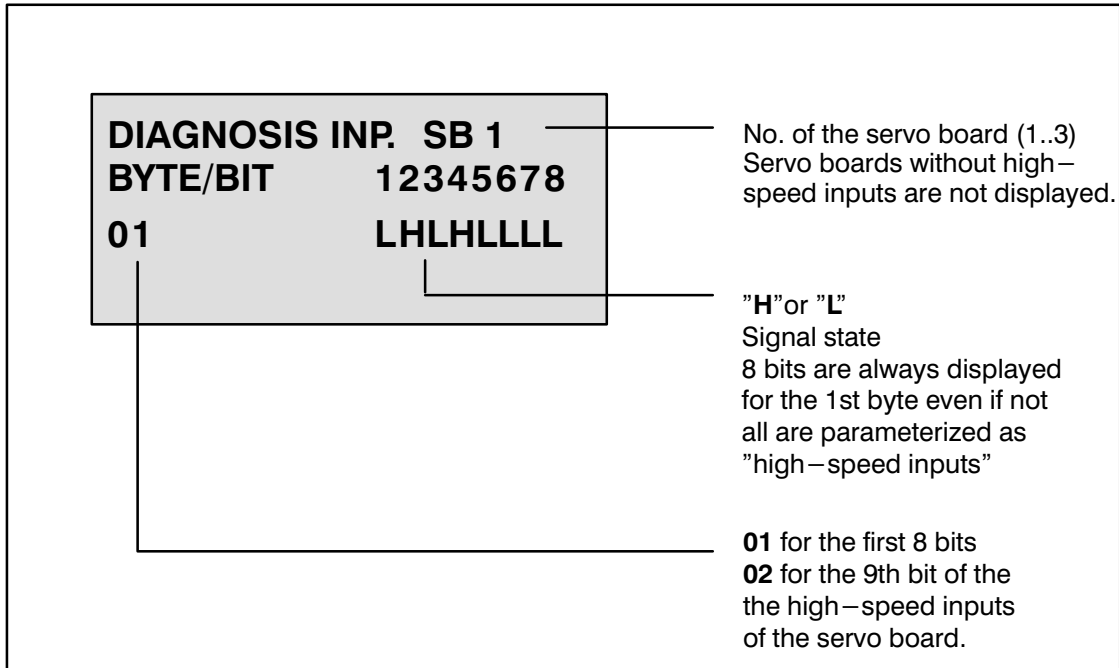
### 6.1. Selecting the display on the PHG

Display is similar to that for digital inputs in "DIAGNOSIS", menu point 13, "SK inputs" (Mode 7.13).



If at least one high – speed input is specified in machine parameter P11, the conditions of all the high – speed inputs of this servo board are displayed. Servo boards which have no hardware-end high – speed inputs are not shown (they are skipped).

**6.1.1 Structure and contents of the display**



## 7 Changes to the machine parameter input process

In order to facilitate making machine parameter inputs, the functions described in the following are available starting with software version TO04H.

### 7.1 Entering a password

With the new software version it is possible to enter a password in the Set mode (Mode 7.8.2) before each password-protected parameter.

If a parameter is selected after selecting the MPP (which cannot be changed without a password) without first entering the password, the following error message is issued:

"Enter password!  
any key --> continue"

After pressing any key, the following text appears:



" **Password**"

The password must then be entered.



### 7.2 Scrolling when selecting a group

When selecting the parameter groups **PALL,P100** and so on, it is possible to scroll forward or backward beyond the first and last parameter.

Example 1:

If parameter **P101** is shown and the key  +  is pressed, the last parameter of the selected group is switched to (here: **P130**).

Example 2:

If parameter **P130** is shown (currently the last parameter of the **P100** group) and the key  +  is pressed, the parameter **P101** is displayed.

## 8 New parameters starting with version TO04A

- P3 Number of timers and counters
- P25 Reset the A/D/V factors
- P26 Memory check
- P27 Strobe INTEGER inputs
- P28 Display available options

### P3 number of timers and counters

The number of the usable timers and counters in the PIC250 program is set with parameter P3.

Permissible entries are:

8, 16, 24 or 32.

The maximum number of permissible timers and counters is specified with the options byte. The default value is 8 when the control is delivered.

### P25 Resetting the A/D/V factors

The reset behavior of the global and kinematic-dependent A/D/V factors is established with P25.

Permissible input values are "0" and "1".

The following table shows how the default settings (all "0") of the individual factors work:

Operation	global factors	kin.-dependent factors
CONTROL RESET	1.0 (100%)	unchanged
AUTO ==> MANUAL	unchanged	unchanged
MANUAL ==> AUTO	unchanged	unchanged
PROCESS SELECTION	unchanged	1.0 (100%)

Global factors for	become/remain
CONTROL RESET	unchanged
AUTO ==> MANUAL	1.0 (100%) unchanged
MANUAL ==> AUTO	1.0 (100%)
PROCESS SELECTION/START	1.0 (100%) (*)

### ATTENTION !

**The change marked with \* acts on all kinematics, including the kinematics already moved in a process.**

Kinematic-depend. factors for :	become/remain
CONTROL RESET	1.0
AUTO ==> MANUAL	1.0
MANUAL ==> AUTO	1.0
PROCESS SELECTION/START	unchanged

Changing with option (No. 28) is still possible.

### **P26 Memory checks**

The RAM and EPROM tests which are executed during the start-up phase can be deactivated or activated by means of P26. Input possibilities:

0 = RAM and EPROM test active (default)

1 = RAM and EPROM test deactivated

Changing with option (No. 20) is still possible.

### **ATTENTION !**

**The control RAM and EPROM test is a measure for satisfying the requirements placed on controls in accordance with VDI 2853. Deactivating the RAM or EPROM tests is prohibited on all systems to which these VDI or similar directives apply.**

### **P27 Strobe INTEGER inputs**

The strobe for the INTEGER inputs can be switched off with P27. Input possibilities:

0 = with strobe (default)

1 = without strobe

Changing with option (No. 20) is still possible.

## 8.1 Displaying set options

When **P28** is selected the values of all the set options are displayed; i.e., option byte <> 0.

The values of the corresponding option byte are displayed as two-place hexadecimal numbers.

The display begins with the first set option (according to the option list) and ends with the last set option (max. 64). The options that can be changed by parameters or that are not used will not be displayed.

No cursor is shown in **P28**. Only the keys "SHIFT-ARROW UP (scroll up), "SHIFT-DOWN ARROW, or ENTER (scroll down) are allowed; any other inputs generate an error message.

Example of the display under **P28**:

### **P28 activated options**

Circular:	01
Mirror:	01
Gripper-coord.:	01
File-I/O:	01
Linear-interp.:	01
Program-slope:	01
Sinus*2-slope:	01
Cod. text.,MSD:	01
Tool:	01
A/D/V-factor IF:	01
Read POS:	01
Several kin.:	01
Coupl.-coord.:	01
FA witho. ref.-p.:	01
Decel.progslope:	01
Logbook:	01
Passing:	01
3964R protocol:	01
Min.p-clocknumb.:	03
Global data:	01
Decel.passing:	01
Toler.driveoff:	02
Set machpos.:	01
Online-funct.:	01
Belt stop:	01
BSYN witho. axis:	01

### **ATTENTION !**

**Option bytes may only be changed by authorized personnel and in consultation with BOSCH. Incorrect option values can lead to control malfunctions.**

## 9 Recording the reference path (movement history)

The special function described in the following allows the user to retrace the movement sequence of his machine in the event of an abort condition.

The use of these functions is useful in those places, for example, where the kinematics move synchronously with a belt and the belt cannot be moved properly by means of the robot control.

If the belt stops during synchronization, it "runs down" in a given specific time. The functions described in the following can be used to retrace the movement of the machine (of the belt).

In addition, the functions make it possible to record measured values in synchronization with the programmed movement sequence via belt channels which are assigned to the moved kinematics. This in turn allows the dynamic recording of contour deviations (dynamic measuring).

### 9.1 Activating the reference path recording function, special function 29

With special function 29, the function for saving set point values is activated. The data are stored in a ring memory. The ring memory contains 50 entries. Joint coordinates as well as world coordinates are saved. The points in time when a save operation is executed are yielded from the path change of a freely selectable measuring system input. The path change, the measuring system input as well as the allocated kinematics are all parameters of the special function.

#### Declaring special function 29

<b>SPC_FCT : 29 = PKT_SPEI_EIN(</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>: KIN_NO</b>
	<b>VALUE</b>	<b>INTEGER</b>	<b>: MS_NO</b>
	<b>VALUE</b>	<b>REAL</b>	<b>: DISTANCE)</b>

**KIN\_NO** - Kinematics number:

KIN\_NO indicates the kinematics whose coordinate values are stored. The coordinate values of the belts which belong to these kinematics are also stored.





## 9.2 Storing the position values in the transformation cycle or rough interpolation cycle (clock pattern)

For special function 29 of the function " **Recording movement history**", it is also possible to store points which are in the time-slot pattern of the transformation cycle.

Switch-over, by means of a machine parameter, is optional.

OPTION BYTE = 0 : The reference path is recorded if the tolerance specified in *DISTANCE* ([mm] or [degrees]) is exceeded

OPTION BYTE = 1 : Points are stored in the transformation time-slot pattern  
*DISTANCE* = number of transformation cycles

In the parameter " *DISTANCE* of special function 29, the number of transformation cycles (CLOCKS P5) is transferred as a *decimal number* which serves as a time base for storing the point values.

### Example:

Declaration :

```
SPC_FCT : 29 = PKT_SPEI_EIN ( VALUE INTEGER :KIN_NO
                               VALUE INTEGER :MS_NO
                               VALUE REAL :DISTANCE )
.
.
.
```

Call:

```
PKT_SPEI_EIN (1, 4, 10)
.
.
.
```

In the example, the point values of the first kinematics are stored in each 10th transformation cycle (rough interpolation cycle).

In this case the measuring system number (4) is redundant. Nevertheless, a valid value must be transferred when the special function is called in order not to trigger an error message. If an invalid MS\_NO number is programmed, the process is aborted with the following runtime error message:

"Inadmiss. meas. sys."                      Error code : 91

### 9.3 Deactivating the reference path recording function, special function 30

Special function 30 can be used to deactivate the function for saving set point values.

The data stored in the ring memory are retained until special function 29 is called with the same kinematics number.

The special function provides the number of points stored until the time of deactivation in parameter "NO\_POINTS".

#### Syntax of special function 30

```
SPC_FCT : 30 = PKT_SPEI_AUS ( VALUE      INTEGER : KIN_NO
                               INTEGER   : NO_POINTS )
```

**KIN\_NO** - Kinematic number:

KIN\_NO indicates the kinematics whose coordinate values are no longer to be stored.

**NO\_POINTS** - Number of stored points (max. 50)

Check-back parameter of the special function. This indicates the number of valid points in the ring memory which have been stored since special function 29 was called.

#### Example (BAPS instructions):

```
;;CONTROL = RHO 3
;;KINEMATICS : (1=ROB_1,2=ROB_2)
;;ROB_1.JC_NAMES = A11,A12,A13,BN1
;;ROB_1.WC_NAMES = K11,K12,K13,BK1
;;ROB_2.JC_NAMES = A21,A22,A23,BN2
;;ROB_2.WC_NAMES = K21,K22,K23,BK2
```

- ;A11 is the 1st measuring system
- ;A12 is the 2nd measuring system
- ;A13 is the 3rd measuring system
- ;A21 is the 4th measuring system
- ;A22 is the 5th measuring system
- ;A23 is the 6th measuring system
- ;BN1 is the 7th measuring system
- ;BN2 is the 8th measuring system

```
INTEGER : NO_POINTS
PKT_SPEI_AUS(2, NO_POINTS)
```

In the example above the values of the second kinematics

("A21", "A22", "A23", "BN2" and "K21", "K22", "K23", "BK2")

are no longer stored as of the point of execution of the special function (PKT\_SPEI\_AUS). The BAPS variable "NO\_POINTS" is occupied by the special function with the number of stored points and can be used for reading out the points.

**9.4 Reading the reference path value, special function 31**

Special function 31 can be used to read stored setpoint values in joint coordinates and in world coordinates. The data are stored in a ring memory.

To get valid data, the special functions 29 and 30 must be called before calling special function 31. If special function 31 is used for several kinematics, a separate special function name must be declared for each of the kinematic (for example: PKT\_SPEI\_LES , PT1\_SPEI\_LES , PT2\_SPEI\_LES ...).

In order to always read valid data from the ring memory, note the check-back parameter "NO\_POINTS" of the special function 30. However, if an attempt is made to access an element in the ring memory which is not yet stored, the process is aborted and the following runtime error is issued:

'Point is invalid' Error code : 90

**Declaring special function 31**

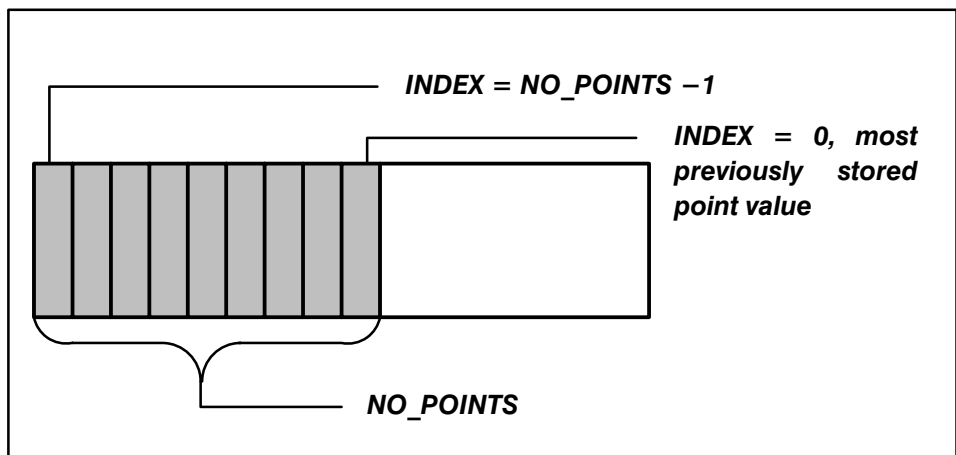
```
SPC_FCT : 31 = PKT_SPEI_LES( VALUE INTEGER : KIN_NO  
                             VALUE INTEGER : INDEX  
                             JC_POINT      : @POSITION  
                             POINT        : POSITION )
```

**KIN\_NO** - Kinematics number:

If the special function is called, "KIN\_NO" indicates the kinematics whose coordinate values are returned. The coordinate values of the belts which belong to these kinematics are also available.

**INDEX** - Ring memory index

INDEX indicates which ring memory entry should be read.



**POSITION** – World coordinate values

The world coordinate values of the kinematics specified in KIN\_NO are in POSITION.

**@POSITION** – Joint coordinate value

The joint coordinate values of the kinematics specified in *KIN\_NO* are in *@POSITION*.

**Example (BAPS instructions):**

```

;;CONTROL = RHO 3
;;KINEMATICS : (1=ROB_1,2=ROB_2)
;;ROB_1.JC_NAMES = A11,A12,A13,BN1
;;ROB_1.WC_NAMES = K11,K12,K13,BK1
;;ROB_2.JC_NAMES = A21,A22,A23,BN2
;;ROB_2.WC_NAMES = K21,K22,K23,BK2

;A11 is the 1st measuring system
;A12 is the 2nd measuring system
;A13 is the 3rd measuring system
;A21 is the 4th measuring system
;A22 is the 5th measuring system
;A23 is the 6th measuring system
;BN1 is the 7th measuring system
;BN2 is the 8th measuring system

;***** Declaration for kinematics 1 *****
SPC_FCT : 31 = PKT_SPEI_LES ( VALUE INTEGER : KIN_NO
                               VALUE INTEGER : INDEX
                               ROB_1.JC_POINT : @POSITION_1
                               ROB_1.POINT    : POSITION_1 )

;***** Declaration for kinematics 2 *****
SPC_FCT : 31 = PK2_SPEI_LES ( VALUE INTEGER : KIN_NO
                               VALUE INTEGER : INDEX
                               ROB_2.JC_POINT : @POSITION_2
                               ROB_2.POINT    : POSITION_2 )

;;KINEMATICS = ROB_2
POINT      : WC_POSITION
JC_POINT   : @JC_POSITION
PK2_SPEI_LES(2,1,@JC_POSITION,WC_POSITION)

```

In the example above the values of the second kinematics ("A21", "A22", "A23" and "BN2") are returned: in *WC\_POSITION* the world coordinates, and in *JC\_POSITION* the joint coordinates. These values are derived from the previous save operation.

## 9.5 Program examples (partial)

```
::CONTROL = RHO 3
::KINEMATICS : (1=ROB_1,2=ROB_2)
::ROB_1.JC_NAMES = A11,A12,A13,BN1
::ROB_1.WC_NAMES = K11,K12,K13,BK1
::ROB_2.JC_NAMES = A21,A22,A23,BN2
::ROB_2.WC_NAMES = K21,K22,K23,BK2
```

### PROGRAMMAIN

EXTERNAL:MOVEMENT,BELTSTOP,BELTSTART

```
::CONTROL = RHO 3
::KINEMATICS : (1=ROB_1,2=ROB_2)
::ROB_1.JC_NAMES = A11,A12,A13,BN1
::ROB_1.WC_NAMES = K11,K12,K13,BK1
::ROB_2.JC_NAMES = A21,A22,A23,BN2
::ROB_2.WC_NAMES = K21,K22,K23,BK2
```

### PROGRAM MOVEMENT

ROB\_2.POINT: POS1,POS2,POS3,POS4

```
SPC_FCT : 29 = PKT_SPEI_EIN    VALUE INTEGER : KIN_NO
                                VALUE INTEGER : MS_NO
                                VALUE REAL      : DISTANCE )
```

BEGIN

```
::KINEMATICS = ROB_2
```

.

.

.

```
PKT_SPEI_EIN(2.8.13)
```

.

.

.

```
MOVE LINEAR POS1           ;The program must be structured in such a way that
MOVE LINEAR POS2           ;operations continue at the desired position
MOVE LINEAR POS3           ;after a belt stop
MOVE LINEAR POS4           ;when a restart is executed.
```

.

.

.

### PROGRAM\_END

```

;;CONTROL = RHO 3
;;KINEMATICS : (1=ROB_1,2=ROB_2)
;;ROB_1.JC_NAMES = A11,A12,A13,BN1
;;ROB_1.WC_NAMES = K11,K12,K13,BK1
;;ROB_2.JC_NAMES = A21,A22,A23,BN2
;;ROB_2.WC_NAMES = K21,K22,K23,BK2

PROGRAM BELTSTOP

INPUT : 9 = BS

ROB_2.POINT: POS1,POS2,POS3,POS4,BNDPOS,WC_POSITION,SAVE_POS,DELTA
ROB_2.JC_POINT: @JC_POSITION

SPC_FCT : 30 = PKT_SPEI_AUS ( VALUE INTEGER : KIN_NO
                               INTEGER : NO_POINTS )

SPC_FCT : 31 = PKT_SPEI_LES ( VALUE INTEGER : KIN_NO
                               VALUE INTEGER : INDEX
                               ROB_2.JC_POINT : @POSITION
                               ROB_2.POINT : POSITION )

EXTERNAL : MOVEMENT
REAL : DIFF
INTEGER : NO_K1_PNTS, NO_K2_PNTS

BEGIN
;;KINEMATICS = ROB_2
.
.
    WAIT UNTIL BS=1 ;WAIT UNTIL BELT STOP SIGNAL
    STOP MOVEMENT ;ABORT MOVEMENT PROGRAM
    PKT_SPEI_AUS(2, NO_K2_PNTS) ;DEACTIVATE SAVE
    MOVE LINEAR SAFE_POS ;MOVE KINEMATICS TO SAFE AREA
    IF NO_K2_PNTS < 3 THEN ERROR_01 ;AT LEAST 3 POINTS STORED ?
    WAIT 1 ;WAIT UNTIL CONVEYOR IS STATIONARY

    BNDPOS = ROB_2.POS ;CURRENT ROBOT POSITION
    PKT_SPEI_LES(2,2,@JC_POSITION,WC_POSITION)
    DIFF = BNDPOS.BK2 - WC_POSITION.BK2 ;RESUME BELT AFTER STOP
    DELTA = SAFE_POS
    DELTA.K21 = DELTA.K21 + DIFF
    MOVE LINEAR DELTA ;MOVE IN BELT DIRECTION
    WC_POSITION.K21 = WC_POSITION.K21 + DIFF
    MOVE LINEAR WC_POSITION ;RESET OLD ORIENTATION AND DISTANCE
                                ;OF THE ROBOT TO THE BODY
.
.
PROGRAM_END

SR ERROR_01
.
.
SR_END

```

```
::CONTROL = RHO 3
::KINEMATICS : (1=ROB_1,2=ROB_2)
::ROB_1.JC_NAMES = A11,A12,A13,BN1
::ROB_1.WC_NAMES = K11,K12,K13,BK1
::ROB_2.JC_NAMES = A21,A22,A23,BN2
::ROB_2.WC_NAMES = K21,K22,K23,BK2
```

PROGRAM BELTRUN

```
INPUT      :    11 = BA
EXTERNAL   :    MOVEMENT
```

ANFANG

```
.
.
      WAIT UNTIL BA=1                ;WAIT UNTIL BELT START SIGNAL
      BEGIN MOVEMENT                ;START MOVEMENT PROGRAM
.
.
PROGRAM_END
```



## 10 Limiting axis speeds in linear mode

Randomly high axis speeds may occur for linear or circular movements if path speeds are firmly established. This is caused by the coordinate transformation, particularly around singularities (for example, extended position when a SCARA robot is used).

With the new option, the resulting axis speeds for linear movements can be monitored

- (a) only in manual mode or
- (b) in manual and automatic mode.

Monitoring in manual mode is with the fast joint coordinate-jog speed set in machine parameter **P114**, and in automatic mode with the default maximum speeds set in machine parameter **P103**.

If an axis exceeds its default maximum value, an **Interpolator Stop** is triggered (warning), and all axes move at reduced speed until they are no longer in the critical speed range. Movements are not aborted.

To avoid sudden speed deviations on the one hand and to keep path deviation to a minimum on the other, a precise interpolation operation is executed for the critical axis (or axes). This reduces the speed of **a l l** axes if the speed of a single axis is excessive. The extent and duration of the reduction is yielded by the ratio of the default maximum axis speed and the calculated set axis speed.

### Example:

Maximum speed : 15 mm/clock (normed to mm/clock)

Set value : 40 mm/clock

- $40/15 = 3$  (integer division); 3 clocks are required to cover the specified path without exceeding the maximum speed
- The same path is specified at the set value in each of the 3 clocks, i.e.,  $40/3 = 13.33$  mm/clock
- This speed reduction to 33.33% of the set value is performed for **all** the axes of the kinematics.

**ATTENTION !**

- If the default maximum speed is exceeded, neither the default or programmed path speed nor the programmed path (in world coordinates) can be maintained.
- "Feed enable" and "Feed hold" then act only by means of the normal interpolator. This means that values determined by the interpolator may be output as reduced values if the axis monitoring function is active. The result here is that though the time until standstill is increased, the path covered nevertheless remains independent of the axis monitoring function.
- Due to certain internal control features, the last part of the interpolation of a traverse block cannot be reduced in automatic mode. The resulting unmonitored path is yielded in [mm/Clock] by the programmed speed and the set clock time (machine parameter P5).

## 10.1 Machine parameters for limiting axis speed

Axis speed monitoring is set with parameter **P306**, subquery 6:

Monitoring axis WC

- 0 = No monitoring
- 1 = Monitoring with movement abort --> 'Ax-Velocity exceeded'
- 2 = Monitoring only in manual mode with axis speed limitation --> 'Interpolator Stop'; No monitoring in automatic mode
- 3 = Monitoring in manual and automatic mode with axis speed limitation --> 'Interpolator Stop'

## 10.2 Previous axis speed limits

Previous software versions (older than TO04) only offer the following alternatives in the control with machine parameter P306, subquery 6 "Monitoring axis WC":

- the resulting axis speed is not monitored at all, which most of the time leads to a servo error (READY2 was opened, meaning that the control must restart afterwards)
- the resulting axis speed is monitored with the maximum axis speeds set in machine parameter P103 and, if the maximum speed is exceeded, a movement abort is forced with the runtime error message 'Ax-Velocity exceeded' (READY2 closed).

## 11 Manual axes as endless axes

For endless axes as of software version TO04, a distinction is made between main axes and manual axes. The parameter P303 (axis type) has been upgraded to be upstream compatible.

P303 =    0 normal main axis  
          1 normal manual axis  
          2 currently no meaning  
          3 endless axis as main axis  
          4 endless axis as manual axis

### Example

One kinematic with 5 endless axes, all with a modulo value of 360 (P311). These are 1 master and 4 slave axes:

The above problem can be solved as follows:

- Zero is set as the robot type (P306: RTYP=0) and the master axis must be the first axis of the kinematics.
- The master axis is declared as the main axis (P303=3) and the slave axes as manual axes (P303=4).

Axis 1 : P303 = 3 endless axis as main axis  
Axis 2 : P303 = 4 endless axis as manual axis  
Axis 3 : P303 = 4 endless axis as manual axis  
Axis 4 : P303 = 4 endless axis as manual axis  
Axis 5 : P303 = 4 endless axis as manual axis

- If the path is traversed LINEAR at the specified path speed, the speeds of the slave axes adjust to the speed of the master axis.

V=500

```
MOVE LINEAR VIA ( 20, 0, 0, 360, 0 )
MOVE LINEAR VIA ( 90, 120, 0, 0, 0 )
MOVE LINEAR VIA ( 105, 150, 0, 0, 0 )
MOVE LINEAR VIA ( 190, 300, 160, 0, 180 )
MOVE LINEAR VIA ( 200, 320, 180, 160, 240 )
MOVE LINEAR VIA ( 220, 360, 260, 200, 360 )
MOVE LINEAR VIA ( 280, 0, 360, 260, 0 )
MOVE LINEAR VIA ( 360, 0, 0, 320, 0 )
```

The master axis (axis 1) moves through the entire movement cycle at the constant speed of 500 [degrees/sec]; the speeds of the slave axes are slaved to this speed.

**General notes on previous methods of operation :**

Previously, it was only possible in the controller to declare endless axes as main axes.

This can lead to problems, particularly, if a endless master axis is to move at constant speed and several endless slave axes must execute additional movements in synchronization with the master axis. In this concrete example, undesirable changes to the master axis speed resulted as all the endless axes were included in the speed calculation.

## 12 The BAPS2 instruction ASSIGN

The BAPS2 instruction ASSIGN makes it possible to change the names of files while the program is running.

In previous software versions, the name of the file variable is also the name of the file in the control (the extension ".DAT" is automatically added).

In the following program section, the file would receive the name CADDATEN.DAT.

```
...  
FILE : CADDATEN  
...  
...  
...  
WRITE_BEGIN CADDATEN  
...
```

The BAPS2 instruction ASSIGN can be used to change the name of the file by the user program.

This allows more flexible use of file operations.

### 12.1 Syntax ASSIGN instruction

The syntax is as follows:

<p><b>SYNTAX:</b></p> <p><b>ASSIGN</b> <i>File_Variable</i> , <i>Text_Expression</i></p>
--

The following applies:

<i>File_Variable</i>	:	BAPS name (max. 12 characters, when the program is running only the first 8 characters are evaluated, or an error message "file name n. allowed" is created)
<i>Text_Expression</i>	:	Text constant or variable of type TEXT

## 12.2 ASSIGN instruction

The Assign instruction allows (almost) any file name extension.

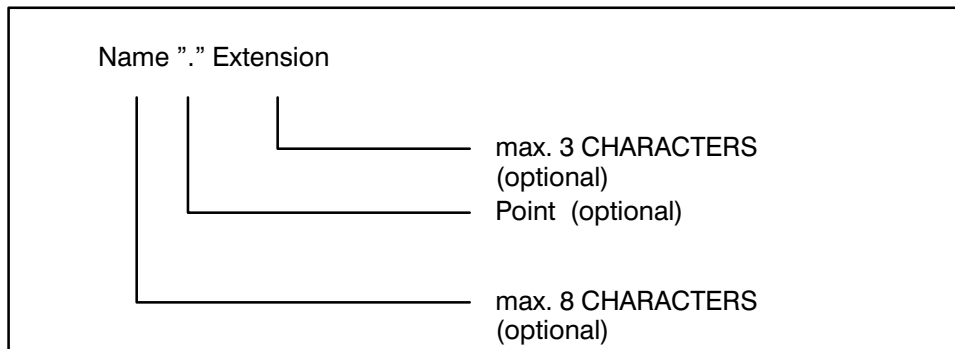
A name is valid if :

- The first character of the name is a letter;

The following applies for the file's extension:

- The first character of the extension is a letter, or the extension consists of three blanks,
- Only the characters [ "0" .."9" , "A" .."Z" , "\_" ] are used,
- Only upper case letters are used!

The structure of a file name must satisfy the following requirements :



- Umlauts (modified vowels) are prohibited.
- DAT is the default extension if no extension (and no point) is specified. If a point is added, the extension becomes " " (three blank spaces).

## 12.3 Compatibility

The ASSIGN instruction can be implemented downstream. As before, the name of the file variable is automatically the name of the file in the control (with the extension ".DAT") for implicit opening at the beginning of the program.

## 12.4 Using ASSIGN on standard channels

The standard channels can be redefined! Standard channels in BAPS2 are predefined names for communication interfaces. The following are standard channels:

- PHG
- V24\_1 to V24\_4
- SER\_1 to SER\_4

## 12.5 Runtime conditions

The ASSIGN instruction can only be used on closed files!

It may be necessary to close a file intended for the ASSIGN instruction with the close instruction.

Using the ASSIGN instruction on an open file results in a runtime error if error monitoring is not deactivated with

**;;FILE\_ERROR -**

The most previously assigned file name is retained and is not changed when the file is closed (as before).

## 12.6 Compiler instruction FILE\_ERROR

If an error occurs during an ASSIGN instruction operation, the process is aborted with a runtime error message. In certain cases this is undesirable. In such cases, the compiler instruction FILE\_ERROR is inserted.

A process abort can be prevented with the compiler instruction **FILE\_ERROR**. The following table applies:

Syntax	Effect
<b>;; FILE_ERROR -</b>	no program abort
<b>;; FILE_ERROR +</b>	program abort
<b>;; FILE_ERROR</b>	program abort

### Comments:

- The compiler instruction is only permissible at the beginning of the program and may occur only once.
- It is valid for the entire program. Main process values are taken for internal subprocesses; for external subprocesses (generated by the start instruction), program abort is the default.
- Errors which occur are stored internally in a buffer and can be called using the standard function CONDITION.

Accessing an error with the standard function CONDITION deletes the internal buffer (this means that it may be necessary to store the value in the BAPS program).

- If an error occurs the most previously assigned file name is retained.

If the compiler instruction FILE\_ERROR is not programmed, an allocation error leads to a process abort, and an error message is issued.

## 12.7 Restrictions on selecting file names

If no valid file name is provided, a runtime error is generated at program runtime (for ;;FILE\_ERROR +).

## 12.8 Application example for ASSIGN

```
;;CONTROL = RHO 3
;;FILE_ERROR -
PROGRAM ASSIGN
FILE      : DatVar
TEXT      : DatName, Oneline
INTEGER   : BEGIN
WRITE CLS                               ; Clear PHG display
WRITE ' file name : ',..                ; Read file name
READ DatName
ASSIGN DatVar, DatName
IF CONDITION ( DatVar ) = 0THEN
BEGIN                                     ; no errors
    READ_BEGIN DatVar
    READ DatName, Oneline
    WRITE Oneline
    CLOSE DatVar
END
ELSE
BEGIN
    WRITE 'File allocation error !'
    HALT
END
ASSIGN DatVar, 'ASSIGN.QLL'
IF CONDITION ( DatVar ) = 0THEN
BEGIN                                     ; no errors
    READ_BEGIN DatVar, 10
    READ DatName, Oneline
    WRITE PHG, Oneline
    CLOSE DatVar
END
ELSE
BEGIN
    WRITE 'File allocation error ! ("ASSIGN.QLL")'
    HALT
END
PROGRAM_END
```



## 13 Calling operating system functions from the BAPS2 user programs

Using the special function 4, rho 3 operating system functions (commands) which previously could only be executed by means of the PHG can now be activated from within BAPS2 user processes.

### 13.1 Operating system commands

The following operating system functions can currently be used:

COMPILE	QLL program file
COPY	any file
DELETE	any file
START	user process (IRD file)
STOP	active user processes

### 13.2 Declaration part

Special function 4 must be declared in the declaration part of a BAPS program.

#### Declaring special function 4

```

SPC_FCT:    4 = COMMAND (VALUE           INTEGER: CMD_ID
                                     TEXT:   SOURCE,TARGET
                                     INTEGER: STATUS      )
    
```

*COMMAND* :

Name which is used to call the special function. The user can assign any name, with the exception of the key words reserved for BAPS.

*CMD\_ID* :

Indicates which operating system command should be executed. Currently available:

- 1 = COMPILE
- 2 = COPY
- 3 = DELETE
- 4 = START
- 5 = STOP

*SOURCE,TARGET* :

Contain necessary parameters for the respective function, for example, file names. If only one parameter is required, the second remains unchanged meaning that in this case the second parameter can be specified twice. File names are stored in the parameters in the form of an ASCII character string, including the file name extension.

*STATUS :*

Provides the results of the executed function in the form of an INTEGER.

- 0 No error, command processed correctly.
- 1 Illegal command transferred to special function.
- 2 Error in target file name while copying  
or  
Error in name of the file to be compiled.  
or  
Error in name of the file to be deleted.
- 3 File name extension is not "QLL" while compiling.
- 4 Error in QLL file name.
- 5 File to be deleted or target file not still open during copying.
- 6 Error while copying, for example insufficient memory
- 7 Compiler is already active when call executed.
- >0 Sum of all compiler errors and compiler warnings  
  
or  
error number of the error which occurred when the specified process started.
  
- 1051 User process already exists.
- 1092 Selected program not available.
- 1073 Program can only be run as an external subroutine
- 1057 File repeatedly open.
- 1085 The point file is illegal for the  
selected process.
- 1112 The selected IRD file or the respective  
PNT file could not be sequentialized  
because of insufficient memory.  
or
- 1079 A subprocess attempts to  
stop its own main process.

### 13.3 Commands

Use of the special function is described in detail below.

#### 13.3.1 Compiling BAPS programs

Calling "Special function 4" with CMD\_ID=1 calls the BAPS2 compiler in the rho 3.

The instruction line in a BAPS program appears as follows:

**COMMAND (1,FILE\_NAME,TEMP\_NAME,RESULT)**

The file whose name is in the variable FILE\_NAME is compiled. The result, that is, 0 or a value <>0, is available in the RESULT variable in the BAPS program.

In addition, the compiler issues a corresponding message on the PHG, but without waiting for a key to be pressed on the PHG.

If the operator wants to prevent the output of compiler messages on the PHG, the RC input 35 "Output of system messages on the PHG disabled", PIC address O4.3, must be set to "1". The parameter TEMP\_NAME is not evaluated and not influenced, meaning that FILE\_NAME can also be used as a parameter.

#### 13.3.2 Copying files

In special function 4, CMD\_ID=2 can be used to copy files.

During runtime, the special function checks whether the file name extensions of the source and target file are permissible. Error states are returned by means of CONDITION.

The instruction line in a BAPS program appears as follows:

**COMMAND (2,SOURCE\_NAME,TARGET\_NAME,CONDITION)**

The variable SOURCE\_NAME contains the name of the file to be copied. TARGET\_NAME contains the file name to which the copy operation is intended. If the target file already exists, it is deleted by the copy operation. A CONDITION message is issued only if the target file is open for writing or reading.

Permissible file name extensions are:

Source	Target
QLL	QLL,ERR,DAT
DAT	QLL,ERR,DAT
PKT	PKT
SYM	SYM
IRD	IRD

### 13.3.3 Deleting files

In special function 4, CMD\_ID=3 can be used to delete files. The instruction line in a BAPS program thus appears as follows:

**COMMAND (3,FILE\_NAME,TEMP\_NAME,CONDITION)**

FILE\_NAME contains the name of the file to be deleted; TEMP\_NAME is not used and not effected. CONDITION provides the result of the delete operation.

Files intended for deletion must not have been opened for reading or writing by another BAPS process; i.e., they must not be open.

### 13.3.4 Starting processes

In special function 4, CMD\_ID=4 can be used to start processes.

**COMMAND (4,FILE\_NAME,TEMP\_NAME,CONDITION)**

FILE\_NAME contains the name of the program to be started. The name must not contain a file name extension (i.e., without ".IRD"). TEMP\_NAME is not used and is not effected. CONDITION provides the result of the start operation.

### 13.3.5 Stopping processes

In special function 4, CMD\_ID=5 can be used to stop processes.

**COMMAND (5,FILE\_NAME,TEMP\_NAME,CONDITION)**

FILE\_NAME contains the name of the program to be stopped.

The name must not contain a file name extension (i.e., without ".IRD"). TEMP\_NAME is not used and is not effected.

CONDITION provides the result of the stop operation.

### 13.4 Example

The following example shows how special function 4 is used.

```

;;PROCESS_KIND=PERMANENT           ;Only so that the program
                                   ;can always be run

;;CONTROL=RHO3
PROGRAMM SPCFCT4

SPC_FCT      : 4 = COMMAND (VALUE INTEGER: CMD_ID
TEXT         : SOURCE,TARGET
INTEGER      : CONDITION)
CONST       : COMPILE=1; COPY=2; DELETE=3,
              START=4, STOP=5

TEXT        : Q_TXT, T_TXT, Z_TXT
INTEGER     : RESULT,I
OUTPUT      : 1=DISABLE_DISP
INPUT       : 1=PROG_START, 2=PROG_STOP
BINARY      : STARTED

BEGIN

DISABLE_DISP=1                       ;System messages can be disabled
                                       ;on the PHG

Q_TXT = 'BASIS.QLL'
T_TXT = 'SOURCE.DAT'
Z_TXT = 'TARGET.QLL'
COMMAND(KOPIERE,Q_TXT,T_TXT,ERGBNIS)   ;Copies BASIS.QLL to SOURCE.DAT
IF RESULT < 0 THEN JUMP FEHLER_BEH
                                       ;Logic can appear here which can be used
                                       ;to modify the SOURCE.DAT.

COMMAND(COPY,T_TXT,Z_TXT,RESULT)       ;Copies SOURCE.DAT to TARGET.QLL.
IF RESULT < 0 THEN JUMP FEHLER_BEH
COMMAND(COMPILE,Z_TXT,Z_TXT,RESULT)    ;Compiles TARGET.QLL
IF RESULT < 0 THEN JUMP ERROR_RT
COMMAND(DELET,T_TXT,T_TXT,RESULT)      ;Deletes SOURCE.DAT
IF RESULT < 0 THEN JUMP ERROR_RT

```



### **13.5 Restrictions, error treatment**

- When special function 4 is used, a plausibility check of the file name extension is not performed when files are copied.
- Names and file name extensions are not checked for validity.
- The compiler must not be called simultaneously by several processes. Error code "-7" is issued if this is attempted.

## 14 Belt type selection, special function 21

Three types of belt synchronization are available in the rho 3 starting with software version TO04.

Types of belt synchronization:

- 1 : Standard belt synchronization (as programmed in BAPS)
- 2 : Belt synchronization without parallel belt traversing capability
- 3 : Belt-synchronous "cam interpolation"

After control start-up, the default belt synchronization is type 1.

**Special function 21** is used to switch between the type of belt synchronization in a BAPS2 program. The selected setting then applies to **all** processes until the synchronization type is changed with this function or the control is restarted.

### 14.1 Declaring special function 21

```

SPC_FCT : 21 = BELT_TYPE (VALUE INTEGER : BELT_NO
                        VALUE INTEGER : TYPE_BELT)
    
```

*BELT\_NO* : Number of the belt for which the value is set (1..8).

*TYPE\_BELT* :

- 1 = Standard belt synchronization  
see also "rho 3 BAPS2 programming instructions"
- 2 = Belt synchronization without parallel belt traversing capability
- 3 = Synchronous belt "Cam interpolation"  
see also description of option.

For belt synchronization types 2 and 3, it is additionally possible to set the belt synchronization tolerance in the 10's place.

Standard setting: 5 x distance / interpolation cycle

Example: 72 = 7 x distance / interpolation cycle  
2 = belt synchronization without belt parallel axis

Input range (ten's place): 1..12

Permissible values:

- Axis 1: 1
- Axis 2: 2, 12, 22, 32, ..., 122
- Axis 3: 3, 13, 23, 33, ..., 123



## 14.2 Using special function 21

A BAPS2 model program is provided in the following which illustrates the use of the special function.

**PROGRAM** SPC21

; Declaration of the special function

**SPC\_FCT: 21 = BELT\_TYPE (      **VALUE INTEGER** : BELT\_NO  
                                  **VALUE INTEGER** : TYPE\_BELT )**

**BEGIN**

.

; Selection of the type of belt by using the special function

**BELT\_TYPE ( 1, 2)**

.

.

**PROGRAM\_END**

### 14.3 Notes and error messages regarding special function 21

If a value outside the range 1..P501 (number of belts) is programmed as BELT\_NO, the following runtime error message occurs:

**'Inadmiss. Belt-No '                      Code 93**

If a value outside the admissible range (cf. chapter 14.1) is programmed for TYPE\_BELT, the following runtime error message occurs:

**'wrong belt-kind/-tol'                      Code 92**

The belt type remains active until special function 21 is called again (for the same belt).

If the control starts again, the default value is "1".

**This special function is not required if a belt is operated in belt synchronization type 1.**

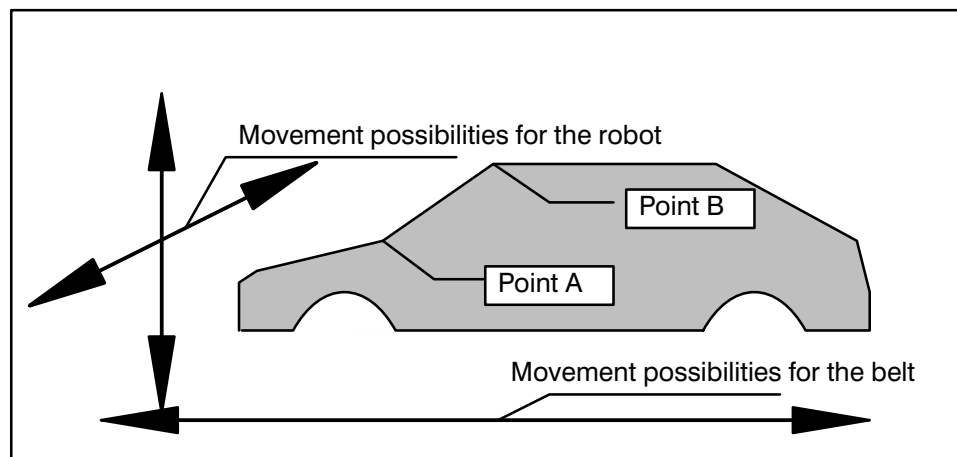
If a belt is operated with only one belt synchronization type, we recommend calling this special function in the program **INIT**.

If a belt is operated with different belt synchronization types, we recommend calling this special function before each **SYNCHRON** command.

## 15 Belt synchronization without the robot being able to move parallel to the belt

The following chapter describes the function, the parameters and programming of belt synchronization type 2. Select the belt synchronization type with special function 21 (see chapter 14).

In this type of belt synchronization, the robot's kinematics do not allow parallel belt movements.



The robot is moved so that it and the belt reach the programmed end point at the same time.

The programmed speed is not considered here. It depends on the belt speed and on the programmed lead angle which results from the position of the interpolator points and their belt component.

The programmed acceleration is considered in this type of synchronization. On the one hand, considering the programmed acceleration reduces robot wear, but on the other hand leads to speed-dependent deviations from the programmed path.

## 15.1 Operating conditions for belt synchronization type 2

The following conditions apply for the synchronization part of the program of belt synchronization type 2:

- If kinematics move according to belt synchronization, no other processes can be active with these kinematics. A process must wait until the synchronous part is concluded.
- If runtime errors occur, the READY contact is opened in order to stop the belt itself, if required. The runtime error **EMERGENCY STOP – Input**, code 528, is additionally shown.
- If kinematics movements are hindered by Feed hold, etc., the program is aborted and the kinematics runtime error **BS–tolerance too big**, code 944..953, is issued.
- The interpolation type PTP is prohibited. In case of PTP, the program aborts with the runtime error message 'PTP not allowed for belt synchronous or sensor movement', code 23.
- For the first belt synchronous movement, a pause is made until the belt has reached the belt value which was programmed during the last traverse block before the belt synchronization part.
- If before the first belt synchronous movement the belt moves opposite the programmed direction, the program is aborted and the runtime error code **wrong belt direction**, code 928..937 is issued.
- Deviation from the programmed belt position is checked at the traverse block transitions. If this deviation is greater than the path covered by the belt in a specific number of interpolation cycles (5 are standard), the program aborts and issues the error message (which depends on the kinematics):

**'BS–tolerance too big' code 944..953.**

The scope of the tolerance range can be changed by calling

**SPC\_FCT : 21 = BELT\_TYP (VALUE INTEGER : NO VALUE INTEGER : TYP)**

(cf. chapter 14).

- The belt difference between two sequential traverse blocks must not be 0 (for example, MOVE\_REL (0, 0) !!) Otherwise, the program aborts and issues the runtime error '**beltdistance is zero**', code 89.
- The belt difference must have the same sign for each traverse block within the belt synchronization program section; otherwise, the error message '**wrong belt–direction**', code 928..937 is issued.
- A MOVE\_REL as the first traverse block in the SYNCHRON part of the program is allowed.
- Here, the commands SYNCHRON and SYNCHRON\_END do not stop block preparation.
- Kinematic runtime errors and '**EMERGENCY STOP – INPUT**' are displayed in the process display as '**other errors**'.

## 15.2 Special function for parameterisation of the belt characteristics

In the following, the parameters are described which are necessary to adapt the type of synchronization to the specific characteristics of the conveyor system. Parameterisation is with special function 15.

The thresholds for recognizing the belt status "Belt stopped" and "Belt running" and the times tolerated before error messages are issued are programmed by means of special function 15.

<b>SPC_FCT : 15 = BELT_PARAM (VALUE INTEGER :</b>	<b>BELT_NO</b>
<b>VALUE REAL :</b>	<b>V_BLT_STOP</b>
<b>VALUE REAL :</b>	<b>V_BLT_RUN</b>
<b>VALUE INTEGER :</b>	<b>TIME_MS )</b>

*BELT\_PARAM* : Freely assignable function name

*BELT\_NO* : Number of the belt for which the values are set (1..8).

*V\_BLT\_STOP* : Threshold for the belt speed in mm/sec at which "Belt stopped" is recognized.

*V\_BLT\_RUN* : Threshold for the belt speed in mm/sec at which "Belt running" is recognized.

*TIME\_MS* : Time in milliseconds in which error messages are tolerated.

Rounding is to the whole interpolation cycle.

**Example:**

**PROGRAM SPC15**

```
SPC_FCT: 15 = BELT_PARAM (  VALUE INTEGER :  BELT_NO
                           VALUE REAL :    V_BLT_STOP
                           VALUE REAL :    V_BLT_RUN
                           VALUE INTEGER :  TIME_MS  )
```

**BEGIN**

```
BELT_PARAM (1, 10.0, 50.0, ROUND ( 160.00 / 66.6 ) *1000.0 )
```

**PROGRAM\_END**

In the example, the names used for the special function and its parameters are, as with other special functions as well, merely place holders. They can be renamed without effecting a function's outcome. However, the types of parameters must be declared in the manner and order provided above.

To change a belt's parameters the kinematics of that belt are briefly occupied. If another process is being executed with these kinematics synchronous to a belt, the belt parameters are, at the earliest, changed at the end of the synchronous belt program section.

If the control starts again, the following values are default values:

"Belt stopped"	= 0.005 mm/interpolation cycle
"Belt running"	= 0.005 mm/interpolation cycle
Time	= 0 milliseconds

The belt parameters remain active for all processes until special function 15 is called again (for the same belt).

If the belt drifts below the threshold for "Belt stopped", the monitoring functions are deactivated.

Belt speed is calculated from the number of measuring system pulses per interpolation cycle. The result is that the belt speed is graduated.

## **15.3 Belt stop/belt start for belt synchronization type 2**

If severe changes in belt speed occur in conjunction with Belt stop/Belt start when using the function for belt synchronization without parallel belt axes, there may be cases in which a program will abort with different error messages.

These error messages occur because when the robot's acceleration and braking phase is considered during a reduced traverse path, either the position corresponding to the belt value cannot be reached in time ("Belt synchr. tolerance too big"), or an attempt was made to specify a speed in space which exceeds a maximum axis speed.

The program abort caused by such error messages can be avoided by temporarily allowing a deviation for the robot (for belt stop/belt start) which is greater than the programmed tolerance.

To recognize a belt stop or belt start, the control requires threshold values for the belt status "Belt stopped" and "Belt running".

It is assumed that the belt runs only in the programmed direction. The threshold value for "Belt stopped" can be easily used for suppressing the error message "wrong belt direction" which is triggered by the hunting oscillation of the belt position when it is at standstill if a program waits for a belt to exceed the start value.

### **15.3.1 Meaning of the threshold value for the belt status "Belt stopped"**

The belt status "Belt stopped" is recognized when the absolute value of the belt speed is less than the specified threshold value. In this state, the robot is braked to a speed of 0 and the monitoring functions "wrong belt direction" and "BS tolerance" are deactivated.

Only when this threshold is exceeded is a speed calculated for the robot and an attempt is made to have the robot and the belt reach the interpolation point at the same time.

The monitoring function "Belt synchr. tolerance too big" is deactivated for a programmable time starting at that point in time when the threshold was last exceeded.

If the belt nevertheless moves opposite the programmed direction, the program is aborted immediately after the threshold is exceeded.

### 15.3.2 Meaning of the threshold value for the belt status "Belt running"

The belt status "Belt running" is recognized when the absolute value of the belt speed is greater than the specified threshold value. In this state, the belt speed is averaged for (up to 10) interpolations increments and the speed specification for the robot is calculated from the averaged speed. If the unaveraged belt speed falls below this threshold value, belt speed averaging is deactivated and the monitoring function "Belt synchron. tolerance too big" is deactivated for a programmable interval.

If the belt does not stop completely but continues at a speed below the threshold for "Belt running", the monitoring function "Belt synchron. tolerance too big" is automatically activated at the end of the set time interval.

The monitoring function for "Belt synchron. tolerance too big" remains deactivated only if the belt speed falls below the threshold for "Belt stopped".

### 15.4 Bypassing the error message "Ax velocity exceeded"

The limiting function for the path speed can only roughly limit the axis speed which is yielded by the transformation. For this reason, this error message can be bypassed so that instead of axis speed monitoring by means of machine parameter **P306**, subquery 6, the value 3 can be used to set limits for the axis speed (in automatic mode for the values in machine parameter **P103** and in manual mode for the values in machine parameter **P114**).

Set axis speeds which exceed the maximum axis speed are divided into two or more interpolation intervals (50% .. 100% of the maximum value) and the interpolator is halted for the set duration.

Limiting the axis speed leads to an extension of the movement time and thus to a greater deviation of the robot position from the path.

### 15.5 General notes on synchronization type 2

The coupling factors (P503) are not considered for this type of belt synchronization.

The change in speed must be only small enough that it can be followed by the robot.

This type of belt synchronization is activated by means of special function 21 and can thus be changed according to specific user requirements.

The speed estimated in the first interpolation cycle is repeated in each cycle. In this way, accuracy problems do not result, even in case of longer blocks.



## 15.6 Restrictions due to this type of belt synchronization in conjunction with other options

If this type of belt synchronization is active and the kinematics are stopped with either Feed hold or Feed allow, the runtime error

**'BS tolerance too big' code 944..953**

is generated and at the present, the program remains stationary and the error 'other errors' is issued.

A wait condition in the belt-synchronous program section cannot be specified with the WAIT command. A subsequent traversing movement would recognise 'BS tolerance too big'. Instead, a traversing movement has to be programmed which only changes the belt coordinate, e.g. MOVE-REL (0,100).

Currently, errors which occur are not displayed if they are process-related.

### NOTE:

**The "spatial passing" function cannot be used in conjunction with this type of belt synchronization.**

**15.7 Program example**

```
1      ;;KINEMATICS : 1 = rob1
2      ;;rob1.WC_NAMES = k1, bnd
3      ;;rob1.JC_NAMES = k1, bnd
4
5      PROGRAM beltyp2
6
7      SPC_FCT: 21 : belt_typ (          VALUE INTEGER:belt_no
8                                          VALUE INTEGER:typ_belt)
9
10     rob1.BELT :          501 = CONVEYOR
11     INPUT :          1 = CONVEYOR_SYNC
12     BEGIN
13
14     belt_typ ( 1, 2 )
15     A = 100 , V = 500
16     MOVE LINEAR VIA (0.50)
17     WAIT UNTIL CONVEYOR_SYNC = 1
18     WAIT 0.05          : may be necessary for synchronization scan
19                       : time of PLC and controller cycle
20     SYNCHRON rob1 CONVEYOR
21
22     MOVE LINEAR ( 150, 200 ) ;
23     MOVE LINEAR ( 150, 240 ) ;corresponds to wait until
24                               ;CONVEYOR >= 240
25     MOVE LINEAR ( 0,360 )
26     SYNCHRON_END
27     .
28     .
29     PROGRAM_END
```

Explanation of the above program example:

The BAPS2 keywords are in uppercase letters, other freely selectable names are in lowercase letters.

Lines 7 and 8 contain the declaration of special function 21 for changing over the belt synchronization types. It is called up in line 14.

The belt counter is reset via RC input no. 376, which is additionally available in the BAPS program as input CONVEYOR\_SYNC.

In line 16 rob1 is moved to position 0. The belt value of 50 has no meaning for this movement, however, it is important for the first belt – synchronous traverse block in line 22. There, rob1 waits with its movement until the belt has reached the value 50. If the belt moves away from its initial value, i.e. in negative direction in this example, the program is aborted with the error message

**'BS wrong belt direction'                      code 928 . . . 937.**

When the belt value 50 was passed, the control checks in each scanning step on the basis of the average belt speed whether the kinematics rob1 will presumably reach the value 150 at the same time as the belt reaches the value 200, and calculates a new set speed if necessary.

From line 22 to line 23 the position of rob1 does not change. Here, it only waits until the belt has reached the value 240 before starting the movement in line 25.

## 16 Belt synchronization type 3 (cam interpolation)

Three types of belt synchronization are available in the rho 3 controller. This chapter describes synchronization type 3 (cam interpolation).

The kinematics do not allow parallel belt movement. In the synchronous part the position of the robot is rigidly coupled to a master axis (belt axis) by means of a parameterizable curve.

Special function 21 can be used to select the type of belt synchronization.

The command SYNCHRON is used to activate belt synchronization. After this point, movements are synchronized with the belt in accordance with the set belt synchronization type.

The command SYNCHRON\_END deactivates belt synchronization.

### 16.1 Operating conditions for belt synchronization type 3

The following conditions apply for the synchronization part of the program of belt synchronization type 3:

- If kinematics move according to belt synchronization, no other processes can be active with these kinematics. A process must wait until the synchronous part is concluded.
- If runtime errors occur, the READY contact is opened in order to stop the belt itself, if required. The runtime error '**EMERGENCY STOP – Input**', code 528, is additionally shown.
- If kinematics movements are hindered by Feed hold, etc., the program is aborted and the kinematics runtime error '**BS–tolerance too big**', code 944..953, is issued.
- The interpolation types PTP and CIRCULAR are prohibited. In case of PTP, the program aborts with the runtime error message "PTP not allowed for belt–synchronous or sensor movement", code 23 and in case of CIRCULAR, with '**Belt-sync. n.possible**', code 20.
- For the first belt synchronous movement, a pause is made until the belt has reached the belt value which was programmed during the last traverse block before the belt synchronization part.
- If before the first belt synchronous movement the belt moves opposite the programmed direction, the program is aborted and the runtime error code '**wrong belt direction**', code 928..937 is issued.
- If during the first belt synchronous movement the belt moves opposite the programmed direction, the kinematics within this partial block also move opposite the programmed direction.

**However, the program does not run backwards.**

- Deviation from the programmed belt position is checked at the traverse block transitions. If this deviation is greater than the path covered by the belt in a specific number of interpolation cycles (5 are standard), the program aborts and issues the error message (which depends on the kinematics):

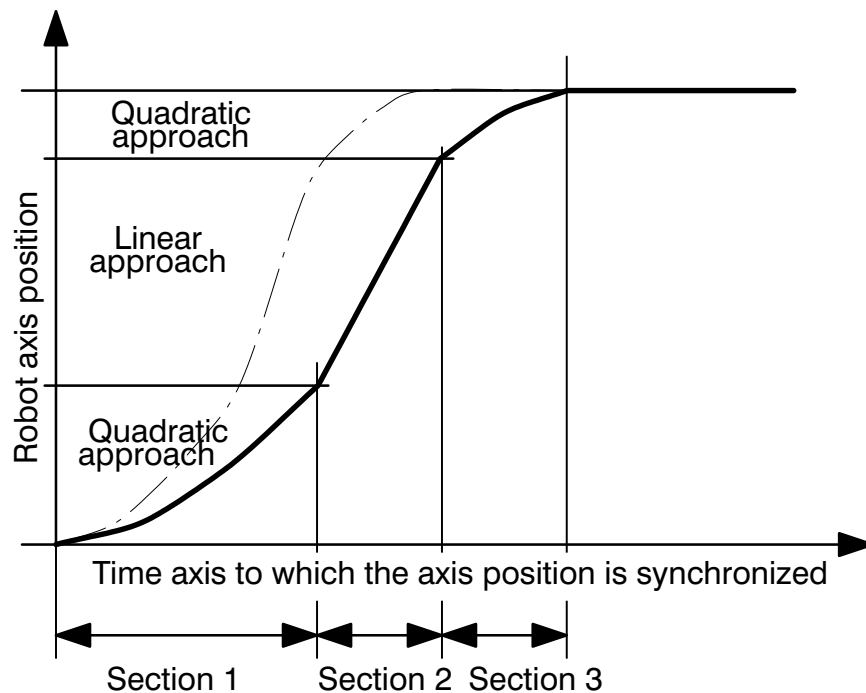
**'BS-tolerance too big'** code 944..953.

The scope of the tolerance range can be changed by calling

**SPC\_FCT : 21 = BELT\_TYP (VALUE INTEGER : NO VALUE INTEGER : TYP)**

- The belt difference between two sequential traverse blocks must not be 0 (for example, MOVE\_REL (0, 0) !!) Otherwise, the program aborts and issues the runtime error **beltdistance is zero**, code 89.
- The belt difference must have the same sign for each traverse block within the belt synchronization program section; otherwise, the error message **wrong belt-direction**, code 928..937 is issued.
- A MOVE\_REL as the first traverse block in the SYNCHRON part of the program is allowed.
- Here, the commands SYNCHRON and SYNCHRON\_END do not stop block preparation.
- Kinematic runtime errors and **'EMERGENCY STOP – INPUT'** are displayed in the process display as **'other errors'**.

## 16.2 Description of the parameterized curve



The programmed belt path is divided into three sections. In the first section, the robot position follows the belt position starting with the initial slope 0 with quadratic dependency, in the second section with linear dependency, and in the third section with quadratic dependency, and ends again with the slope 0.

The transitions between the sections are smooth. The slope of the second section (and thus the transitions as well) is calculated from these marginal conditions.

The scope of the first and third sections can be programmed over far ranges. The scope of the second section is yielded from the programmed overall length.

## 16.3 Programming the sections

The following are required for determining the curve:

- the entire belt path
- the belt path of the first section
- the belt path of the third section
- the distances for the robot coordinates

The entire belt path is yielded from the distance between the programmed belt coordinates of the end point and the belt coordinates of the most previously approached point.

To program the belt path of the first section, the meaning of the acceleration specification, which is not required here, is redefined. The value from A, kinematics AFACTOR and global AFACTOR correspond to this belt path.

To program the belt path of the third section, the meaning of the braking specification, which is not required here, is redefined. The value from D, kinematics DFACTOR and global DFACTOR correspond to this belt distance.

During the movement, all coordinates are moved along the curve specification in accordance with their programmed path end differences, even, if required, in opposite directions. It is possible to use the MOVE\_REL command for programming.

for example, MOVE\_REL ( 100, 200, 0, 0, 200 )

## 16.4 Special cases

A section is set to the minimum value of one percent if it is less than one percent of the overall length. If the sum of the first and third sections is greater than the overall length, the second section is set to the minimum value and the rest of the overall path is divided proportionally between the first and third sections.

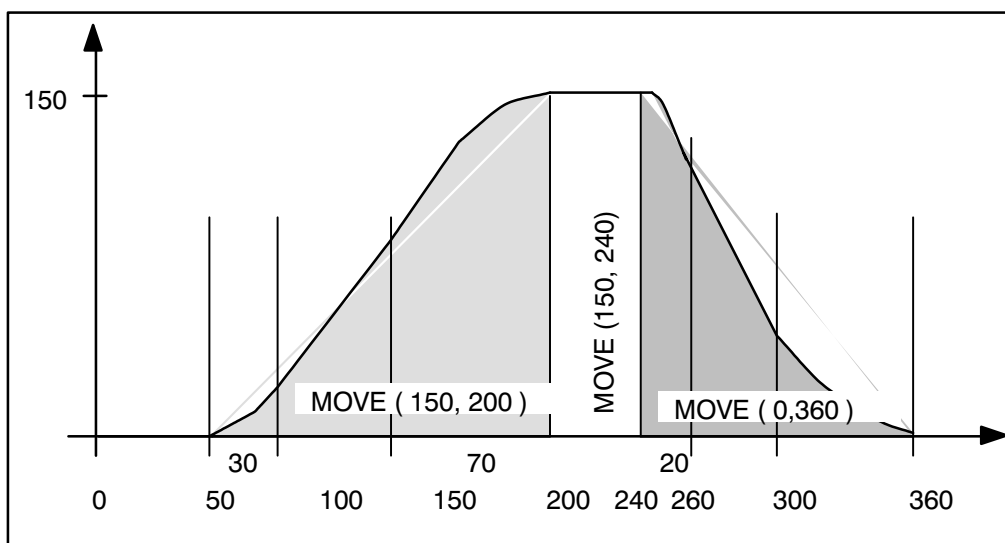
### 16.4.1 Comparison with normal traverse blocks

For traverse blocks with block-by-block ramp slopes, a 3-section curve is also yielded for the robot's position. Here, however, the scope of the sections depends on the programmed speed and acceleration/braking.

For belt type 3, however, the values for the sections are coupled directly with the belt position so that they are independent of the programmed speed.

At high speeds or short belt distances, of course, the effects of scanning must be taken into consideration.

Diagram for program example (cf. section 16.5)



## 16.5 Program example

```
30      ;;KINEMATICS : 1 = rob1
31      ;;rob1.JC_NAMES = k1, bnd
32      ;;rob1.WC_NAMES = k1, bnd
33
34      PROGRAM belt_type3
35
36      SPC_FCT : 21 = belt_type (      VALUE INTEGER : belt_no
37                                     VALUE INTEGER : belt_type)
38
39      rob1.BELT :      501 = rotary_axis
40
41      BEGIN
42
43      BELT_TYPE ( 1, 3 )
44      A = 100
45      loop
46      SYNC rotary_axis >= 360.0
47      MOVE LINEAR VIA (0,50)
48      SYNCHRON rob1 rotary_axis
49          AFACTOR = 0.3      ;A*AFACTOR = 30
50          DFACTOR = 0.7      ;A*DFACTOR = 70
51          MOVE ( 150, 200 )      ;
52          MOVE ( 150, 240 )      ; corresponds to wait until
53                                  ;rotary_axis >=240
54          A = 100      ;
55          AFACTOR = 0.2      ;A*AFACTOR = 20
56          DFACTOR = 0.6      ;A*DFACTOR = 60
57          MOVE: ( 0,360 )
58      SYNCHRON_END
59      JUMP loop
60      .
61      .
62      PROGRAM_END
```



Explanations of the program example above :

The BAPS code words are in upper case letters; otherwise, freely selectable names are in lower case letters.

Lines 36 and 37 contain the declaration for special function 21 for switching between belt synchronization types. It is called in line 43.

The belt counter is reset with the SYNC command in line 17.

In line 47 , rob1 is moved to position 0. The belt value 50 has no meaning for this movement, but is important for the first synchronous belt traverse block in line 51. There, rob1 holds off movement until the belt value reaches 50. If the belt moves away from the initial value, which is in negative direction in this case, the program is aborted and the following error message is issued:

**'wrong belt-direction' Code 928...937**

If the belt value 50 is exceeded, the position of rob1 is directly dependent on the belt value. In every scan increment, the position is determined using the curve from the belt value.

In line 49 ( 55 ), the belt path of the first section is set to the value 30 (20). In line 50 (56), the belt path of the third section is set to the value 70 (60). The values in parentheses refer to the Move instruction in line 57.

This yields the following sequence for the MOVE instruction in line 22:

From belt value 50 to belt value 50+30, the position of rob1 is quadratically dependent on the belt value. From value 80 to 200-70, the position is linearly dependent on the value. And from 200-70 to 200, it is again quadratically dependent. At the end rob1 is at 150.

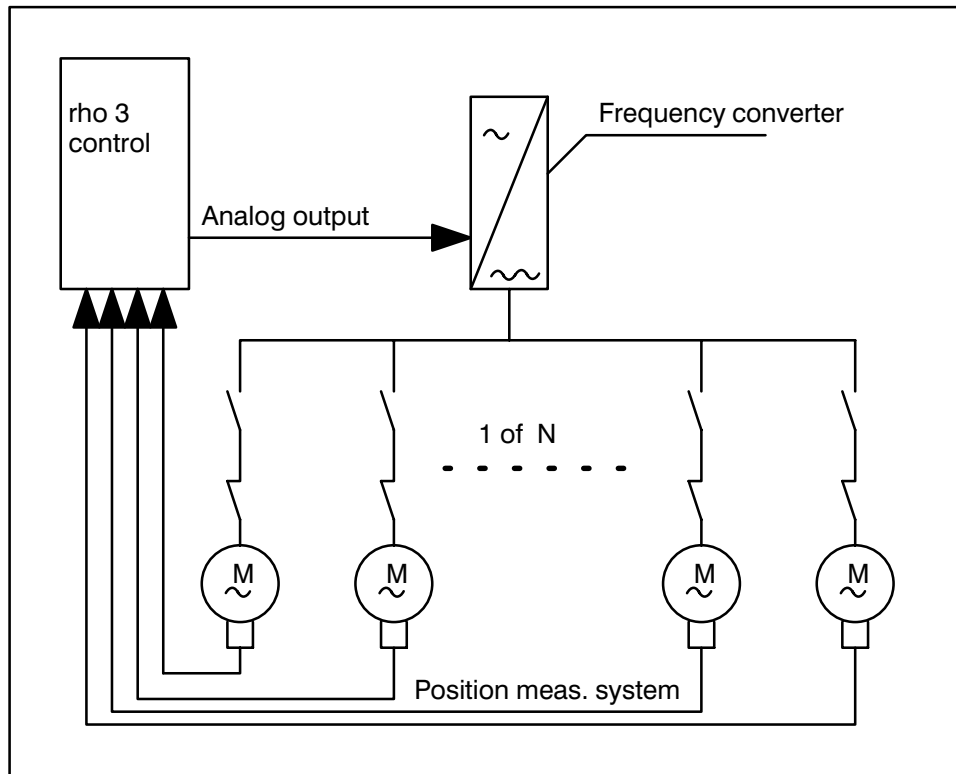
## ATTENTION !

Note the following restrictions when using this function:

- The meaning of the acceleration and braking values changes in the synchronous belt program section.
- Here, if the global or kinematic values for AFACTOR or DFACTOR are reduced, acceleration or braking is increased.
- The premature initiation of block preparation is not considered with the result that program branching operations can lead to program aborts due to input signals.
- The value for the programmed A is limited internally to  $2000/(P5)^2$ ; for example, 5 for  $P5 = 20\text{ms}$ . If smaller values are desired for the path, they must be implemented using correspondingly small values for AFACTOR and DFACTOR (see example).

**17 Several axes on one setpoint output**

In order to operate several axes with one frequency converter, the setpoints of several axes are output to the same setpoint output. The requirement in this case is that only one axis at a time is operated. If this requirement is not satisfied, the program is aborted with a servo error.



The Inpos signals of individual axes can be used to unambiguously reassign the setpoint outside of the controller to a given axis. If the Inpos signal of an axis is lost, the current setpoint at the axis must be assigned. The Inpos signals of all other axes are present.

The drives are switched on or off in dependence on the Inpos signals of the individual axes. The controller outputs 0 volts while an axis is in the Inpos range so that the drives do not operate in opposition to the brake. Compensation can be made with the correction time using machine parameter 201 for the delay times which elapse until the brake and drive are switched on or off. After the moved axis reaches the Inpos range, the control remains in the position controller until the correction time has elapsed.

## 17.1 Setting machine parameters

If several axes are to be operated using one setpoint output, the setpoint outputs must be assigned in P401 as follows:

### **P401 - Installing the measuring system board**

The entry is three-place for the subquery 6) allocation of the setpoint outputs.

For example: 101 – The measuring system is allocated the setpoint output 1.

The setpoint output is coded in the hundreds place. The one's place is the consecutive number of the axes which refer to setpoint output 1. Numbering is always continuous beginning with 1.

For example: One kinematics consisting of three axes and to be supplied via one setpoint output:

### **P401 subquery 6)**

Setpoint output 1st axis	101
Setpoint output 2nd axis	102
Setpoint output 3rd axis	103

### **P201 Inpos range**

Jog-Inpos range	[mm or degrees]	(Axis number values)
Auto-Inpos range	[mm or degrees]	(Axis number values)
Correction time	[1/1000sec]	(Axis number values)

### **The method of operation of the function in Jog mode can also be selected.**

Value = 0 as described above.

Value = 1 :

In the jog mode, the setpoint of the most previously operated axis is switched to the setpoint output. This axis remains in the position controller until a new axis is operated.

To complete the switch-over operation, a movement command must be generated. The Jog keys must be "blocked" in opposition to each other in the PLC program so that only one axis can be moved.

In addition to the Inpos signals, the Jog-RC inputs can also be used in the Jog mode.

## 18 ROPS3/IQPRO

In order to be able to use the additional language elements and online functions of the rho3 operating system version TO04, you need the current ROPS3/IQPRO versions W2C or W2B. All functions of the ROPS3/IQPRO software for program development (BAPS2 and BAPS/PIC compiler) are only operative with a hardlock (dongle) as of version W2C. The functions for data transmission to and from the rho control are not inhibited by the hardlock.

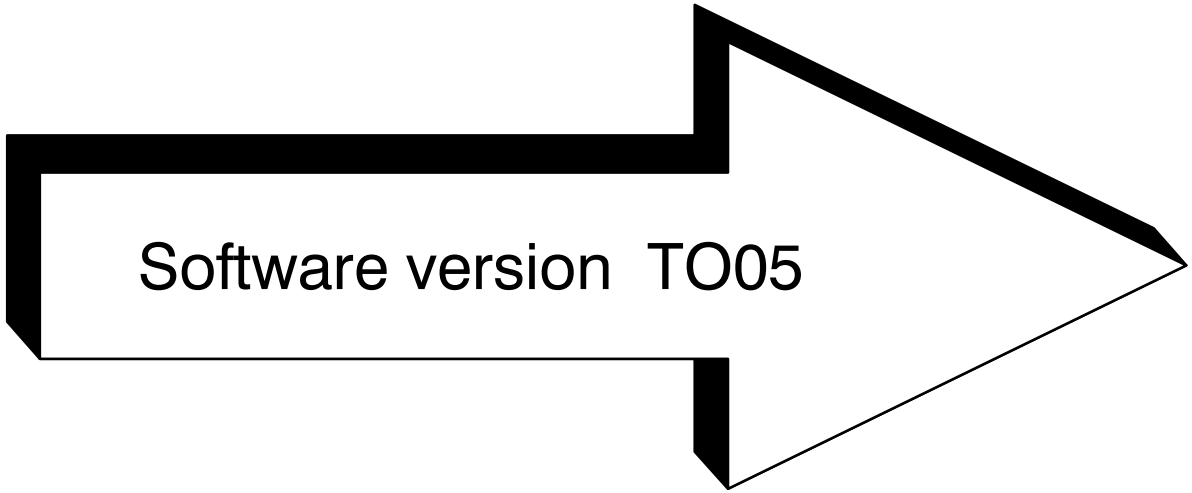
### 18.1 Compatibility list:

Software version						
ROPS3 / IQpro	Control rho 3/IQ140	BAPS2 compiler version	Control IQ120	PROFI SOFTWARE	PIC250 Program file .QLS for BAPS–PIC .P2O for PROFi	CL300 Program file .P3O for PROFi
W1C W1F	TO01E TO01I			2.43	R3_1I_D.QLS	RHO3_1I.P3O
W1J	TO02F	2.3	1.10	3.0	R3_2D_D.QLS R3_2D.P2O	RHO3_2D.P3O
W1K	TO03F	2.31	2.00	3.0	R3_2D_D.QLS R3_2D.P2O	RHO3_2D.P3O
W1L	TO03F	2.32	2.00	3.12	R3_4A_D.QLS R3_4A.P2O	RHO3_4A.P3O
W2B	TO04H	2.33	2.00	3.12	R3_4B_D.QLS R3_4B.P2O	RHO3_4B.P3O
W2C	TO04H	2.34	2.00	3.12	R3_5A_D.QLS R3_5A.P2O	RHO3_5A.P3O

New versions may be incompatible due to additional functions. For example: In version W1F, the function "ORD" was added to the BAPS2 language scope. BAPS2 programs containing this function cannot be compiled or run in older control versions than TO01I. However, all BAPS2 programs generated with W1C can also be compiled in TO01I or W1F.

#### Note:

Please also note the information in the Readme.txt and ReadPLC.txt files (which you will usually find in your "ROPS3" directory).



Software version T005

## Description of software extensions TO05

<b>1</b>	<b>Generating the PHG display</b> .....	<b>5 – 1</b>
1.1	READ PHG expansion .....	5 – 2
<b>2</b>	<b>Fast position controller on intelligent servo boards</b>	
2.1	Activation of the software function .....	5 – 3
<b>3</b>	<b>Asynchronous speed and target position stipulation</b>	
	<b>Special functions 41, 42</b>	
3.1	Required options .....	5 – 4
3.2	Modified "MOVE UNTIL" instruction MOVE UNTIL 'probe input' .....	5 – 4
3.3	Asynchronous distance stipulation .....	5 – 5
3.4	Asynchronous speed stipulation .....	5 – 5
3.5	Restrictions .....	5 – 5
<b>4</b>	<b>Writing, reading of binary files</b> .....	<b>5 – 7</b>
4.1	Read operations .....	5 – 8
4.2	Write operations .....	5 – 9
4.3	Data format .....	5 – 9
4.4	Restrictions .....	5 – 10
<b>5</b>	<b>Move_file</b>	
5.1	Configuration of the binary file .....	5 – 11
5.1.1	File header .....	5 – 11
5.1.2	Data division .....	5 – 12
5.2	The special function MOVE_FILE .....	5 – 14
5.2.1	Opening the BNR file .....	5 – 15
5.3	Integrity checks .....	5 – 17
<b>6</b>	<b>CAN bus for 12 axes SERVODYN-GC</b>	
6.1	Variable assignment of the CAN interface .....	5 – 20
6.2	Possible configurations of the CAN module .....	5 – 20
6.3	Axis control via CAN .....	5 – 21
6.4	Transmission rate .....	5 – 21
6.5	Restrictions .....	5 – 21
6.6	Digital I/O via CAN .....	5 – 22
6.7	Possible configurations .....	5 – 22
6.8	Transmission rate .....	5 – 22
6.9	Machine parameters .....	5 – 23
6.9.1	Fixing the axis configuration .....	5 – 23
6.10	I/O configuration of the CAN bus .....	5 – 25
6.11	Address ranges of the CAN inputs .....	5 – 26
6.12	Address ranges of the CAN outputs .....	5 – 26
6.13	Diagnosis of the local I/O .....	5 – 27
6.13.1	Display CAN input/output signals (MODE 7.14/7.15/7.9) .....	5 – 27
6.14	Error messages .....	5 – 28
6.14.1	Start-up and initialisation phase .....	5 – 28
6.14.2	Runtime errors .....	5 – 29

<b>7</b>	<b>Drive parameter download to Servodyn – GC via the CAN bus</b>	
7.1	Input of the drive parameters .....	5 – 30
7.1.1	Input and optimisation directly at the drive booster .....	5 – 30
7.1.2	Input via rho3 machine parameter program .....	5 – 30
7.2	Transmission of the parameters .....	5 – 30
7.3	Allocation of drive parameters – rho3 parameters .....	5 – 31
7.4	Machine parameters rho3 .....	5 – 33
7.5	Calculation of drive parameters from rho3 parameters available .....	5 – 33
<b>8</b>	<b>“Flying” measurement by means of probe input</b>	
	<b>Special functions 43 and 44 .....</b>	<b>5 – 35</b>
8.1	Restrictions .....	5 – 36
<b>9</b>	<b>Bit coupler with I/O board in the RC cardrack</b>	
	<b>I/O configurations 15 and 16</b>	
9.1	Description of function .....	5 – 37
9.2	Setting machine parameters .....	5 – 37
9.3	Address range .....	5 – 38
9.4	Restrictions .....	5 – 38
<b>10</b>	<b>Deleting the write/read buffer</b>	
10.1	Introduction .....	5 – 39
10.2	Command syntax .....	5 – 39
10.3	Deleting the write buffer .....	5 – 39
10.4	Deleting the read buffer .....	5 – 40
10.5	Availability/restrictions .....	5 – 41
10.6	Examples .....	5 – 41
<b>11</b>	<b>Disabling individual axes</b>	
11.1	Function .....	5 – 43
11.2	Safety notes .....	5 – 43
<b>12</b>	<b>The special function 46 (BLOCK_SEARCH)</b>	
12.1	Declaration .....	5 – 44
12.2	Call-up in BAPS .....	5 – 44
<b>13</b>	<b>Asynchronous inputs</b>	
13.1	Declaration .....	5 – 45
13.2	Asynchronous inputs in the BAPS program .....	5 – 45
13.3	Application example of asynchronous inputs .....	5 – 46

<b>14</b>	<b>Change of function operation</b>	
	<b>Directly approaching points in the Teach in mode</b>	
14.1	Operation .....	5 – 49
14.2	Restrictions .....	5 – 50
14.3	Preventing points which have been taught in from being overwritten .....	5 – 51
<b>15</b>	<b>Expansion of the function "fast, smooth start-off" .....</b>	<b>4 – 52</b>



## 1 Generating the PHG display

Versions from TO05G upward provide improved support for the change between operating system displays at the PHG and BAPS processes.

Storing the PHG display

The current display at the PHG is stored at the positive edge of the DIS\_PHGM\_RCI signal.

A display at the PHG is initiated at the negative edge of this signal with the previously stored display. If a dynamic display is active, e.g. an axis display, the stored display is updated immediately afterwards.

If, however, a PHG interface selection is made while the DIS\_PHGM\_RCI is active, no display is initiated at the falling edge of this signal.

### Application example 1

A message from the BAPS process should be displayed at the PHG and acknowledged by the operator pressing any key. Following this, the operator should be able to continue with his/her previous operation.

#### (Program only in excerpts)

```
DIS_PHGM_RCI = 1           ; output, which affects the RC input
                           ; signal no. 35 (byte 3 bit 4) in the PLC prog.
WAIT 0.1                   ; signal time RC -> PLC -> RC

WRITE PHG, MESSAGE, ANY_KEY
STATUS_PHG = STATUS(PHG)   ; waits until issued

                           ; so that acknowledgment does not occur
                           ; accidentally:
WAIT 2.0                   ; minimum time for display
READ_BEGIN PHG             ; deletes any characters already entered

READ PHG, KEY              ; waits, until key is pressed

DIS_PHGM_RCI = 0           ; shows the old display
```

### Application example 2

Operating system functions (e.g. MSD display, process-display, define) should be called up from a BAPS dialog process.

Determining which functions can be called up takes place via the PHGCODE.DAT file (see also option description version 4, pages 4–8).

```
e.g.:    0=0                ; for basic-level-return
         1=4.2              ; define
         2=7.2              ; errors/warnings
         3=7.12.2           ; automatic MSD display
```

(Program only in excerpts)

```
DIS_PHGM_RCI = 1 ; output, which affects the RCI
                ; signal no. 35 (byte 3 bit 4) in the PLC prog.
```

```
; dialog at the PHG for function selection,
; describing the variable CODE, e.g. CODE=3
```

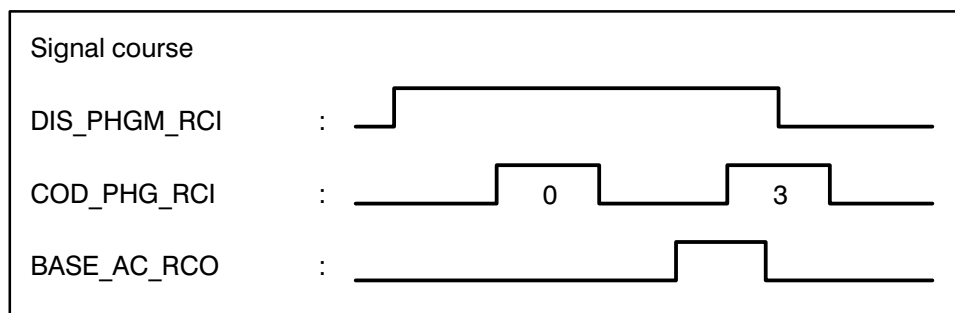
```
IF NOT BASE_AC_RCO ; input, corresponding to the RC output
                  ; signal no. 104 (byte 6 bit 1) in the PLC prog.

THEN  START
      PHGCODE=0 ; INTEGER output, connected with the signals for
              ; the PHG functions in the PLC program
              ; neg. edge of PHGCODE=0 activates RETURN
              ; in base level
      WAIT UNTIL BASE_AC_RCO=1 MAX_TIME=3.0.
              ; or special procedure due to e.g. EDITOR
      END
```

```
PHGCODE=CODE ; Selection of desired function via
              ; INTEGER output, connected with the signals for
              ; the PHG functions in the PLC program
```

```
WAIT UNTIL BASE_AC_RCO=0 MAX_TIME=3.0.
```

```
DIS_PHGM_RCI = 0 ; displays directly the new PHG function
                  ; readmits operating system displays
```



## 1.1 READ PHG expansion

If the DIS\_PHGM\_RCI signal is set, PHG keys are no longer interpreted immediately by the system, but collected in a buffer. The READ PHG is read from this Buffer by BAPS processes as well as by the operating system, although only after the display has been re-enabled. This allows characters to be entered in advance at the PHG with DIS\_PHGM\_RCI=1, even if a BAPS process is not yet present at the READ PHG. The PHG key signals do, however, remain available to the internal interface of the PLC.

## 2 Fast position controller on intelligent servo boards

The functions described below allow the rho3 – control system to be used for industrial applications where position profiles with a correspondingly fast position controller are necessary. Such an application is e.g. the removal of a volume of glass from a length of liquid glass, as is customary in the production of hollow glass.

### 2.1 Activation of the software function

The servo board software optimised to runtime can be activated with the help of machine parameter P29. This allows the realisation of considerably shorter position controller scan times than is possible with standard servo board software.

#### **ATTENTION !**

**The machine parameter P29 may only be altered after consulting BOSCH.**

This runtime-optimised software can only be used with certain hardware configurations.

Keep to the following list of restrictions:

- an intelligent servo board must be present (Servo-6i, -4i )
- a maximum of 2 axes of the available measuring system inputs may be used
- only incremental measuring systems may be used
- axis no. and module no., see also rho3 machine parameter P401, must agree
- measuring systems and setpoint outputs must not be allocated twice
- the axes must be controlled; forward control and controlled operation are not possible
- in test mode, only traversing “*with movement*” is possible
- a maximum of 2 High Speed inputs may be used
- the measuring position @MPOS can only be accepted with the probe
- analog outputs may not be used
- the watchdog is not retriggered; i.e. maximum 26 [msec] P2 clock time

The runtime–optimised servo board software can be activated individually for every servo board. A “1” should be entered for the corresponding servo board in machine parameter **P29**.

**The following position controller scan times are attained:**

15 MHz :	1 axis	500 µsec
	2 axes	700 µsec
25 MHz :	1 axis	400 µsec
	2 axes	600 µsec

### **3 Asynchronous speed and target position stipulation**

#### **Special functions 41, 42**

The move to “Probe” command is modified to be able to fulfill special requirements of e.g. balancing machines. During the balancing process, a continual course of speed should be guaranteed for all phases of balancing. At first the guinea pig is brought to the balancing speed. When the measuring process is finished, the control obtains an angle of rotation, which should be positioned. The angle of rotation refers to a reset pulse. During transfer, the speed is simultaneously set to the so-called start-up speed. If the reset pulse is detected, the previously transferred angle is quickly entered as the new set position.

#### **3.1 Required options**

As it is not known exactly how long the balancing process takes, a very large set position is passed on to the control. When measurement is complete, the angle determined by the balancing computer is transferred asynchronously to the control. This occurs by means of special function 42 from a parallel task. At the same time, the set speed is brought to start-up speed asynchronously to the movement. The special function 41 has been developed for this, making it possible to alter the V factor kinematically during movement.

#### **3.2 Modified ”MOVE UNTIL” instruction MOVE UNTIL ’probe input’**

In contrast to the previous “move until” command, movement with a modified move until is not cancelled abruptly when the condition is met; instead, the distance transmitted via special function 42 is traversed specifically. This occurs rapidly without Up and Down slope (time saving). In both cases, new contacts are then made (P1 side read POS). The alternative effect of the MOVE UNTIL instruction is coded via the channel number. The base of channel number 700 stands in contrast to “high-speed measuring” (see description of high-speed measuring option).

### 3.3 Asynchronous distance stipulation

The asynchronous distance for the modified Move until command is stipulated via special function 42.

#### Special function 42:

MOVE\_DIST (           VALUE INTEGER :       KIN\_NO  
                  VALUE DEC        :       MOVE\_DIST )  
                  KIN\_NO               Kinematic number  
                  MOVE\_DIST            Traverse distance to probe input

### 3.4 Asynchronous speed stipulation

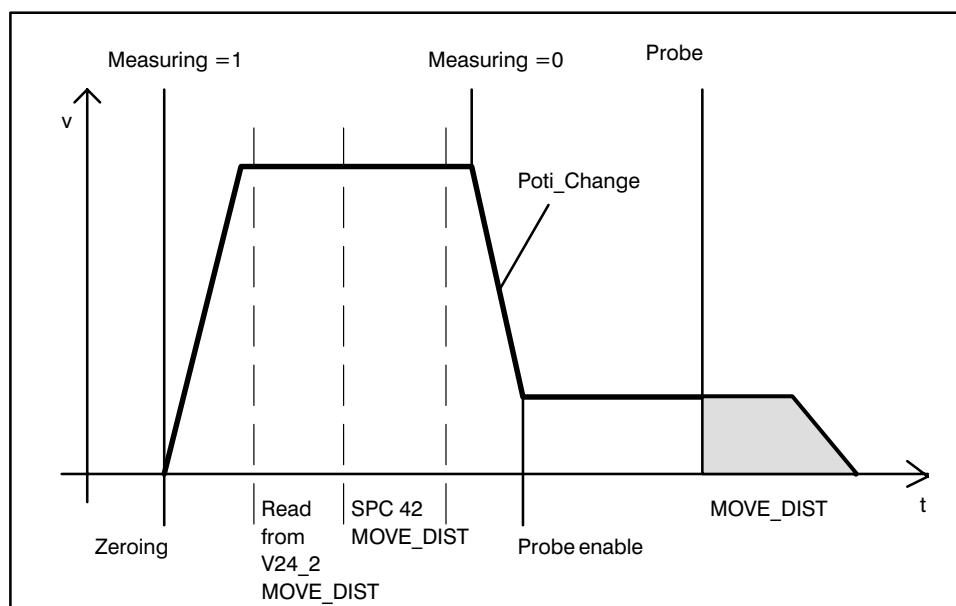
The asynchronous speed (e.g. from a parallel process) is stipulated via special function 41.

#### Special function 41:

POTI\_MODIFY (        VALUE INTEGER :       KIN\_NO  
                  VALUE DEC        :       KIN\_POTI )  
                  KIN\_NO – Kinematics number  
                  KIN\_POTI – asynchronous V factor  
                  Replaces the VFACTOR value specified in the program.

### 3.5 Restrictions

The modified 'Move until' command only works for a probe input on the servo board.



Program example

```
;;Control = rho3  
;;Kinematics: (1=balance)  
;;Balance.JC_NAMES = A_1  
;;Balance.WC_NAMES = X_K
```

Program balance

```
SPC_FCT : 3 = zero_reset      (Value integer :      KIN_NO  
                             JC_POINT   :      @Machpos)  
SPC_FCT : 41 = poti_modify   (Value integer :      KIN_NO  
                             Value REAL  :      kin_poti)  
SPC_FCT : 42 = move_dist     (Value integer:KIN_NO  
                             Value REAL:move_dist)
```

```
JC_POINT : @Zeropos  
Input    : 1 = measure  
Input    : 710 = probe  
REAL     : in_dist
```

Start

```
V_PTP = 0.5  
@Zeropos = @(0.0)
```

Loop:

```
Wait until measure = 1  
Zero_reset (1,@Zeropos) ; Zero reset of actual position
```

parallel

```
V factor = 1.0  
Move until probe =1 after @(9999999.0) ; Move until probe; with distance stipulation
```

also

```
Read V24_2,in_dist
```

also

```
wait until measure = 0  
poti_modify (1,0.2) ; asynchronous V factor change  
move_dist (1,in_dist) ; asynchronous distance stipulation
```

parallel\_end

program\_end

## 4 Writing, reading of BINARY files

The files I/O option is extended in order to guarantee processing of binary files within the BAPS program. In addition to the already implemented ASCII file I/O, the BAPS syntax now also offers the keyword `BNR_FILE`. The read and write operations have the same syntax as the commands of the ASCII file I/O. Differentiation between previous ASCII files I/O and binary files I/O is made when the files are declared. Binary files possess the file name extension "BNR".

ASCII files, designated by the file extension "DAT", are declared with the code word `FILE` and without type information. The `CHAR` type is then assumed implicitly. A declaration with explicit information on the `CHAR` type leads to the same result.

Example:

```
FILE : DAT_1
```

or

```
FILE CHAR : DAT_1
```

There are two types of declaration for binary files. The first uses the new BAPS code word `BNR_FILE`. A file declared in this way can contain data of various BAPS data types in binary format.

The second possibility requires only the already known BAPS code word `FILE` and additionally the data type of the data stored in the file. This type of declaration uses monotype files. This means that a file may only contain data of one BAPS data type. All file operations are checked at the point of transfer for type compatibility. File operations with `BNC` files are executed quicker than with `DAT` files, as there is no transfer of data from ASCII to BINARY or vice versa. During reading, the data are not accessed via the block-orientated file manager, but directly read from the `BNR` file, although this is on the condition that the file is stored sequentially in the user memory. The file is sequenced in the `READ_BEGIN` command, which simultaneously opens the file as well.

**BAPS commands of the binary file I/O :**

<b>Declaration A</b>	<b>BNR_FILE : FILE_1, FILE_2</b>	<b>Mixed BNR_files</b>
<b>Declaration B</b>	<b>FILE JC_POINT : P_JC FILE POINT : P_WC FILE REAL: REAL_VAR . .</b>	<b>Monotype BNR_files</b>
<b>Read operations</b>	<b>READ_BEGIN FILE_1 READ_BEGIN FILE_1, BYTE_OFFSET READ FILE_1, @P1, @P2</b>	<b>Positioning the read indicator</b>
<b>Write operations</b>	<b>WRITE_BEGIN FILE_1 WRITE_BEGIN P_WC, BYTE_OFFSET WRITE_END FILE_1 WRITE FILE_1, @P1, @P2</b>	<b>Positioning the write indicator</b>
<b>Close command</b>	<b>CLOSE FILE_1</b>	
<b>File-end function</b>	<b>END_OF_FILE (FILE_1)</b>	

## 4.1 Read operations

BNR files are opened with the READ\_BEGIN command and sequenced in the user memory, in order to guarantee quick access. If it is not possible to sequence the file due to storage limitations, the process involving the programming of the READ\_BEGIN command is interrupted with the runtime error message "file seq. failed.". In the READ\_BEGIN command, it is possible to give a BYTE offset for positioning the internal read indicator on any BYTE address.

Example: READ\_BEGIN FILE\_1, 80

The first 80 bytes of the file are left out in the example. Reading starts from the 81st byte. If a byte offset is programmed, it must be ensured that a meaningful value compatible with the read command is present at the appropriate position in the file.

The next example shows the reading of point values from a BNR file. The file to be read has a header with variable length and then a data section of variable size. The length of the header is stored in the first 4 bytes of the file as an INTEGER value.



**Example:**

```
PROGRAM L_BNR

BNR_FILE   :   D_PNT           ;*** mixed BNR file
INTEGER    :   HEADER_LENGTH
POINT      :   P1

BEGIN

;***** Read file header length from BNR file *****
READ_BEGIN D_PNT
READ D_PNT, HEADER_LENGTH

;***** Position read indicator at start of operand *****
READ_BEGIN D_PNT, HEADER_LENGTH

;***** Read points from BNR file and approach all points up to file end *****
MARK_1:    ;***** Read loop *****
           READ D_PNT, P1
           MOVE LINEAR VIA P1
           IF NOT END_OF_FILE (D_PNT)
           THEN JUMP MARK_1

CLOSE D_PNT

PROGRAM_END
```

## 4.2 Write operations

BNR files can be created or rewritten in the rho3. The new file is available for further processing after it has been closed with the BAPS command CLOSE. At this point the file is not necessarily sequenced. All files are sequenced with the “reset” interface signal. Similar to ASCII file I/O, files can be created or overwritten if they already exist. The WRITE\_END command is used to add data to that already present. The WRITE\_BEGIN enables a BYTE offset to be stated. With the subsequent WRITE commands, as many bytes are retained as are given in the BYTE offset. The remaining bytes of the file are overwritten with new data.

## 4.3 Data format

INTEGER and BINARY variables are stored as 4–byte integer values in two’s complement. Logical 1 is represented as INTEGER value 1 and logical 0 as INTEGER value 0. REAL values are also assigned 4 bytes and are stored in the 32-bit IEEE floating point format. The POINT and JC\_POINT types consist of REAL values. The number depends on the number of axes (point components). Other BAPS data formats are not permitted. BNR files do not contain any separators.

## 4.4 Restrictions

A file may not be write-opened simultaneously by several processes (subprocesses). After a file has been closed by one process, another process may open the file.

### **Write-opened files may not be opened for reading.**

In a subprocess (PARALLEL/ALSO branch), the file which has been write-opened by the appertaining main process or another subprocess may not be simultaneously read from. Should, for example, data from one file be read simultaneously into two subprocesses, the file must first be copied. Subprocess 1 then works with BNR file 1 and subprocess 2 with BNR file 2. Copying can be carried out on BAPS level with the help of special function 4 (COMMAND).

CLOSE commands are only effective in the process (subprocess) in which the READ\_BEGIN command is also programmed. If the process is finished, the files opened by this process are closed automatically if no CLOSE command was programmed.

## 5 MOVE\_FILE

The “move\_file” option provides for an exact reproduction of status profiles in the rho3. One freely definable position set point can be transferred for every position controller cycle.

This option is implemented as a special function. The transfer parameter is the name of the file which is to be executed and an identifier which indicates whether its interpolation points are stored in interpolation cycles or position controller cycles. This file is a binary file and contains the set points of the individual axes of a kinematic in binary and unformatted form for each cycle. It does not contain any separators. The axis set points are always position set points.

The option cannot run on rho3.2.

### 5.1 Configuration of the binary file

Binary files for the special function MOVE\_FILE contain the file extension “BNR”. BNR files can be generated offline on an AT or in the rho3. Binary files consist of a header and the actual data division.

#### 5.1.1 File header

The file header contains various file parameters, which are used for addressing the data division and for comparison to the machine parameters or special function parameters within the control system.

The file header has the following structure:

	File parameter	Length	Type
1	HEADER LENGTH	4 bytes	Integer
2	INTERPOLATION / POSITION LOOP BASE	4 bytes	Integer
3	TIME BASE	4 bytes	Integer
4	NO. OF AXES	4 bytes	Integer
5	VERSION IDENTIFICATION	4 bytes	Integer
6	DISABLE BASE CHECK	4 Bytes	Integer
7	RESERVE	14 * 4 bytes = 56 bytes	Integer

#### 1 HEADER LENGTH

The header length of a BNR file created in the rho3 is 80 bytes. BNR files with other header lengths can, however, also be used in conjunction with the MOVE\_FILE option. It is important that the correct header length has been entered in the file header, in order to be able to address the start of the data division.

**2 INTERPOLATION/POSITION CONTROLLER BASE (IP-PL-BASE)**

Parameter 2 indicates whether the individual point values of the file are stored in interpolation cycles or, as is possible when using a servo i switching group, in the further subdivided position controller cycles of the servo board.

IP-PL-BASE = 1 signifies : the points are stored in interpolation cycles

IP-PL-BASE = 2 signifies : the points are stored in position controller cycles

**3 TIME BASE**

This parameter contains the time base in microseconds in which the file was generated. Depending on parameter 2 this is the value of the machine parameter P5 (Clock Time) \* 1000, if necessary divided by the servo board divider.

**4 NO. OF AXES**

This parameter contains the number of axes of the point values stored in the file.

**5 VERSION IDENTIFICATION**

This parameter contains an integer number which corresponds to the appertaining control software version. The version identification for software version TO05 is equal to 500.

**6 DISABLE BASE CHECK**

The checking of the INTERPOLATION / POSITION LOOP BASE and the TIME BASE is switched off by the defined setting of this parameter to value 1. With values not equal to 1, these file parameters are compared with the special function parameters and the parameters within the control system.

**7 RESERVE**

The rest of the file header is laid out as a reserve buffer of 56 bytes (14 \* 4 bytes) in length and is available for user-specific file identifications and extensions. This results in a total header length of 80 bytes. The reserve buffer should be preset with 0.

**5.1.2 Data division**

The data division follows directly after the header division and contains point components in the 32-bit IEEE floating point format without separators. A point component is 4 bytes in length. The storage location for a point is calculated as NUMBER OF AXES \* 4 bytes. The address of the first point of the file results from the start address of the file + HEADER LENGTH.

The order within the data division is as follows:

Interpolation point 1 :	1st cycle	axis 1
	1st cycle	axis 2
	1st cycle	axis 3
Interpolation point 2 :	2nd cycle	axis 1
	2nd cycle	axis 2
	2nd cycle	axis 3

etc.

Picture 1 : Configuration of the binary file

HEADER LENGTH
IP-PL-BASE
TIME BASE
NO. OF AXES
VERSION IDENT.
BASE-CHECK OFF
RESERVE
DATA DIVISION

Picture 2 : Example of a binary file

Address relative to the file start (HEX):

00	80
04	2
08	400
0C	1
10	500
14	0
18	RESERVE
50	0.0288
54	0.0864
	0.1728
	0.2880
	0.4320
	0.6048
	.
	.
	.
	.
	.
	.
	.
	.
	.
	359.712
	359.827
	359.914
	359.971
	360.000

**Observation :**

- All address information is represented in hexadecimal form.
- The data division consists of 32-bit IEEE floating point numbers.

Picture 2 shows a binary file for a single-line kinematic. The data division contains position set points which refer to a position controller cycle of 400 microseconds. The data format corresponds to the TO05 (version identification = 500)

BASE CHECK OFF = 0 signifies that the parameters IP PL BASE and TIME BASE are being checked.

## 5.2 The special function MOVE\_FILE

Calling up the special function MOVE\_FILE causes a curve shape stored in a binary file to be traversed; the values from the file are read in this curve and issued unchanged in interpolation cycles or in position controller cycles of the servo board file as position set points. The calling-up of the special function MOVE\_FILE will hereafter be called MOVE\_FILE block.

### Declaration of the special function

<b>SPC_FCT : 45 = MOVE_FILE (</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>: KIN_NO</b>
		<b>BNR_FILE</b>	<b>: CURVE_X</b>
	<b>VALUE</b>	<b>INTEGER</b>	<b>: BASE</b>
		<b>JC_POINT</b>	<b>: @MODULO_FLAG</b>
	<b>VALUE</b>	<b>FIELD[1..6] INTEGER</b>	<b>: RESERVE )</b>

The special function requires the parameters **kinematic number**, **file variable**, **base** and **modulo flag**. The RESERVE parameter is meant for possible extensions in future operating system versions.

### Special function parameters

**KIN\_NO**     **Kinematic number**

Specifies the kinematic to be traversed.

**CURVE\_X**   **File variable**

Specifies the name of the BNR file for the movement.

**BASE**        **Base**

The base indicates whether the position set point of the file was issued in interpolation cycles or in position controller cycles. This special function parameter is compared with the header parameter 2 (IP-PL-BASE); see 1.3.

**@MODULO\_FLAG**     **Modulo flag**

The modulo flag is required for **endless axes** and must be **preset with the value 0 for other axes**. Every component of the modulo flag is assigned to an axis of the kinematic. The individual components are of data type REAL and serve as identifiers for whether and with which value the modulo computation of the endless axis concerned has been executed at the end of the MOVE\_FILE block.

Following REAL values are permissible as identifiers :

**0.0**            no modulo computation

**1.0**            modulo computation at the end of the block. Modulo value is the last axial value of the BNR file.

**2.0**            modulo computation at the end of the block. Modulo value is the default machine parameter value

Other REAL values are treated as 0.0 (no modulo computation).

If at the end of a MOVE\_FILE block a modulo computation is to be executed for an endless axis, there are two possible modulations. Modulation is possible either with the file end value of the axis concerned or with the modulo value set in machine parameter P311.

The modulo flag must always be preset, as otherwise the process is aborted with the runtime error message "MF: MOD-Flag n. init." (error code 101).

*RESERVE*   **Reserve**

### 5.2.1 Opening the BNR file

Before initial call-up of MOVE\_FILE, the appertaining binary file must first be opened for reading by the BAPS command READ\_BEGIN. When opening the file, the read indicator within the control system must be positioned at the start of the data division of the BNR file. This is attained by including a byte offset in the READ\_BEGIN command. The offset is in this case equal to the file header length (80 bytes). The command for opening the file is:

```
                                READ_BEGIN CURVE_X, 80
or:
                                CONST : HEADER_LENGTH = 80
                                .
                                .
                                READ_BEGIN CURVE_X, HEADER_LENGTH
                                READ_BEGIN CURVE_Y, HEADER_LENGTH
```

After the file has been opened, it can be used as often as desired in MOVE\_FILE call-ups. The file must be closed again with the BAPS command CLOSE after the last MOVE\_FILE call-up. If no READ\_BEGIN was programmed before the first MOVE\_FILE block, the runtime error message "READ\_BEGIN expected" (error code 63) is issued.

The following example shows the declaration and call-up of the special function. Two curve shapes, which are stored in two BNR files JC\_0\_90, JC\_90\_0, are traversed alternately.

Program example :

```

;;CONTROL = RHO3
;;KINEMATICS : (1 = ROBI1)
;;ROBI1.JC_NAMES = A1, A2
;;ROBI1.WC_NAMES = K1, K2
PROGRAM MOVEFILE
SPC_FCT : 45 = MOVE_FILE (      VALUE    INTEGER      : KIN_NO
                                BNR_FILE    : CURVE_X
                                VALUE    INTEGER      : BASE
                                JC_POINT    : @MODULO_FLAG
                                VALUE    FIELD[1..6] INTEGER : RESERVE      )

BNR_FILE      : JC_0_90, JC_90_0
JC_POINT      : @MOD_FLAG
FIELD[1..6] INTEGER : RESERVE

CONST          : IP_CYCLES   = 1,
                PL_CYCLES   = 2,
                HEADER_LENGTH = 80

START

;***** open BNR files *****
READ_BEGIN JC_0_90, HEADER_LENGTH
READ_BEGIN JC_90_0, HEADER_LENGTH

@MOD_FLAG = @(0, 0)                ;*** no endless axes

;***** traverse cycle *****
RPT 10 TIMES
    MOVE_FILE (1, JC_0_90, PL_CYCLES, @MOD_FLAG, RESERVE)
    MOVE_FILE (1, JC_90_0, PL_CYCLES, @MOD_FLAG, RESERVE)
RPT_END

;***** close BNR files *****
CLOSE JC_0_90
CLOSE JC_90_0

PROGRAM_END

```



### 5.3 Integrity checks

The entries in the file header are compared with the machine parameters within the control and the programmed special functions in the MOVE\_FILE block.

The following runtime errors may arise during this check:

Runtime error messages	Error code	Explanation of the runtime error messages:
Inad. Real-Expression	38	<p>The first and last points of the BNR file are checked at the start of the MOVE_FILE block to ascertain whether their components are meaningful floating point numbers in IEEE format. For runtime reasons, however, the complete BNR file cannot be checked.</p> <p>In order to completely exclude FPU traps (system errors), the user can use a BAPS program to check the BNR files for unpermitted floating point numbers (type REAL). This check only has to be executed once per BNR file and can take place at a temporally uncritical point in the BAPS program or once before the start of the MOVE_FILE process.</p> <p>The BAPS program could resemble the L_CURVE example program depicted below. The L_CURVE program reads a BNR file CURVE.BNR and in addition creates from this a readable DAT file D_CURVE.DAT.</p>
MF: Version identification	95	The version identification of the file is not compatible with the rho3 operating system version installed.
MF: No. of axes	96	The number of axes of the kinematic is not equal to the number of axes entered in the file header.
MF: Read marker err.	97	The read indicator within the control has not, in the READ_BEGIN command, been positioned within the data division of the BNR file. It points to the file header.
MF: Time base	98	The file parameter does not fit the corresponding internal cycle time (interpolation or position controller cycle). This monitor can be switched off by the file parameter BASE_CHECK_OFF.
MF: IP-PL base	99	The file parameter IP-PL-BASE does not fit the special function parameter BASE (interpolation or position controller base). This monitor can be switched off by the file parameter BASE_CHECK_OFF.

Runtime error messages	Error code	Explanation of the runtime error messages:
MF: Format error	100	<p>Either the data division of the BNR file has an incorrect length, or the internal read indicator has, in the READ_BEGIN command, been wrongly positioned inside the data division of the BNR file. The 2nd case arises when the read indicator is not positioned at the start of the data division, which is theoretically permitted. In this case, it should absolutely be ensured that the number of bytes from the read indicator to the file end is divisible by (4 * number of axes). The following permissible values then result for the offset in the READ_BEGIN command:</p> $\text{OFFSET} = \text{HEADER LENGTH} + N * (4 * \text{NUMBER OF AXES})$ <p>N : integer &gt;= 0 HEADER LENGTH = 80</p>
MF: MOD flag not init.	101	<p>The special function parameter @MODULO_FLAG has not been preset. For all axis types it must be preset with at least 0, as the data type of the flag is itself @JC_POINT type.</p>
MF: No endless axis	102	<p>The value 1.0 has been transferred for the axis component of the modulo flag in the MOVE_FILE special function call-up, although the corresponding axis in the machine parameter program is not declared as an endless axis.</p>

Example:

```
PROGRAM L_CURVE

CONST          : AX = 2          ;*** no. of axes ***

BNR_FILE       : CURVE          ;*** source, mixed BNR file (INTEGER + REAL)
FILE           : D_CURVE        ;*** destination, DAT file
FIELD[ 1..20 ] INTEGER : HEADER;*** HEADER LENGTH = 80 BYTES = 20 INTEGER values
FIELD[ 1..AX ] REAL   : DATA    ;*** no. of field elements = no. of axes

BEGIN

;***** read FILE HEADER from BNR file CURVE.BNR and write to DAT-file D_CURVE.DAT *****
READ_BEGIN CURVE
WRITE_BEGIN D_CURVE

READ CURVE, HEADER
WRITE D_CURVE, HEADER

;***** read DATA from BNR file CURVE.BNR and write to DAT-file D_CURVE.DAT *****
MARK_1:          ;***** read loop *****
    READ CURVE, DATA
    WRITE D_CURVE, DATA
    IF NOT END_OF_FILE (CURVE)
    THEN JUMP MARK_1

CLOSE CURVE
CLOSE D_CURVE

PROGRAM_END
```

## 6 CAN bus for 12 axes SERVODYN-GC

Up to 6 axes can be operated via every CAN bus. The axes 7..12 receive the same telegram addresses (identifiers) as axes 1..6. This makes it possible to connect a complete, second SERVODYN-GC drive assembly to the second CAN bus.

### 6.1 Variable assignment of the CAN interface

Up to 2 CAN buses can be configured via the CAN module on the I/O board. This results in new applications for the CAN interface.

### 6.2 Possible configurations of the CAN module

The two interfaces can be used selectively for axes and digital I/O signals.

Due to the data required for the messages to the SERVODYN-GC, the following configurations are possible on every CAN bus.

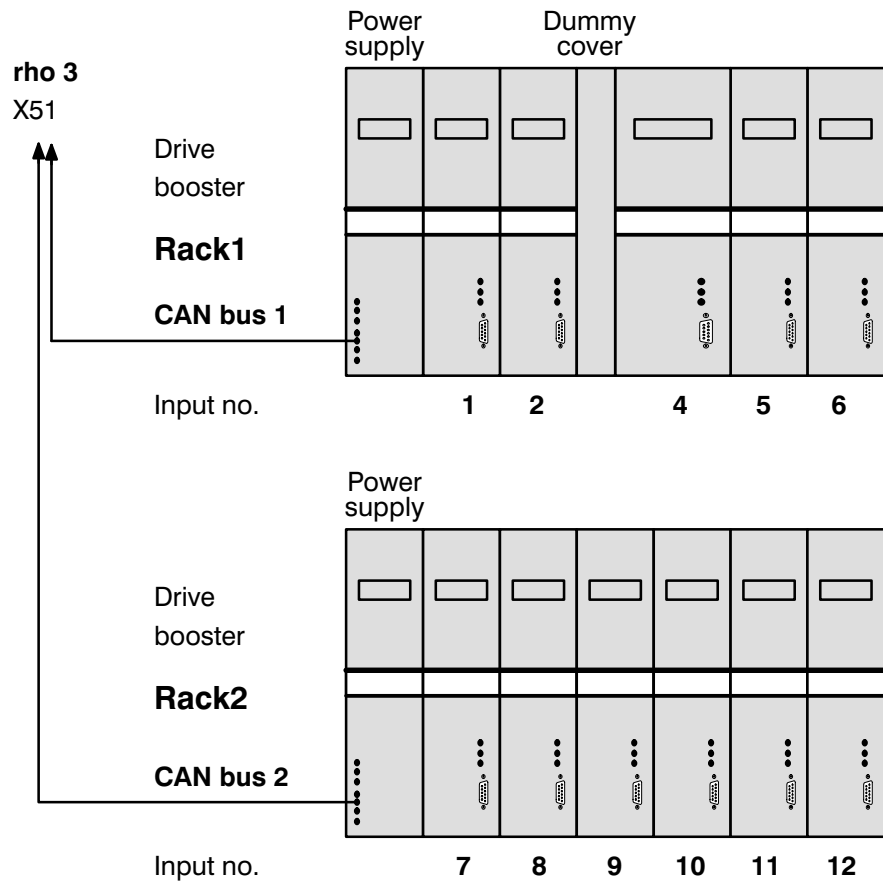
Number of axes	Max. number of dig. inputs [byte]	Max. number of dig. outputs [byte]
6	none	none
4/5	8	8
3	16	16
2	24	24
1	32	32
none	40	32

The configuration is set via machine parameter program see page [5 – 23](#).

The CAN bus 2 can also be used when the CAN bus 1 is not connected.

**6.3 Axis control via CAN**

Up to 6 axes can be operated via every CAN bus. The axes 7..12 (2nd CAN bus) receive the same telegram addresses (identifiers) as axes 1..6. This makes it possible to connect a complete, second SERVODYN-GC drive assembly to the second CAN bus.



**6.4 Transmission rate**

The transmission speed (baud rate) is fixed at the drive boosters to 1 Mbaud.

**6.5 Restrictions**

When more than 6 axes are in use, a CP/MEM 5 board with a 30 MHz processor is required.

**6.6 Digital I/O via CAN**

**6.7 Possible configurations**

The CAN interface for digital I/O should be used as a substitute for or an extension of the digital input/output boards. Application is possible with all I/O configurations (PIC250, ext. PLC coupling). Up to 5 blocks each with 8 bytes of input signals + 4 blocks each with 8 bytes of output signals can be defined. Allocation is determined via machine parameters.

**6.8 Transmission rate**

The max. transmission rate (baud rate) is dependent on the cable length between rho3 and I/O modules.

Max. cable length [meters]	Baud rate
up to 20	1 Mbaud
up to 40	500 kbaud
up to 80	250 kbaud
up to 160	25 kbaud

As the cable length can be greater than 20 m with local I/O modules, it must be possible to set the baud rate accordingly via machine parameters. It must likewise be possible to set the baud rate at local I/O modules (e.g. dip switch).

**NOTE:**

**With baud rates  $\lt \gt$  1 Mbaud, no axes can be connected at the corresponding CAN interface.**

## 6.9 Machine parameters

### 6.9.1 Fixing the axis configuration

#### **P15 servo board types**

In order to be able to couple axes via the CAN interface, one of the following servo board types must be entered under P15.

- 3: CAN module. This module can currently only be inserted on the I/O board (in connection with a CP/MEM 4). Identification for CAN module can be entered at any rack position. For combinations with narrow or wide servo board, see identifications 5 and 6.
- 5: Combination narrow servo board without Intelligence + CAN module. Can only be entered for place 1.
- 6: Combination wide servo board without Intelligence + CAN module. Can only be entered for place 1. The remaining places can in addition be equipped with any other servo boards. An assignment with 2 CAN modules is not possible.

#### **P5 clock start time**

When coupling axes via CAN interface, the max. programmable clock start time is 32 ms (limited by drive booster). If the control is operated without additional SERVO-I, the max. clock start time is reduced to 26 ms (rho3-internal watchdog). When configuring the system, it should be noted that a CP/MEM 5 module with a 30 MHz processor is used if these times are exceeded.

#### **P401 equipping the measuring system boards**

##### **1. Subquery : servo board**

- Number of the plugged servo board. The servo board plugged-in furthest to the left has the number 1. For CAN the number of the SB place which was assigned the CAN interface under P15 is to be entered.

##### **2. Subquery : No. of plug**

- For CAN measuring system only X51 is permissible.

##### **3. Subquery : module number**

- Where there are several servo boards: with every additional servo board, the module number restarts at 1.  
With potentiometer modules, the continuous number of the plug should be entered.  
For all axes with CAN, the first module number not used by other measuring systems is to be used.

##### **4. Subquery (dependent on measuring system)**

###### **4. 1.Incremental measuring system**

.  
.

## 4.2. Absolute measuring system

.

.

## 4.3. CAN interface

### 4.3.1. Input on module

- **Allocation axis number --> Input on CAN module when using Servodyn-GC:**

The axes are allocated via these subparameters to the rack positions in the Servodyn-GC racks. Every CAN bus can be assigned a drive booster rack, i.e. up to 2 racks can be connected to a rho3 CAN module. The boosters in rack 1 (for CAN bus 1) are continuously numbered 1..6. The boosters for bus 2 receive the numbers 7..12. A booster rack can also be connected to bus 2 if bus 1 is assigned either only with a digital I/O or not at all. The booster racks do not have to be fully equipped. Gaps between the boosters are also permitted. It should, however, be ensured that the input numbers are entered according to the actual state of assembly. If a drive booster is using 2 rack positions, the right-hand position is to be entered as the input number for the CAN module input.

### 4.3.2. Referencing mode

- **MODE = 0**  
“Normal” referencing  
Axis moves to the next zero crossing of the measuring system after recognition of the reference point switch.
- **MODE = 1**  
Referencing with correct orientation  

With special SCARA kinematics, missing mechanical strokes may cause the gripper rotational axis to execute more than one rotation. If referencing is started with a contorted gripper, the control must ensure that during referencing the gripper axis is turned back to its original position.

A resolver measuring system on the motor axis measuring over 360 degrees absolute measures the position of the sleeve of the gripper when the reference point switch is reached.

If the motor axis opposite the sleeve axis is turning via a gear with non-integer transmission ratio, the number of rotations made from the original point is calculated from the position of the sleeve within the 360 degrees.
- **MODE = 2**  
“Resolver-suitable” referencing  
Axis does not approach the next zero point after recognition of the reference point switch. When the reference point is reached, the resolver position is read in and the axis halted immediately.
- **MODE = 3**  
Combination of MODE 1 and MODE 2



**4.3.3. Gear factor**

- Query is only performed if **ref. mode = 1** or **ref. mode = 3**
- entry necessary with decimal placing

**4.3.4. Pulses/rotation**

- see information from the drive booster producer.

**4.4. Potentiometer**

.

.

**6.10 I/O configuration of the CAN bus**

**Machine parameter P30**

The configuration of the CAN bus 1+2 is set with P30.

Entry:

No. of input blocks [8 Byte] : 0..5  
No. of output blocks [8 Byte] : 0..4  
Baud rate : 0..3

Parameter value	Transmission speed
0	1 Mbaud
1	500 kbaud
2	250 kbaud
3	125 kbaud

It should be ensured that the maximum possible number of digital I/O blocks is not exceeded (see point 6.2).

**NOTE:**

The baud rate for a CAN bus may only be set not equal to 1 Mbaud when the bus is not occupied by axes.

## 6.11 Address ranges of the CAN inputs

### P31

Determination of the address ranges and identifiers for digital input signals which are to be read in via the CAN interface. The PIC byte addresses are to be given (see signal description rho3). The block length indicates how many bytes are being transferred from the corresponding input nodes, i.e. with block lengths < 8, the full range of a block is not utilised.

Entry:

Start address digital inputs CAN block 1..5:	0..127
Block length digital inputs CAN block 1..5:	1..8
Identifier digital inputs CAN block 1..5:	541..550

#### NOTE:

The number of entries is dependent on the number of input blocks as determined in P30.

## 6.12 Address ranges of the CAN outputs

### P32

Determination of the address areas and identifiers for digital output signals which are to be issued via the CAN interface. The PIC byte addresses are to be given (see signal description rho3). The block length indicates how many bytes are being transferred to the corresponding output nodes, i.e. with block lengths < 8, the full range of a block is not utilised.

Entry:

Start address digital output CAN block 1..4:	0..127
Block length digital outputs CAN block 1..4:	1..8
Identifier digital outputs CAN block 1..4:	471..480

#### NOTE:

**The number of entries is dependent on the number of output blocks as determined in P30.**

It should be ensured that the address ranges are not overlapped when additional digital I/O boards (I/O 64/40, PC I/O coupling) are in use. It is therefore sensible to use the high-order I/O addresses first, as the lower address ranges are permanently used by the I/O boards.

## **6.13 Diagnosis of the local I/O**

### **6.13.1 Display CAN input/output signals (MODE 7.14/7.15/7.9)**

#### **MODE 7.14 DISPLAY DIGITAL CAN INPUTS**

Display of digital inputs of the CAN I/O modules.

Select with arrow up/arrow down, confirm with ENTER when desired bus and identifier appear.

Dynamic display of the input signal statuses (High/Low) of the CAN I/O modules.

#### **MODE 7.15 DISPLAY DIGITAL CAN OUTPUTS**

Display of digital outputs of the CAN I/O modules.

Select with arrow up/arrow down, confirm with ENTER, when desired bus and identifier appear.

Dynamic display of the output signal statuses (High/Low) of the I/O board.

#### **MODE 7.9 Test digital outputs**

When testing the digital outputs, the output signals on the I/O modules coupled via the CAN bus are tested in parallel to the output signals on the I/O boards.

## 6.14 Error messages

### 6.14.1 Start-up and initialisation phase

<b>3716 = System error 133</b>	<b>:</b>	<b>MODE not 10</b>
PHG display	:	“MODE not 10”:CAN I/O”
Cause	:	Download of the axis processor code does not function or error during initialisation of the axis processor logic.
Comment	:	Hardware not completely equipped (for example, no FPU or small CAN module available).
<b>3732 = System error 149</b>	<b>:</b>	<b>CAN dual port overflow</b>
PHG display	:	“CAN DP overflow”
Cause	:	Too many user CAN input/output blocks have been defined.
Comment	:	Check P30/P401 (no. of CAN axes).
<b>3733 = System error 150</b>	<b>:</b>	<b>Wrong baud rate at CAN interface</b>
PHG display	:	“CAN incorr. BAUDR.”
Cause	:	The baud rate for a CAN bus has been set <> 1 Mbaud, although this bus is occupied with axes.
Comment	:	Check P30/P401 (allocation of CAN axes).
<b>3734 = System error 151</b>	<b>:</b>	<b>Undefined CAN I/O block</b>
PHG display	:	“CAN I/O block undef.”
Cause	:	CAN I/O block was not defined via P31/P32.
Comment	:	Check P31/P32.

**6.14.2 Runtime errors**

**380..381 = Impermissible data on CAN bus**

PHG display : "inadm. data CAN x ly"  
x = Number of the CAN bus, y = Number of the input block

Cause : CAN input module has sent data at an unpermitted time.

Comment : Input module does not correspond to the specification, input module defective.

PHG display : "imperm. data CANx Oy"  
x = Number of the CAN bus, y = Number of the input block

Cause : Write-access on CAN module is disabled.

Comment : CAN module in rho3 defective.

**380..381 = No data transmission from CAN input module**

PHG display : "no transm. CANx ly"  
x = Number of the CAN bus, y = Number of the input block

Cause : No data have been recieved from the CAN input module for a P2 cycle at least.

Comment : CAN bus connection not working (check plug and cable), input module defective.

## **7 Drive parameter download to Servodyn – GC via the CAN bus**

From rho3 software version TO05 onwards, it is possible to store the parameters of the drive boosters in the control (rho3) which are connected via CAN bus and, if required, transmit them via the CAN interface to the drive boosters.

### **7.1 Input of the drive parameters**

#### **7.1.1 Input and optimisation directly at the drive booster**

There remains the possibility of direct parameter input at the drive booster. This is above all useful for optimising the control parameters.

#### **7.1.2 Input via rho3 machine parameter program**

Once the parameters for the individual drives are fixed, they can be stored via the rho3 machine parameter program in the EEPROM of the rho3. The rho3 machine-parameters are further extended by the **P 600** for CAN drive parameters. Parameters for which a dependency between control and drive exists (e.g.: scan time) or which are already defined in the control (e.g.: SW limit switch) are determined directly from the control parameters present. Determination and allocation of the parameters see [pnt. 6.9](#)

### **7.2 Transmission of the parameters**

The drive parameters stored in the rho3 EEPROM are transmitted to the drive boosters in the start-up phase. The drive boosters read the parameters and store them in the local EEPROM.

The transmission of the parameters can be switched off via the rho3 machine parameter P601. In optimisation phase, this prevents control parameters optimised at the drive from being overwritten by the parameters stored in the rho3. On the other hand, the start-up phase is shortened when switched-off parameters are transmitted. Therefore, parameter transmission should only be activated when commissioning, or when changing a booster for a unique transmission of the drive parameters.

**7.3 Allocation of drive parameters – rho3 parameters**

Para. No.	rho3 parameter	Comments	BOSCHTRM command
1	P601	Drive stores parameter into EEPROM if TRUE	–
2	P602	Drive booster type	–
3	P603	Motor type	SM
4	P5	CLOCK start-time (rho 3)	SC
5	P604	Loop gain of the speed controller	SP
6	P605	Integral action time of speed controller	SI
7	P606	Loop gain of the position controller	SG
8	Calc. from P204	Software limit switch in clockwise direction	OL
9	from P205	Software limit switch in counterclockwise direction	OL
10	P607	Current limit value in automatic mode	ST
11	P607	Current limit value in manual mode	ST
12	P103	Limit values for speed in automatic mode	SL
13	P114	Limit values for speed in manual mode	SL
14	P607	Current limit value for emergency stop	ST
15	P608	Pitch of the emergency stop delay ramp	SE
16	P609	Speed at which the brakes are locked	SL
17	P610	Max. static position error	SS
18	P611	Lag errors relative to motor speed Max. lag = Motor speed * Kev_SI	SF
19	P612	Filter band width for torque signal	SW
20	P613	Damping factor (Zeta) of the 2nd order filter	SZ
21	P401	Scaling of the actual position check-back signal 1 = 16384 inc/rev, 2 = 8192 inc/rev 3 = 4096 inc/rev	OR
22	P614	Zero shift of the position check-back signal	OO
23	P401	Motor/Encoder direction of rotation TRUE: Positive direction = clockwise FALSE: Positive direction = counterclockwise	OD
24	P615	Overtemperature protection TRUE: Overtemperature protection active FALSE: Overtemperature protection not active	OW
25.. 49			
50		Flag non-standard motor (currently not used) TRUE: Non-standard motor FALSE: Standard motor	
51.. 56			







**Position\_Scaling (para. no. 21, scaling of the actual position check-back signal)**

The transferral value for Position\_Scaling (1..3) is derived from the value for puls/rot. entered under P401.

**Direction\_Flag (para. no. 23, motor/encoder direction of rotation)**

Only positive i.e. clockwise (Direction\_Flag = TRUE) values are given to the drive as a direction of rotation. If a spindle reverse is desired, it is executed within the rho3 (setting via parameter P401, measuring system evaluation or set point output number).

## 8 “Flying” measurement by means of probe input Special functions 43 and 44

In order to be able to measure with as short a delay as possible when an external signal is received, the “flying measurement by means of probe input” option has been developed. When the input signal is received, the actual position is stored with a potential reaction time of 0.01ms.

Two special functions are defined which help activate or deactivate flying measurement. Analogous to the “flying measurement” option, the value measured is written into the standard variable @MPOS and can be read within a user program.

### Syntax of special function 43

<b>SPC_FCT: 43 = MEASURE_ON (</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>: KIN_NO</b>
	<b>VALUE</b>	<b>INTEGER</b>	<b>: EDGE</b>
		<b>INPUT</b>	<b>: CHANNEL_NO )</b>

### Syntax of special function 44

<b>SPC_FCT : 44 = MEASURE_OFF (</b>	<b>VALUE</b>	<b>INTEGER</b>	<b>: KIN_NO</b>
	<b>VALUE</b>	<b>INTEGER</b>	<b>: EDGE</b>
		<b>INPUT</b>	<b>: CHANNEL_NO )</b>

### Signification of the parameters:

**KIN\_NO** : Number of the kinematic for which the function is to be activated.

**EDGE** : Defines the value at which the function is executed.  
Possible values are 0 or 1

**CHANNEL\_NO** : Defines the probe input of the servo board

The channel number can be queried as to whether or not a measurement has taken place (see example). If measurement has taken place, there will be a one in the channel number. After this, @MPOS can be read and further processed. If no measurement has taken place, the probe logic is deactivated. This occurs with the special function probe\_off.

If no measurement has taken place in the active program, @MPOS is not occupied, i.e. when reading from @MPOS, the runtime error 'point not defined' occurs.

A maximum of one probe is available per servo board. The desired probe input is coded via the channel number.

e.g.: 610 signifies probe input on the 1st servo board

The second transfer parameter of the special function *MEASURE\_IN* is required in order to be able to indicate with which edge, positive = 1 or negative = 0, the actual position is stored.

Program example :

```
;;Control = Rho3  
;;Kinematics: (1=Robi1)  
;;Robi1. JC_NAMES = A1,A2,A3  
;;Robi1. WC_NAMES = K1,K2,K3
```

Program measure

```
Const:   SPC_NR1  = 43,  
         SPC_NR2  = 44,  
         Pos_edge = 0,  
         Neg_edge = 1
```

Robi1.JC\_point : @P1

Input: 610 = Probe 1

```
SPC_FCT : SPC_NR1 = Probe_e (value integer : kin_no value integer : edge  
input          : channel_number)  
SPC_FCT : SPC_NR2 = Probe_a (value integer : kin_no  
input          : channel_number)
```

BEGIN

```
    Probe_e (1, pos_edge, probe 1)  
    .  
    .  
    .  
    If probe 1 = 1 then @P1 = @MPOS  
        otherwise probe_a (1, probe 1)
```

Program\_end

## 8.1 Restrictions

The probe input only works with incremental encoders.

The precondition for the full acceptance of actual values is that all axes of a kinematic are located on the servo board on which the probe is defined. Only those axis positions which are located on the servo board selected via the channel number are accepted. Accessing an unassigned component of @MPOS results in the run-time error

**“point not defined”.**

## **9 Bit coupler with I/O board in the RC cardrack I/O configurations 15 and 16**

The rho3 control system function and parameterisation described below allows the employment of a standard I/O module in the RC rack with RC-PLC coupling via bit coupling as well. The signals supplied to the I/O board can also be RC system signals (e.g. reference point switches).

This makes it possible to standardise the circuitry of an RC control cabinet to RC-I/O and, if required, to use the RC with a PLC bit coupling - without changing the circuitry.

### **9.1 Description of function**

The I/O signals of the RC-I/O module must be transmitted via bit couplers to the PLC, i.e. the number of bytes to be transmitted to the PLC is raised by the number of input and output bytes of the I/O module.

For this reason a new I/O module configuration is necessary which is set up in machine parameter P20.

### **9.2 Setting machine parameters**

The "P20 I/O module configuration" parameter is enhanced by the following configurations:

Configuration 15:

CL300/CL500 range coupling and additional I/O boards in the RC cardrack coupled via PLC

Configuration 16:

Bit coupling to PC600 and additional I/O board coupled via PLC

#### **Configuration 15: CL300/CL500 range coupling**

The "P21 address range for PLC bit coupling" parameter is set as beforehand.

Default values:

I.START: 0 Start address of the RC outputs in the PLC input range.

I.END: 44 End address of the RC outputs in the PLC input range.

O.START: 0 Start address of the RC inputs in the PLC output range.

O.END: 44 End address of the RC inputs in the PLC output range.

In addition, 8 more bytes are transmitted in the PLC input range and 5 more in the PLC output range (for I/O board 64/40).

**Configuration 16 : Bit coupling to PC600**

The “P21 address range for PLC bit coupling” parameter is set as follows.

- I.START: 0 Start address of the RC outputs in the PLC input range.
- I.END: 87 End address of the RC outputs in the PLC input range.
- O.START: 0 Start address of the RC inputs in the PLC output range.
- O.END: 87 End address of the RC inputs in the PLC output range.

**9.3 Address range**

For an explanation of the assigned addresses on the PLC side, the possible starting values are described in the following table.

I/O module configuration	15	16
Start of the input range	0	0
End of the input range	44	87
Start of the output range	0	0
End of the output range	44	87
PLC INP. I/O module	I45.0-I52.7	I88.0-I95.7
PLC OUTP. I/O module	O45.0-O49.7	O88.0-O92.7

**Note:**

If the start address is altered, the end address and the addresses of the I/O modules must then also be altered in the same way.

**Example:** Shift of addresses by 10 bytes

I/O module configuration	15	16
Start of the input range	10	10
End of the input range	54	73
Start of the output range	10	10
End of the output range	54	91
PLC INP. I/O module	I55.0-I62.7	I74.0-I81.7
PLC OUTP. I/O module	O55.0-O59.7	O92.0-O96.7

**Note on configuration 16:**

With this variant the transmission of MTB and panel signals has been left out, as there is no corresponding functionality in the RC.

**9.4 Restrictions**

The maximum end address now lies at

- 247** for the PLC input range and
- 250** for the PLC output range

If an RC-I/O module with 32 inputs and 24 outputs is used, an 8 byte PLC input range and a 5 byte PLC output range must still be reserved.

## 10 Deleting the write/read buffer

### 10.1 Introduction

Communication between two units via a serial interface often involves the problem that at the start of communication, characters from a previously aborted communication or from connection disturbances can be stored and can then lead to disturbances or misinterpretations.

For this reason it should be possible to be able to delete characters from the BAPS program which have been temporarily stored in the control, without having to use the interface blocking signals.

### 10.2 Command syntax

The language elements READ\_BEGIN and WRITE\_BEGIN were so far used only with files are now used for the above purpose as BAPS language elements.

**READ\_BEGIN**            <File variable> [, line number]

or            **WRITE\_BEGIN**            <File variable> [, line number]

A unit name (PHG, V24\_1 .. V24\_4) or a file variable which has been allocated a unit with the ASSIGN function can be used as a file variable. If no file variable is entered, the BAPS compiler takes the PHG as the default.

The indication of a line number possible with files is not evaluated here.

### 10.3 Deleting the write buffer

During writing on an interface characters can be stored

- on the interface processor
- in the buffer memory of the output process belonging to the process
- in the buffer memory of the output processes belonging to other processes and
- in the buffer memory of the communication partner

The characters potentially stored in the buffer memory of the communication partner cannot be deleted by the command **READ\_BEGIN**. In such cases commands (reset, restart) should be provided during communication to inform the communication partner that it should delete its stored data.

The characters potentially stored in other output processes are not deleted. If these characters are to be deleted, the appertaining process must be stopped.

Outputs of the same process which have already been prepared are deleted.

Outputs which have already been passed on to the interface processor are likewise, as far as is possible, deleted. This will lead to cut-off outputs.

**NOTE:**

When the command **WRITE\_BEGIN** is called up, the **STATUS** of the unit for this process is set to zero.

If the protocol 3964R is set with this unit, the prepared outputs are deleted.

Outputs of the driver process are not cut off, but are fully executed in order to keep complications on the drive level minimal. No repetitions are made, however, which may be required as a result of the protocol. If data transmission has not yet begun, transmission is ended, even after a successful connection configuration.

An error in transmission will no longer change the **STATUS**, it remains zero.

It should be remembered when communication is restarted that the driver process aborted with **WRITE\_BEGIN** is allowed 4 secs. (repetition delay time) for repetitions if a repetition would have been necessary.

## 10.4 Deleting the read buffer

During reading from an interface, characters can be stored

- in the buffer memory of the communication partner (e.g. inhibited by handshake) or
- on the interface processor or
- in a driver process (3964R)

The characters stored in the buffer memory of the communication partner cannot be deleted by the command **READ\_BEGIN**.

It may also be the case that unexpected characters are received from the communication partner even after execution of the **READ\_BEGIN** command, e.g. answers from previously deleted actions with long reaction or processing times.

Characters on the interface processor are only deleted when the interface allocated to the unit (PHG, V24\_1 .. V24\_4) is not occupied by other processes i.e. no interface switching is active.

**NOTE:**

If a read buffer is deleted just as coherent data are being transmitted, this may lead to misinterpretations of the rest of the data or to protocol errors.

For this reason, if the protocol 3964R is set in the unit, only the input buffer of the driver process is deleted, but not the characters on the interface processor. The driver receives these characters and interprets them according to the protocol.

When the command **READ\_BEGIN** is called up, the **STATUS** of the unit for this process is set to zero.



## 10.5 Availability/restrictions

### ATTENTION !

The option can be compiled perfectly with the BAPS compiler 2.0, but is only available on the control from the robot control version 5 (TO05x) upwards.

If the commands READ\_BEGIN and WRITE\_BEGIN are executed with the robot control versions 1–3 (e.g. TO01J, TO02J, TO03G), this leads to various system errors, e.g. system error 83 : “invalid device address”, or system error 22: “AT SEMA\_RESET: DEV-I/O”. With RBS version 4 (e.g. TO04J), no errors are expected to arise, but no function will be executed either.

If the “files I/O” option in versions 1–4 is disabled, only the runtime error message “option not found” appears, while from version 5 the “deletion of write/read buffers” can be executed even if the “files I/O” option is not active.

## 10.6 Examples

```
1 PROGRAM rcmaster
2     FILE: barcodereader
3     PUBLIC INTEGER : barcode
4 BEGIN
5     ASSIGN barcodereader, "V24_1. "
6
7     READ_BEGIN barcodereader
8     WRITE barcodereader, "send number"
9     READ barcodereader, barcode
10
11 PROGRAM_END
```

```

1  ;;PROCESS_TYPE = PERMANENT
2  PROGRAM rcslave
3      FILE : host computer
4      PUBLIC INTEGER : parts_no, number_parts, store_no
5      INTEGER : stat_hc
6  BEGIN
7      ASSIGN host computer, "V24_1. "
8
9      READ_BEGIN host computer
10
11     sta:
12         READ host computer, parts_no, number_parts , store_no
13         stat_hc = STATUS (host computer)
14         IF stat_hc < 0 then
15             BEGIN
16                 WAIT 0.5 ; in case HC still sends something
17                 READ_BEGIN host computer
18             END
19         JUMP sta
20 PROGRAM_END

```

```

1  ;;PROCESS_TYPE = PERMANENT
2  PROGRAM wri_sta
3      INPUT :      1 = abort
4      INTEGER :      stat_v24_1
5  BEGIN
6      WRITE V24_1, "first output"
7      stat_v24_1 = STATUS (V24_1)
8
9      WRITE V24_1, "second", ..
10
11     IF abort = 1
12     THEN BEGIN
13         WRITE_BEGIN V24_1
14         WRITE V24_1, '--- abort ---'
15     END
16
17 PROGRAM_END

```

**Explanation of program WRI\_STA:**

In case of abort, the WRITE\_BEGIN command in line 13 deletes the prepared output in line 9.

Without the STATUS query in line 7, the output in line 6 would be deleted as far as possible. The output could then, for example, appear as: "fir--- abort ---"

## 11 Disabling individual axes

If several kinematics are driven by a rho3, circumstances may make it necessary to disable individual axes in order to service or repair them (e.g.: change of encoder). An opportunity must then be taken to disable individual axes.

### 11.1 Function

The individual axes can be disabled via the RC input signals “disable axes, axis 1..20” (no. 216..235, byte 26.1..byte 28.3).

The “1” signal then means that the axis concerned has been disabled, i.e. error monitoring (for example, measuring system monitoring), is no longer performed for this axis. The output at the set point output is 0 V. Internally the drive-on-enable for this axis is removed, so that error messages and monitoring are activated as in the option description “interruptible axes”, i.e. all axes not disabled can be moved as normal. As soon as a disabled axis is to be moved by manual motion (jog mode) or in automatic mode, the error message “drive on not available” is released.

The axes concerned are active at the “0” signal.

A switch-over of the signal from “0” to “1” is immediately activated when the signal is changed. The axis shutdown remains active until a control start-up is executed with a restored signal (**status “0”**).

If one or more axes have been disabled, the following warning is displayed under MODE 7.2:

Axis disabled  
Axis x  
Code 1408..1427

A referencing process does not have to be carried out for disabled axes.

### 11.2 Safety notes

**If several axes are disabled, it must be ensured through external switching (e.g. switching off the power) that these axes can no longer move, as a monitoring process (measuring system error, servo error) is no longer being conducted by the rho3.**



## 13 Asynchronous inputs

With previous rho3 operating systems, a query of inputs such as, for example, in the **IF .. THEN** command would lead to an internal synchronisation between the block-preparing motion task (user process) and the kinematically-related block execution (control of the axes). The control attempts to prepare as many kinematically-related blocks as possible (up to 11) in parallel to the execution of a traverse block. Normally, block preparation will lead block execution. Calculation of arithmetic expressions is thus executed at a much earlier point than the execution of MOVE blocks which are located in front of these arithmetic operations in the BAPS program. This asynchronicity must be cleared when using inputs, i.e. synchronisation must take place as soon as an input is read in. The **ASYNCHRONOUS INPUTS** option allows this asynchronicity to be retained during input queries. This means that inputs declared asynchronous are read in at the time of block preparation and immediately processed. This generally leads to a time shift during execution of sequential BAPS commands. Therefore, asynchronous inputs should only be used if other BAPS solutions are not sufficient.

### 13.1 Declaration

The declaration of asynchronous inputs takes place when an offset of **1000** is given in the channel number. The following table shows the permitted input types and their channel numbers for synchronous and asynchronous input queries.

Input type	Channel number (synchronous)	Channel number (asynchronous)
BINARY	1 .. 120	1001 .. 1120
DEC	201 .. 224	1201 .. 1224
INTEGER	401 .. 408	1401 .. 1408

The following is supposed to demonstrate the use of asynchronous inputs in BAPS, using the group of binary inputs as an example.

The other input types should be used in the same way. Declaration of asynchronous binary inputs takes place via channel numbers **1001 .. 1120**. With these channel numbers, the inputs operated are physically the same as with channel numbers 1 .. 120.

**INPUT :**    **1001 = ASYN\_INP\_1,**  
                 **1002 = ASYN\_INP\_2**

### 13.2 Asynchronous inputs in the BAPS program

The same BAPS syntax applies for the use of the inputs in the BAPS program as for the normal, binary inputs. It is also allowed to declare the same physical input once as a normal (synchronous) input and in addition as an asynchronous input. In the BAPS program the input can selectively be queried with or without synchronisation.

Example :

```
1      PROGRAM SYN_ASYN
2      INPUT : 1 = I1, 1001 = I1_ASYN
3      BINARY : BIN_1, BIN_2
4      BEGIN
5      MOVE LINEAR VIA P1
6      BIN_1 = I1
7      MOVE LINEAR VIA P2
8      MOVE LINEAR VIA P3
9      MOVE LINEAR VIA P4
10     BIN_2 = I1_ASYN
11     MOVE LINEAR VIA P5
12     MOVE LINEAR VIA P6
13     MOVE LINEAR VIA P7
14     PROGRAM_END
```

In the example the binary input 1 is when first used read in synchronously to the program sequence and assigned to variable BIN\_1. Block preparation is halted between line 5 and line 7. Reading in the input takes place 2 interpolation cycles (machine parameter P5) before the set point output of the end point P1 (synchronous to movement). Blocks P2, P3, P4, P5, P6, P7 are not prepared until the input has been read in.

When used for the second time, the input is used as an asynchronous input. Block preparation does not wait here until P4 has been approached. The input is already being read in while the kinematic is being traversed to point P2. Blocks P5, P6, P7 can likewise be already in preparation while the kinematic is moving to point P2, if the user task receives sufficient computing time.

### 13.3 Application example of asynchronous inputs

When employing the control with quick **packing machines**, the option offers a means of reacting, without set point intrusions, to external events after a settable number of traverse blocks. With the help of the **MOVE\_FILE** special function, a curve shape stored in BNR files can be traversed with this application (see **MOVE\_FILE** specification). The values of this file produce a fixed speed profile connected to the preset interpolation cycle. Using an **asynchronous input** also enables a smooth switch-over (i.e. without set point intrusion) to another BNR file in the traverse loop of the user program. **Block search** (see special function description **46 BLOCK\_SEARCH**) is limited in the example to 5 blocks. This means that the change in speed becomes effective after 5 **MOVE\_FILE** blocks. If a normal, binary input is used instead of the asynchronous input, it is highly probable that set point intrusions will arise, as the block search then synchronises the time of read-in of the input.

The following rule of thumb applies when using **ASYNCHRONOUS INPUTS** :

The larger the blocksearch set, the lesser the probability of set point intrusions and the longer the time taken for the input to affect the kinematic blocks.

<b>Block search</b>	<b>Probability of set point intrusions:</b>	<b>Response time to ASYNCHRONOUS INPUTS:</b>
<b>Large block search</b>	<b>low</b>	<b>slow</b>
<b>Small block search</b>	<b>high</b>	<b>fast</b>

Example :

```
;;CONTROL = RHO3
;;KINEMATICS : (1 = VP_MACHINE)
;;JC_NAMES = A1, A2, A3, A4
```

PROGRAM ASYN\_IN

```
SPC_FCT : 45 = MOVE_FILE ( VALUE INTEGER : KIN_NO
                           BNR_FILE : CURVE_X
                           VALUE INTEGER : BASE
                           JC_POINT : @MODULO_FLAG
                           VALUE FIELD[1..6] INTEGER : RESERVE )
```

```
SPC_FCT : 46 = BLOCK_SEARCH ( VALUE INTEGER : KIN_NO
                              VALUE INTEGER : NUMBER_BLOCKS)
```

```
INPUT : 2 = I2,
       1001 = V_SWITCH
```

```
BNR_FILE : FD_1, FD_2, FD_3,
          FD_1_L, FD_2_L, FD_3_L, ;* slow
          FD_1_S, FD_2_S, FD_3_S, ;* fast
```

```
JC_POINT : @MOD_FLAG
FIELD[1..6] INTEGER : RESERVE
INTEGER : I
```

```
CONST : PL_CYCLES = 2,
        HEADER_LENGTH = 80
```

BEGIN

```
;;KINEMATIC = VP_MACHINE
```

```
BLOCK_SEARCH (1, 5) ; * 5 blocks block search for kinematic 1
```

```
MOVE WITH V_PTP VIA @(57, 0, 12, 12)
```

```
;***** open BNR files *****
```

```
READ_BEGIN FD_1_L, HEADER_LENGTH
READ_BEGIN FD_1_S, HEADER_LENGTH
READ_BEGIN FD_2_L, HEADER_LENGTH
READ_BEGIN FD_2_S, HEADER_LENGTH
READ_BEGIN FD_3_L, HEADER_LENGTH
READ_BEGIN FD_3_S, HEADER_LENGTH
```

```
***** traverse cycle *****
loop:
IF V_SWITCH = 1
  THEN BEGIN (*----- stop files for high speed -----*)
    @MOD_FLAG = @(1, 0, 0, 0)
    MOVE_FILE (1, FD_1_S, PL_CYCLES, @MOD_FLAG, RESERVE)

    @MOD_FLAG = @(0, 0, 1, 1)
    MOVE_FILE (1, FD_2_S, PL_CYCLES, @MOD_FLAG, RESERVE)

    @MOD_FLAG = @(0, 1, 0, 0)
    MOVE_FILE (1, FD_3_S, PL_CYCLES, @MOD_FLAG, RESERVE)
  END

  ELSE BEGIN (*----- stop files for low speed -----*)
    @MOD_FLAG = @(1, 0, 0, 0)
    MOVE_FILE (1, FD_1_L, PL_CYCLES, @MOD_FLAG, RESERVE)

    @MOD_FLAG = @(0, 0, 1, 1)
    MOVE_FILE (1, FD_2_L, PL_CYCLES, @MOD_FLAG, RESERVE)

    @MOD_FLAG = @(0, 1, 0, 0)
    MOVE_FILE (1, FD_3_L, PL_CYCLES, @MOD_FLAG, RESERVE)
  END

IF I2 = 0 THEN JUMP LOOP

***** close BNR files *****
CLOSE FD_1_L CLOSE FD_2_L CLOSE FD_3_L
CLOSE FD_1_S CLOSE FD_2_S CLOSE FD_3_S

PROGRAM_END
```



## 14 Change of function operation Directly approaching points in the Teach in mode

From software version TO51 onwards, operation of the “**directly approaching points in the Teach in mode**” function is only possible as described below.

The function facilitates the easy changing of points already taught in. For example, when a packing system is commissioned the removal positions and deposit positions have already been taught in. Assume the user would like to make slight corrections to these positions. Because the positions are far apart, it is time-consuming to manually move from the removal position to the deposit position. Using a traverse command, the user can move directly to the deposit position which results in considerable time savings.

### 14.1 Operation

Select Teach in mode.

Select Teach in mode on the PHG with one of the following:

- mode 3.1.3 (Programming, BAPS program, Teach in),
- mode 4.2 (Define, Teach in)

or:

- mode 5.14 (BAPS program Test, Teach in).

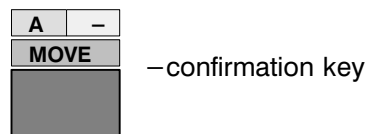
The desired point is selected after the Teach in mode is activated.

Scroll with the following key to go to the point where the selection is to be made:



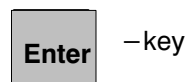
Or, enter the name of the point.

After selecting the point, press



press the confirmation key on the PHG,

and confirm with



movement to the desired point is started.

The robot approaches the coordinate values (already taught in) which are set in the point.

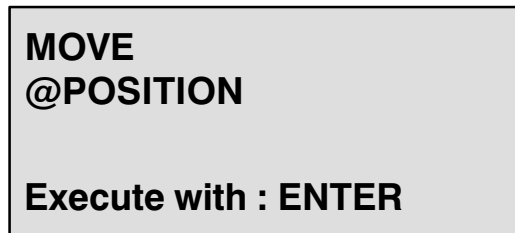
The movement is immediately aborted as soon as the confirmation key or

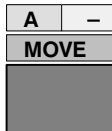


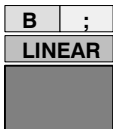
**As of Version TO05I the Enter key must be copied from O6.5 to I2.7 in the PLC program** (is included in the new standard PIC program).

After an interruption, the movement can only be started as above through reselection.

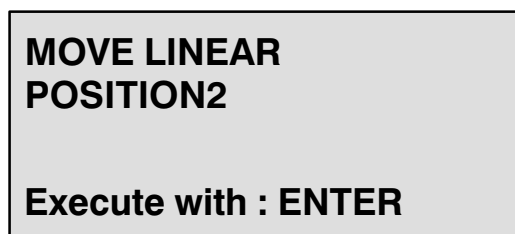
**PHG display:**



If the movement is to be linear, in addition to the  key,

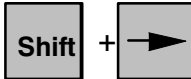
the  key on the PHG must also be pressed.

**PHG display:**



Speed and acceleration are derived from the machine parameters for JOG speeds **P100** group.

The speed override for VFACTOR, AFACTOR and DFACTOR can be set with PHG Mode 11.4,5,6 “Help functions”, as well as with the interface (PIC, PLC).

Using the key combination , the operator can switch to the position display during the movement operation.

When the robot has reached the point, a new value can be taught in for the selected point.

**14.2 Restrictions**

The direct approach of a teach point should be linear. If the robot (for example SR450) is changed from right-handedness to left-handedness (i.e. change of sign on axis 1), the rho 3 issues the message: “point not reachable”. In this case, PTP must be activated to approach the desired point.

### 14.3 Preventing points which have been taught in from being overwritten

When the user is teaching in points and he/she presses the ENTER key, the message

**“pnt. overwr.:ENTER”**

appears to inform him/her that the currently selected point will be overwritten if he presses the ENTER key again. If any other key is pressed, the old point value is retained.

## 15 Expansion of the function "fast, smooth start-off"

Also refer to the description of the function "Fast, smooth start-off" in chapter 2 of this document, and "Expansion of the function "Fast, smooth start-off"" in chapter 4 of this document.

As of operating system version TO05, a defined delay time was added to this function, i.e. a user-selected delay time may be programmed after the command 'WAIT UNTIL high-speed input' before the next traverse block is executed.

Example:

```
.  
.
1      MOVE UNTIL HS_31 = 1 MAX_TIME=0.15
2      MOVE DESTPOS
.  
.
```

When the wait condition is fulfilled, the traversing movement of block (2) is started with a delay of 0.15 seconds (150 milliseconds).

The accuracy resolution of MAX\_TIME depends on the position controller cycle (servo board runtime), i.e. MAX\_TIME should be an integer multiple of the position controller cycle.

### NOTE:

For further information on the software TO0x, please refer to the following TXT files in your ROPS3 directory:

README.TXT	Current INFO's on BAPS, compatibility (software versions: rho – ROPS – PROFI), BATCH files
READPLC.TXT	Current INFO's on BAPS/PIC, standard PIC programs, PROFI support
HARDLOCK.TXT	INFO's on the software protection adapter
LIESMICH.TXT	(Information of README.TXT in German)
LISSPS.TXT	(Information of READPLC.TXT in German)

**Index****A**

absolute measuring system, 5 – 24  
ASC\_ARRAY, 2 – 41  
ASSIGN, 4 – 31

**B**

BATCH–Dateien, 2 – 62  
  Edi.BAT, 2 – 64  
  Info.BAT, 2 – 63  
  PUeb.BAT, 2 – 64  
  RLad.BAT, 2 – 63  
  RLis.BAT, 2 – 63  
  RLoe.BAT, 2 – 63  
  RNeN.BAT, 2 – 63  
  RSet.BAT, 2 – 64  
  RSpe.BAT, 2 – 64  
  RSpeUe.BAT, 2 – 64  
  RUeb.BAT, 2 – 64

**C**

CAN, 5 – 24  
  Module input, 5 – 24  
  Pulses per rotation, 5 – 25  
CLS, 3 – 31

**E**

EEPROM, User memory in, 4 – 7  
EXCLUSIVE, 2 – 42  
EXCLUSIVE\_END, 2 – 42  
EXTERNAL, 2 – 42  
external, program selection, 2 – 47

**G**

go block, limitation, 5 – 44

**I**

incremental measuring system, Queries in P401,  
  5 – 23  
Inputs  
  asynchronous, 5 – 45  
  fast, 4 – 12  
INT\_ASC, 2 – 40  
Interpolator, Stop, 4 – 27

**M**

Module number, 5 – 23

**P**

Plug number, 5 – 23  
Potentiometer measuring system, 5 – 25  
PUBLIC, 2 – 42

**R**

Resolver, orientationally correct referencing, 5 – 24,  
  5 – 25

**S****SAFETY NOTES**

A/D/V factor, 4 – 15  
Max. axis speed, 4 – 28  
Memory test, 4 – 16  
Options, 4 – 17

Schnelles, konstantes Losfahren, Erweiterung,  
  4 – 52

SEMAPHORE, 2 – 42

servo card, 5 – 23

**SPECIAL FUNCTION**

Spec. 29, 4 – 18  
Spec. 30, 4 – 21  
Spec. 31, 4 – 22  
Special function 4; Call system function, 4 – 35  
Special function 15, 4 – 47  
Special function 21, 4 – 42

Special function, 5 – 35

1,2 = set and reset outputsignals, 3 – 27  
3 = setting machine position, 3 – 24  
41 = asynchronous speed change, 5 – 5  
42 = asynchronous target stipulation, 5 – 5  
45 = MOVE\_FILE, 5 – 14  
46 = start block limitation, 5 – 44

**STANDARD PROCEDURE**

ASC\_INT, 2 – 41  
INT\_ASC, 2 – 40

Strobe, INTEGER inputs, 4 – 16

SYNCHRONOUS BELT, without parallel belt axis,  
  4 – 49

Synchronous belt, Synchronization type 3, 4 – 54

**T**

Transformation, Forwards,, 3 – 25

**V**

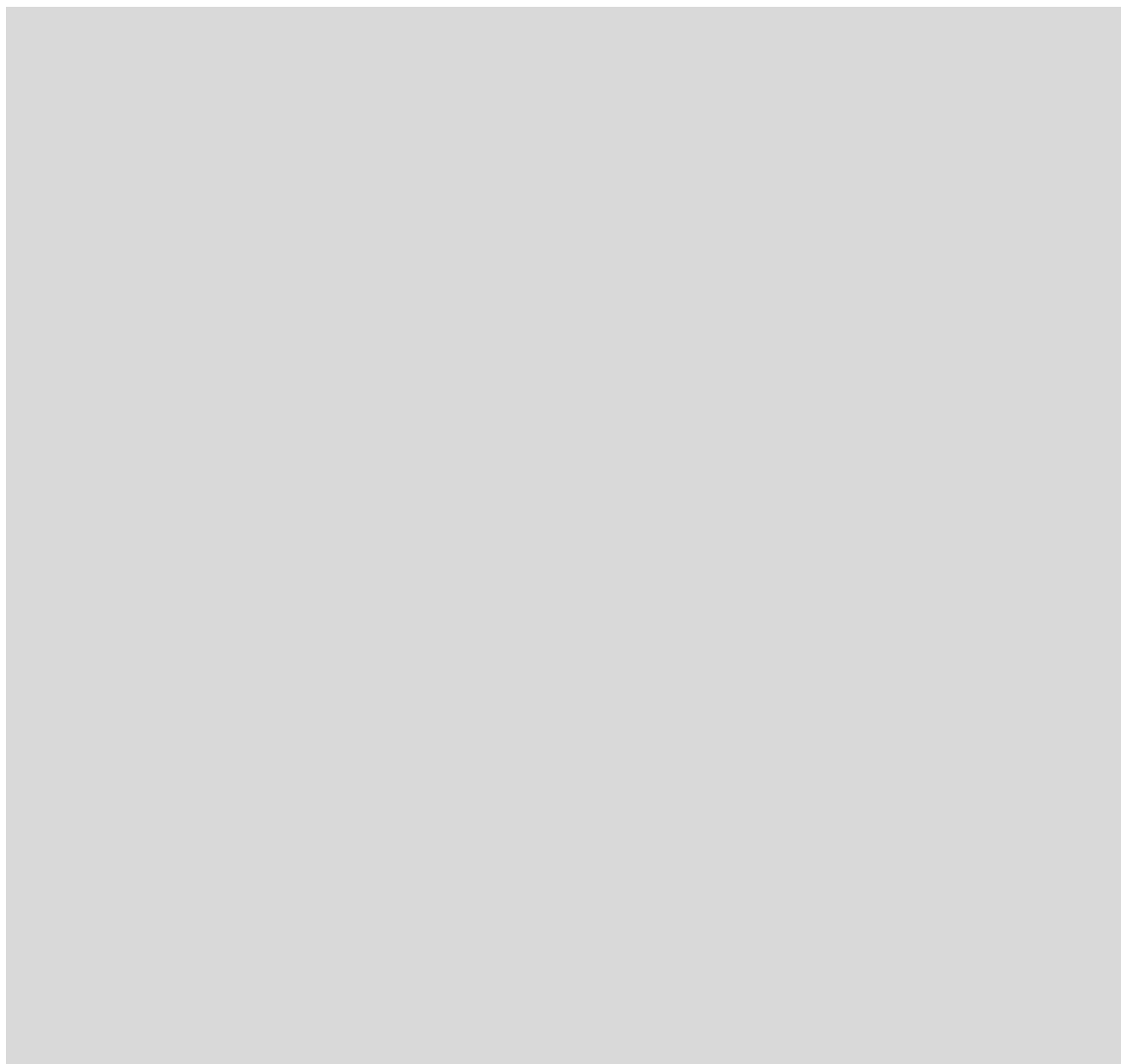
VERSION, 3 – 31

**W**

WARTE BIS, MAXZEIT, 4 – 52

rho 3

# Releases Part 3 Versions TO06, TO07 and TO08



Version

# 107



*rho 3*

# Releases Part 3 Versions TO06, TO07 and TO08

1070 073 078-107 (97.01) GB



Reg. Nr. 16149-03

© 1995 - 1997

by Robert Bosch GmbH,

All rights reserved, including applications for protective rights.  
Reproduction or handing over to third parties are subject to our written permission.

Discretionary charge 50.- DM



## **Safety information and additional literature**

Before you start working with the rho 3, we recommend that you thoroughly familiarise yourself with the contents of this manual. Keep this manual in a place where it is always accessible to all users.

### **Standard operation**

This instruction manual presents a comprehensive set of instructions and information required for the standard operation of the described products. The products described hereunder are used for the purpose of programming and operating the rho 3.

The products described hereunder

- were developed, manufactured, tested and documented in accordance with the relevant safety standards. In standard operation, and provided that the specifications and safety instructions relating to the project phase, installation and correct operation of the product are followed, there should arise no risk of danger to personnel or property.
- are certified to be in full compliance with the guidelines on electromagnetic compatibility (89/336/EEG, 93/68/EEG and 93/44/EEG). Specific compliance with harmonized standards EN 50081-2 and EN 50082-2 is hereby certified.
- are designed for operation in an industrial environment. Prior to the intended installation and/or operation within a private residence or business area, on retail premises or in a small-industry setting, the user will be required to obtain a single operating license issued by the appropriate national authority or approval body. In Germany, this is the Federal Institute for Posts and Telecommunications, and/or its local branch offices.

### **Qualified personnel**

This instruction manual is designed for specially trained PLC personnel. The relevant requirements are based on the job specifications as outlined by the German Electrical and Electronics Manufacturers' Association (ZVEI). Please refer to the following German-language publication:

**Weiterbildung in der Automatisierungstechnik**

**Hrsg.: ZVEI und VDMA**

**MaschinenbauVerlag**

**Postfach 71 08 64**

**60498 Frankfurt**

This instruction manual is specifically designed for RC specialists.





Interventions in the hardware and software of our products which are not described in this instruction manual may only be performed by our skilled personnel.

Unqualified interventions in the hardware or software or non-compliance with the warnings listed in this instruction manual or indicated on the product may result in serious personal injury or damage to property.

Qualified personnel are persons who

- as **planning personnel**, are familiar with the safety guidelines used in electrical engineering and automation technology.
- as **operating personnel**, are familiar with the equipment used in the field of automation technology and are thus familiar with the operating instructions in this manual.
- as **commissioning personnel**, are authorized to commission, ground and classify electric circuits and devices/systems in accordance with the relevant safety standards.

## Safety instructions on the control components

The following warnings and notices may be indicated on the control components themselves and have the following meaning:



Danger: High voltage!



Danger: Battery acid!



Electrostatically-sensitive components!



Disconnect at mains before opening!



Pin for connecting PE conductor only!



This connection for functional earthing or low-noise earth only!



For screened conductor only!

**Safety instructions in this manual**

These symbols are used throughout this manual subject to the following conditions.

**DANGER**

This symbol is used to warn of the presence of **dangerous electrical current**. Insufficient or lacking compliance with these instructions can result in **personal injury**.

Safety instructions accompanied by this symbol are serially numbered, for example 0.1. The appendix provides translations of the safety notes shown here in all the official EC languages.

**DANGER**

This symbol is used wherever an insufficient or lacking compliance with instructions can result in **personal injury**.

Safety instructions accompanied by this symbol are serially numbered, for example 0.1. The appendix provides translations of the safety notes shown here in all the official EC languages.

**CAUTION**

This symbol is used wherever an insufficient or lacking compliance with instructions can result in **damage to equipment or files**.

Safety instructions accompanied by this symbol are serially numbered, for example 0.1. The appendix provides translations of the safety notes shown here in all the official EC languages.



This symbol is used to inform the user of special features.



**We would greatly appreciate any contributions to improve this manual. If you have any suggestions, please fill out the page provided at the end of this manual.**



---

## Safety instructions

**DANGER****0.1**

**Danger to persons and equipment!**

**Test every new program before operating the system!**

**CAUTION****0.2**

**Danger to the module!**

**Do not insert or remove the module when the control is switched on! This can destroy the module. Switch off or remove the power supply module of the control, external power supply and signal voltage before inserting or removing the module!**

**CAUTION****0.3**

**Danger to the module!**

**All ESD protection measures must be observed when using the module! Avoid electrostatic discharges!**

Observe the following protective measures for electrostatically endangered modules (EEM)!

- The employees responsible for the storage, transport and handling must be trained in ESD protection.
- EEMs must be stored and transported in the protective packaging specified.
- EEMs may basically only be handled at special ESD work places set up specifically for this purpose.
- Employees, work surfaces and all devices and tools, which could come into contact with EEMs must be same potential (e.g. earthed).
- Wear an approved earthing strap around your wrist. The grounding bracelet must be connected via a cable with integrated 1 M $\Omega$  resistance with the work surface.
- EEMs may on no account come into contact with chargeable objects, these include most plastics.
- When inserting EEMs into devices and removing them, the power source of the device must be switched off.

## Contents Part 3

### TO06

<b>1</b>	<b>Workpiece coordinate system</b>	
1.1	General .....	6 – 1
1.2	BAPS2 Syntax .....	6 – 2
1.2.1	Declaration of variables .....	6 – 3
1.2.2	Assignments in BAPS .....	6 – 3
1.2.3	Changing over the coordinate system .....	6 – 4
1.3	Machine parameter P313: WCSYS–ROB– ASSIGNMENT .....	6 – 4
1.4	Selection and effect of a workpiece coordinate system in the movement program .....	6 – 6
1.4.1	World coordinate points (WC points) .....	6 – 6
1.4.2	Joint coordinate points (JC points) .....	6 – 6
1.4.3	POS , @POS , @MPOS .....	6 – 7
1.4.4	LIMIT_MIN, LIMIT_MAX .....	6 – 7
1.4.5	Standard functions JC ( ) , WC ( ) .....	6 – 7
1.4.6	Special functions 1, 2: IO/PPO Logic .....	6 – 8
1.4.7	Special function 17: MIRROR .....	6 – 8
1.4.8	Translating the world coordinate system (P310) .....	6 – 8
1.4.9	Limit switch monitoring .....	6 – 8
1.4.10	Program end, abort, control start–up .....	6 – 8
1.4.11	Axis displays .....	6 – 9
1.4.12	Displaying the active WC_SYSTEM .....	6 – 9
1.5	Selection and effect in Manual mode .....	6 – 9
1.6	Safety instruction .....	6 – 10
1.7	Examples of special workpiece coordinates .....	6 – 11
1.7.1	Example 1: Pure height offset .....	6 – 11
1.7.2	Example 2: Height offset and rotation .....	6 – 12

<b>2</b>	<b>rho 3.0</b>	
2.1	Software PLC for the rho3.0 .....	6 – 13
2.2	Runtime display of the software PLC .....	6 – 13
2.3	Communication between rho3.0 – PLC .....	6 – 13
2.4	Using the high-speed inputs and outputs of the rho3.0	6 – 13
2.5	Double assignment of the PHG interface .....	6 – 14
2.6	New or expanded machine parameters .....	6 – 14

**TO07**

	Page
<b>1 PHG 2000 .....</b>	<b>7 – 1</b>
1.1 PHG 2000 modes .....	7 – 3
1.1.1 PHG 3 compatible mode .....	7 – 3
1.1.2 Transparent mode .....	7 – 3
1.1.2.1 Transparent mode with existing menu structure ....	7 – 3
1.1.2.2 Transparent mode with user-defined menu structure	7 – 4
1.2 Key assignment .....	7 – 4
1.2.1 Mask-specific key assignment .....	7 – 4
1.3 BDT editor .....	7 – 5
1.4 Configuration and test .....	7 – 6
<b>2 PHG 2000 System Variables .....</b>	<b>7 – 7</b>
<b>3 Controlling the PHG 2000 output</b>	
3.1 General information .....	7 – 15
3.1.1 Examples of cursor positioning for different character sets .....	7 – 15
3.2 BAPS2 declaration section .....	7 – 16
3.3 Calling subroutines in BAPS2 .....	7 – 17
3.4 Outputs to the PHG 2000 in subroutine technique ..	7 – 18
<b>4 Define / Teach In</b>	
4.1 Introduction .....	7 – 23
4.2 Display layout .....	7 – 23
4.2.1 "Define" layout .....	7 – 23
4.2.2 "Teach In" layout .....	7 – 25
4.3 Operation .....	7 – 26
4.3.1 Operation of "Define" .....	7 – 26
4.3.1.1 Operation of "Define" with three axes .....	7 – 27

4.3.2	Operation of "Teach In" .....	7 – 28
4.3.2.1	Operation of "Teach In" with three axes .....	7 – 28
4.3.2.2	Operation of "Teach In" with five axes .....	7 – 29
<b>5</b>	<b>Dateiliste (File list)</b>	
5.1	File list mask in the BDT editor .....	7 – 31
5.2	Creating the objects in the BDT editor .....	7 – 32
5.2.1	Heading "File list" on the PHG 2000 .....	7 – 32
5.2.2	Listbox for the file list .....	7 – 33
5.3	Displaying the memory allocation .....	7 – 34
5.4	PHG 2000 display .....	7 – 35
<b>6</b>	<b>Process Information</b>	
6.1	Mask in the BDT editor .....	7 – 37
6.2	Generating the objects in the BDT editor .....	7 – 38
6.2.1	Listbox for the process list .....	7 – 40
6.2.2	Global process information .....	7 – 41
6.2.3	Information on the selected process .....	7 – 41
6.3	PHG 2000 display .....	7 – 42
6.4	Stopping processes .....	7 – 43
<b>7</b>	<b>Variable assignment of PHG keys</b>	
7.1	Description .....	7 – 45
<b>8</b>	<b>Selecting a PKT file or point name</b>	
8.1	General .....	7 – 59
8.2	Declaration .....	7 – 59
8.3	Example .....	7 – 60
<b>9</b>	<b>Reversing an absolute measuring system</b>	
9.1	General .....	7 – 61
9.2	Function .....	7 – 61
9.3	Compatibility .....	7 – 61

**10 IAT Programming**

10.1	General .....	7 – 63
10.2	Requirements .....	7 – 63
10.2.1	Hardware .....	7 – 63
10.2.1.1	Switch setting on the IAT module .....	7 – 63
10.2.2	Software .....	7 – 64
10.2.2.1	Programming environment .....	7 – 64
10.2.2.2	Software for supporting PLC programming .....	7 – 64
10.3	Creating PLC programs .....	7 – 65
10.3.1	Recommended procedure .....	7 – 65
10.3.2	Special features .....	7 – 66
10.3.2.1	Data types .....	7 – 66
10.4	Executing the PLC program .....	7 – 66
10.4.1	Displays .....	7 – 66
10.5	Automatic start-up .....	7 – 67

**11 rho3 coupling via CAN bus**

11.1	Coupling of several rho3 controls .....	7 – 69
11.1.1	Linking the controls .....	7 – 69
11.1.2	Setting the identifiers .....	7 – 69
11.1.3	Structure .....	7 – 71

**12 Memory expansion with SRAM board**

12.1	General .....	7 – 73
12.2	Formatting the SRAM board .....	7 – 73
12.3	Inserting the SRAM board into the rho3.0 .....	7 – 73
12.4	Removing the SRAM board .....	7 – 73
12.5	Restrictions .....	7 – 74
12.6	Data protection .....	7 – 74



<b>13</b>	<b>PLC Interface expansion</b>	
13.1	General .....	7 – 75
13.2	Expanded PLC interface for rho3.0 withdrawable module version .....	7 – 75
13.2.1	PLC programming .....	7 – 76
13.2.2	Setting the rho3 machine parameters .....	7 – 77
13.2.2.1	P20 I/O module configuration .....	7 – 77
13.2.2.2	P21 address range for PLC bit coupling .....	7 – 77
13.3	Expanded PLC interface for rho3.0 (bit coupling) ...	7 – 77
13.3.1	PLC programming .....	7 – 78
13.3.2	Setting the rho3 machine parameters .....	7 – 78
13.3.2.1	P20 I/O module configuration .....	7 – 79
13.3.2.2	P21 address range for PLC bit coupling .....	7 – 79
13.4	Expanded PLC interface for rho3.1 bit coupling in the extended I/O field .....	7 – 79
13.4.1	PLC programming .....	7 – 80
13.4.2	Setting the rho3 machine parameters .....	7 – 80
13.4.2.1	P20 I/O module configuration .....	7 – 80
13.4.2.2	P21 address range for PLC bit coupling .....	7 – 80
13.5	Overview of the programs for the Expanded PLC interface .....	7 – 81
13.5.1	Programs for withdrawable rho3.0 module .....	7 – 81
13.5.2	Programs for rho3.1 bit coupler .....	7 – 81
13.5.3	Programs for rho3.1 bit coupler in the extended I/O field .....	7 – 81
<b>14</b>	<b>Expansion of the BAPS2 function CONDITION .....</b>	<b>7 – 83</b>

**TO08**

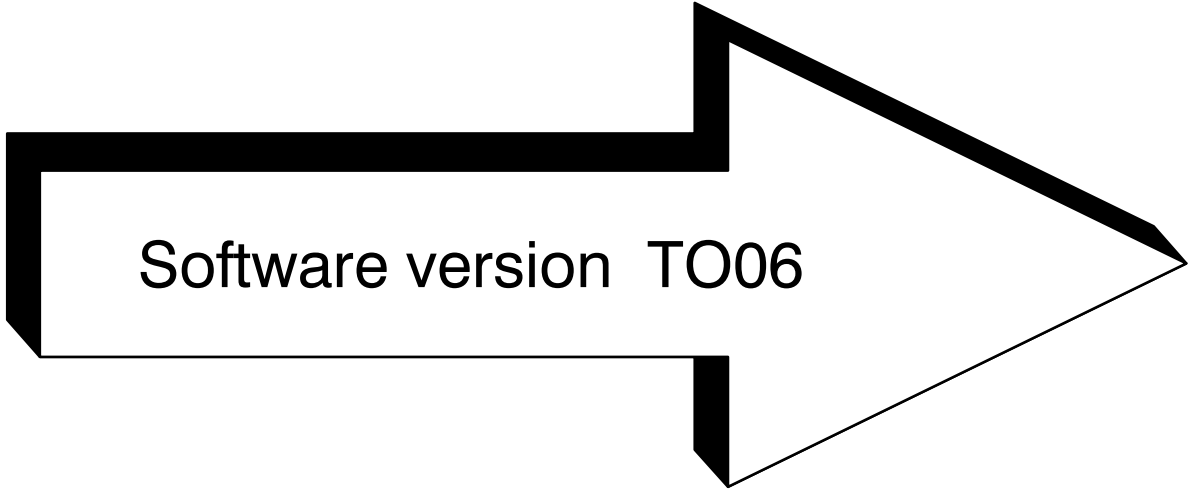
	Page
<b>1</b>	<b>BAPS FUNCTION "BREAK"</b>
1.1	BAPS syntax ..... 8 – 1
1.2	Special features ..... 8 – 2
<b>2</b>	<b>Multi-function I/O's</b>
2.1	General ..... 8 – 3
2.1.1	General conditions ..... 8 – 3
2.2	Multi-function I/O's via SoftPIC for stand-alone rho3.0 .. 8 – 3
2.3	Multi-function I/O's via PLC with withdrawable rho3.0 version ..... 8 – 4
2.4	Machine parameters ..... 8 – 4
<b>3</b>	<b>Initializing SRAM boards</b>
3.1	General ..... 8 – 5
3.1.1	General conditions ..... 8 – 5
3.2	Formatting and initializing the SRAM board ..... 8 – 6
3.2.1	Procedure for initializing an SRAM board ..... 8 – 6
3.2.2	Error messages ..... 8 – 7
3.3	Data protection ..... 8 – 9
3.4	Replacing the SRAM board ..... 8 – 10
<b>4</b>	<b>Belt synchronization with endless belt</b>
4.1	General ..... 8 – 11
4.2	Function ..... 8 – 11
4.3	Restrictions ..... 8 – 11
4.4	Program examples ..... 8 – 12
4.4.1	Example 1 ..... 8 – 12
4.4.2	Example 2 ..... 8 – 13

<b>5</b>	<b>Remote use of the rho3.0</b>	
5.1	General .....	8 – 15
5.2	PLC program examples for remote use of the rho3.0	8 – 16
<b>6</b>	<b>Start-up behavior of multiple rho3.0's (in PLC) .....</b>	<b>8 – 17</b>
<b>A</b>	<b>Annex</b>	
A.1	Abbreviations .....	A-1
A.2	Safety instructions .....	A-2
A.2.1	Dansk .....	A-2
A.2.2	Deutsch .....	A-4
A.2.3	Ελληνικά .....	A-6
A.2.4	English .....	A-8
A.2.5	Español .....	A-10
A.2.6	Français .....	A-12
A.2.7	Italiano .....	A-14
A.2.8	Nederlands .....	A-16
A.2.9	Português .....	A-18
A.2.10	Suomi .....	A-20
A.2.11	Svenska .....	A-22

## List of figures

Fig.		Page
6 – 1	Original coordinate system vs. Workpiece coordinate system .....	6 – 1
6 – 2	Axes and rotation in the workpiece coordinate system .....	6 – 2
7 – 3	PHG2000 Interfacing options .....	7 – 1
7 – 4	General view of the PHG 2000 .....	7 – 2
7 – 5	PHG display and key assignment .....	7 – 4
7 – 6	Define layout with three axes .....	7 – 23
7 – 7	Define layout with five axes .....	7 – 24
7 – 8	Teach In layout with three axes .....	7 – 25
7 – 9	Teach In layout with five axes .....	7 – 26
7 – 10	Operation of Define with three axes .....	7 – 27
7 – 11	Operation of Teach In with three axes .....	7 – 28
7 – 12	Operation of Teach In with five axes .....	7 – 29
7 – 13	BDT mask 'File list' .....	7 – 31
7 – 14	Editing variables with 'Variable bearbeiten' .....	7 – 33
7 – 15	Editing the text list with 'Textliste bearbeiten' .....	7 – 33
7 – 16	Variable for memory allocation .....	7 – 34
7 – 17	PHG2000 display .....	7 – 35
7 – 18	Process information: Mask in BDT editor .....	7 – 37
7 – 19	Process information texts .....	7 – 38
7 – 20	Process information variables .....	7 – 39
7 – 21	Process information – variable text liste .....	7 – 40
7 – 22	PHG2000 display .....	7 – 42
7 – 23	Table 1: PHG key index .....	7 – 49
7 – 24	Table 2/1: Standard key assignment on the normal level .....	7 – 50

7 – 25	Table 2/2: Standard key assignment on the normal level .....	7 – 51
7 – 26	Table 3/1: Standard key assignment on the SHIFT level .....	7 – 52
7 – 27	Table 3/2: Standard key assignment on the SHIFT level .....	7 – 53
7 – 28	Table 4/1: Standard key assignment on the ALT level	7 – 54
7 – 29	Table 4/2: Standard key assignment on the ALT level	7 – 55
7 – 30	Table 5/1: Standard key assignment on the level for PIC monitor and PIC editor .....	7 – 56
7 – 31	Table 5/2: Standard key assignment on the level for PIC monitor and PIC editor .....	7 – 57
7 – 32	Expanded PLC interface for rho3.0 .....	7 – 75
7 – 33	Expanded PLC interface for rho3.1 (bit coupler in I/O array) .....	7 – 77
7 – 34	Expanded PLC interface for rho3.1 (bit coupler in the EI/EO field) .....	7 – 79
8 – 1	CPU occupation with Break command .....	8 – 1
8 – 2	Standard interface assignment with MF I/O's for SoftPIC .....	8 – 3
8 – 3	Standard interface assignment with MF I/O's for withdrawable version .....	8 – 4
8 – 4	Remote operation of the rho3.0 (PLC coupling with PROFIBUS/DP) .....	8 – 15
8 – 5	Start-up signal rho3.0-PLC .....	8 – 17



Software version TO06



# 1 Workpiece coordinate system

## 1.1 General

The option 'Workpiece coordinate system' offers the user the possibility of defining his own world coordinate system in automatic mode and in manual mode.

Thus, a BAPS program created with workpiece coordinates can be adjusted to the actual workpiece position, e.g., a program that was created offline can be adjusted to the actual position in the respective processing phase.

For a painting line, this means: The body data supplied by a programming system refer to a selected slide type (standard slide). Using the option 'Workpiece coordinate system' it is now possible to adapt the painting programs to all potential slide types.

When the slide type has been determined, the related coordinate system is activated. The actual movement programs are not affected.

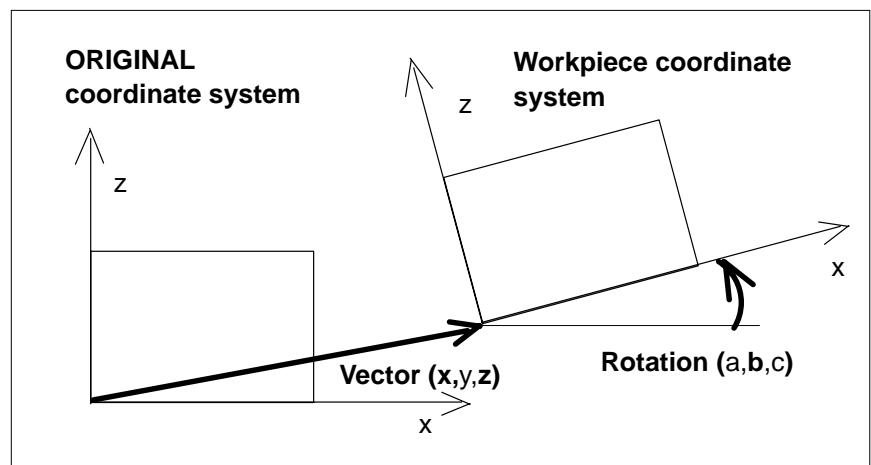


Fig. 6 – 1 Original coordinate system vs. Workpiece coordinate system

The BAPS2 movement program can be created in the original coordinate system or taught in the active workpiece coordinate system (activated by a previously started BAPS2 program using the command WC\_SYSTEM). The movements are purely workpiece-related.

The command WC\_SYSTEM in front of the movement program sets the workpiece zero point in a previously defined relation to the original coordinate system (vector + rotation).



## 1.2 BAPS2 Syntax

The following language elements were introduced in the BAPS programming language for defining the different coordinate systems:

**WC\_FRAME**  
**WC\_SYSTEM**  
**WC\_UR**

**WC\_FRAME** is a kinematic-related standard BAPS command comprising 6 components of the type **REAL**. The 6 components describe the translation and rotation of the workpiece coordinate system within the original coordinate system (robot coordinates). The parameter set indicates the position of the zero point of the workpiece coordinate system in the original coordinate system. The sequence of components is defined as follows:

		Translate original coordinate system into
1st component	= $\Delta x$	x direction
2nd component	= $\Delta y$	y direction
3rd component	= $\Delta z$	z direction
		Rotate original coordinate system by
4th component	= $\Delta a$	x axis
5th component	= $\Delta b$	y axis
6th component	= $\Delta c$	z axis

**This sequence must always be observed!** The translations are entered in [mm], the rotation in [degrees].

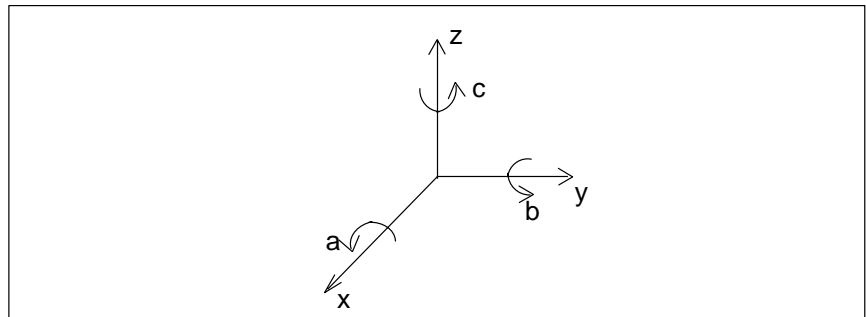


Fig. 6 – 2 Axes and rotation in the workpiece coordinate system

If the workpiece coordinate system is simultaneously rotated by several axes of the original coordinate system, the following definitions must be observed:  $\Delta c$  = rotation by z axis,  $\Delta b$  = rotation by the resulting y' axis,  $\Delta a$  = rotation by the resulting x'' axis.

**WC\_SYSTEM** is a new kinematic-related standard valuable of the type **WC\_FRAME**. This standard variable is used to inform the control about the valid workpiece coordinate system and to activate this system.

**WC\_UR** is a pre-defined BAPS constant of the type **WC\_FRAME**. All of its 6 components have the value zero. If this constant is assigned to **WC\_SYSTEM**, all translations and rotations with respect to the original coordinate system become zero;

e.g. by

```
PAINT_1.WC_SYSTEM = WC_UR
```

The workpiece coordinate system thus becomes identical to the original world coordinate system.

## 1.2.1 Declaration of variables

```
<Kinematics name>.WC_FRAME : <WC name>
```

<WC name> is the user-defined name of the workpiece coordinate system. Its maximum length is 12 characters.

**Note:**

A variable declared in this way may be used **for a single** kinematics only !

Example:

**Declaration of 3 different slide types for 2 kinematics**

```
;;Kinematics : ( 1=PAINT_1 , 2 = PAINT_2 )
:
:
; Slide 1,2,3 for kinematics 1
PAINT_1.WC_FRAME : SLIDE1_K1, SLIDE2_K1, SLIDE3_K1

; Slide 1,2,3 for kinematics 2
PAINT_2.WC_FRAME : SLIDE1_K2, SLIDE2_K2, SLIDE3_K2
```

## 1.2.2 Assignments in BAPS

The assignment of values to the coordinate systems declared above (3 slides for 2 kinematics) in BAPS is done as follows:

```
SLIDE1_K1 = WC_FRAME ( 0, 0, 30, 0, 0, 0 ) ; Slide 1, kin. 1
SLIDE1_K2 = WC_FRAME ( 0, 0, 30, 0, 0, 0 ) ; Slide 1, kin. 2
SLIDE2_K1 = WC_FRAME ( 0, 0, 35, 0, 0, 0 ) ; Slide 2, kin. 1
SLIDE2_K2 = WC_FRAME ( 0, 0, 35, 0, 0, 0 ) ; Slide 2, kin. 2
SLIDE3_K1 = WC_FRAME ( 0, 0, 42, 0, 0, 0 ) ; Slide 3, kin. 1
SLIDE3_K2 = WC_FRAME ( 0, 0, 42, 0, 0, 0 ) ; Slide 3, kin. 2
```

After assignment of the values, the **WC\_FRAME** variables are used for changing over the world coordinate system. In the assignment, the keyword **WC\_FRAME** serves to differentiate between the 6 REAL values and a 6-axis WC point.

## 1.2.3 Changing over the coordinate system

Changing over to a translated workpiece coordinate system is done by assigning a defined variable or constant of the type **WC\_FRAME** to the standard variable **WC\_SYSTEM**.

```
PAINT_1.WC_SYSTEM = SLIDE1_K1
PAINT_2.WC_SYSTEM = SLIDE1_K2      or
PAINT_1.WC_SYSTEM = WC_FRAME ( 0, 0, 30, 0, 0, 0 )
PAINT_2.WC_SYSTEM = WC_FRAME ( 0, 0, 30, 0, 0, 0 )
```

In this example, slide 1 is programmed as a world coordinate system that was translated by +30mm in the direction of z for the two kinematics PAINT\_1 and PAINT\_2. When the world coordinate system has been changed over, the new coordinates remain valid until the WC are changed over or deactivated for the kinematics in question by another command. The only exception is start-up of the control. In this case, the original coordinate system will be active again.

Programmed **deactivation** is achieved with the assignment  
**WC\_SYSTEM = WC\_UR**

In the example, returning to the original coordinate system is programmed by:

```
PAINT_1.WC_SYSTEM = WC_UR
PAINT_2.WC_SYSTEM = WC_UR
```

This is equal to the BAPS instructions:

```
PAINT_1.WC_SYSTEM = WC_FRAME ( 0, 0, 0, 0, 0, 0 )
PAINT_2.WC_SYSTEM = WC_FRAME ( 0, 0, 0, 0, 0, 0 )
```

This programming will cancel all translations and rotations.

## 1.3 Machine parameter P313: WCSYS-ROB-ASSIGNMENT

As described above, 6 components are used for the general definition of the workpiece coordinate system. If the kinematics has less than 6 degrees of freedom, not all translations/rotations of the workpiece coordinate system can be compensated for.

Machine parameter P313 is used to inform the control:

- ★ which workpiece coordinates out of the 6 potential options are accounted for by the mechanics
- ★ which robot coordinate corresponds to which workpiece coordinate.

The resulting **position changes are corrected accordingly**. The resulting orientation change  $\Delta b$  can only be compensated for by machines with the corresponding degrees of freedom.

**Example 1: 3-axis side machine**

e.g. ( YK , ZK , AK , BNK ) = ( 10 , 20 , 30 , 40 )  
          1   2   3    4

**P313 : WCSYS-ROB-ASSIGNMENT**

WCSYS-x 4    the 4th WC PNT component (belt) corresponds to WCSYS-x  
WCSYS-y 1    the 1st WC PNT component (SM-y) corresponds to WCSYS-y  
WCSYS-z 2    the 2nd WC PNT component (SM-z) corresponds to WCSYS-z  
WCSYS-a 0    an orientation change  $\Delta a$  is not compensated for  
WCSYS-b 0    an orientation change  $\Delta b$  is not compensated for  
WCSYS-c 0    an orientation change  $\Delta c$  is not compensated for

**Example 2: 3-axis top machine****P313 : WCSYS-ROB-ASSIGNMENT**

WCSYS-x 4    the 4th WC PNT component (belt) corresponds to WCSYS-x  
WCSYS-y 1    the 1st WC PNT component (TM-y) corresponds to WCSYS-y  
WCSYS-z 2    the 2nd WC PNT component (TM-z) corresponds to WCSYS-z  
WCSYS-a 0    an orientation change  $\Delta a$  is not compensated for  
WCSYS-b 3    the 3rd WC PNT component (TM-b) corresponds to WCSYS-b  
WCSYS-c 0    an orientation change  $\Delta c$  is not compensated for

## 1.4 Selection and effect of a workpiece coordinate system in the movement program

All points described below will be accounted for by the operating system, i.e. the BAPS program does not have to be changed. The operation of the BAPS commands or symbols which are affected by the new option is described below.

### 1.4.1 World coordinate points (WC points)

All world coordinate points used in a BAPS program refer to the local workpiece coordinate system, no matter whether they were programmed offline as texts or taught in online. For as long as no WC\_SYSTEM call is effected in the program after control start-up, the workpiece coordinate system and the robot coordinate system (original coordinate system) are identical, i.e. everything remains as before.

In the movement programs, the command

```
PAINT_1.WC_SYSTEM = SLIDE1_K1
```

or

```
PAINT_2.WC_SYSTEM = WC_FRAME ( PAR_X , PAR_Y , PAR_Z ,  
                                PAR_A , PAR_B , PAR_C )
```

changes from the original world coordinate system to a specific workpiece coordinate system. All programmed world coordinate points are transformed to the current workpiece coordinate system by the control (translation and rotation).

### 1.4.2 Joint coordinate points (JC points)

All points programmed in joint coordinates (JC\_POINT, @ points) **remain unchanged** when a specific workpiece coordinate system is selected. Joint coordinate points describe the axis position of the robot and are therefore independent from the active workpiece coordinate system (e.g. @START\_PNT, @(0,0,0) ).

## 1.4.3 POS , @POS , @MPOS

The values obtained by the assignment

$$\text{CUR\_POS} = \text{POS}$$

depend on the currently active workpiece coordinate system. POS describes the actual position of the robot in local workpiece coordinates.

@POS and @MPOS indicate the axis positions of the robot. They are not dependent upon the active workpiece coordinate system.

## 1.4.4 LIMIT\_MIN, LIMIT\_MAX

The assignments

$$\text{KIN1.LIMIT\_MIN} = \text{MIN\_POSITION}$$
$$\text{KIN1.LIMIT\_MAX} = \text{MAX\_POSITION}$$

can be used to define working range limits. In formal terms, MIN\_POSITION and MAX\_POSITION are normal world coordinate points which refer, just like all WC points, to the local workpiece coordinate system. As a result, LIMIT\_MIN and LIMIT\_MAX must be programmed in workpiece coordinates.

## 1.4.5 Standard functions JC ( ) , WC ( )

The JC standard function provides a conversion from world coordinates into joint coordinates, e.g.

$$\text{@BEG\_POS} = \text{JC}(\text{BEG\_POS})$$

The result is the arm position of the kinematics. Since the starting point is a world coordinate point, the result depends on the active workpiece coordinate system.

The WC standard function provides a conversion from joint coordinates into world coordinates, e.g.

$$\text{BEG\_POS} = \text{WC}(\text{@BEG\_POS})$$

The result are the coordinates related to the local workpiece coordinate system, i.e. the result depends on the active workpiece coordinate system.

## 1.4.6 Special functions 1, 2: IO/PPO Logic

Using special functions 1, 2 position–dependent process parameter outputs can be programmed. The programmed activation/deactivation coordinates refer to the local workpiece coordinate system, for example, switching is always performed in the same body position. With respect to the original coordinate system (robot coordinate system), the switching positions change with different workpiece coordinate systems.

## 1.4.7 Special function 17: MIRROR

With this option, it is possible to mirror any points across the axes of the world coordinate system. As for all other programmed world coordinate points, mirroring refers to the local workpiece coordinate system.

## 1.4.8 Translating the world coordinate system (P310)

Machine parameter P310 is used to define the zero point of the robot coordinate system, i.e. the original coordinate system. The coordinates of the active workpiece coordinate system (i.e. the coordinates transferred by the command WC\_SYSTEM) always refer to the original coordinate system.

## 1.4.9 Limit switch monitoring

In Automatic mode, the software limit switches are monitored in machine coordinates (machine parameters P204, P205). Monitoring is therefore not dependent on the currently active workpiece coordinate system.

## 1.4.10 Program end, abort, control start–up

The coordinate system can only be changed by making the corresponding selection in a BAPS program. After program end, the last active workpiece coordinate system remains selected.

In the event of program abort (e.g. EMERGENCY–STOP, Automatic–Manual changeover, Reset, etc.) the workpiece coordinate system that was active when the program was interrupted remains selected.

After control start–up, the original coordinate system is active, i.e.

WC\_SYSTEM = WC\_UR.

## 1.4.11 Axis displays

The axis displays on the PHG (mode 7.1) are as follows:

The joint coordinates (**JC**) always show the current axis position which is not dependent on the active workpiece coordinate system.

The world coordinates (**WC**) show the values in local workpiece coordinates. The display depends on the active workpiece coordinate system.

A new level, original coordinates (**OC**) is created. The OC show the positions in the original coordinate system. The display is not dependent upon the active workpiece coordinate system. This level is offered for display only if  $WC\_SYSTEM \neq WC\_UR$ , i.e. if a local workpiece coordinate system has been activated.

## 1.4.12 Displaying the active WC\_SYSTEM

In mode 7.1 of the PHG, a new level is created with **<Shift>< →>** which shows the coordinates of the currently active workpiece coordinate system. If all coordinates equal zero, no workpiece coordinate system is active, i.e.  $WC\_SYSTEM=WC\_UR$ .

## 1.5 Selection and effect in Manual mode

As mentioned above, a workpiece coordinate system can only be activated by making the appropriate selection in a BAPS program. Since the last active workpiece coordinate system remains active after program end, a concrete workpiece coordinate system can also be applied in Manual mode, for example by running the program

```
;;INCLUDE HEAD_PAINT
PROGRAM SLIDE_1
;Translate slide 1 in comparison to standard slide
PAINTL.WC_SYSTEM = WC_FRAME ( 0 , 0 , 30 , 0 , 0 , 0 )

END
```

By starting the program SLIDE\_1, the zero point of slide 1 is defined in the original coordinate system. In manual mode, it is now possible to re-teach in points for an offline program.

The option *'Direct approach to teach-in points'* also uses the active workpiece coordinate system.



## 1.6 Safety instruction



### CAUTION

#### 6.4

**At the time of program start, it is mandatory to ensure that the proper coordinate system corresponding to the workpiece being used is enabled. In the event that programs are started with incorrect workpiece coordinate systems, unexpected and undesirable movements may result.**

**The same effect may occur if different workpiece coordinate systems are used for teach-in and for program execution for a taught-in world coordinate point.**

The option '*Workpiece coordinate system*' acts globally for each kinematics, i.e. across program and process limits. If individual subroutines (e.g. BEG\_POS, END\_POS) are called up in several parts of the entire process, it must be ensured through proper programming that they always use the same workpiece coordinate system.

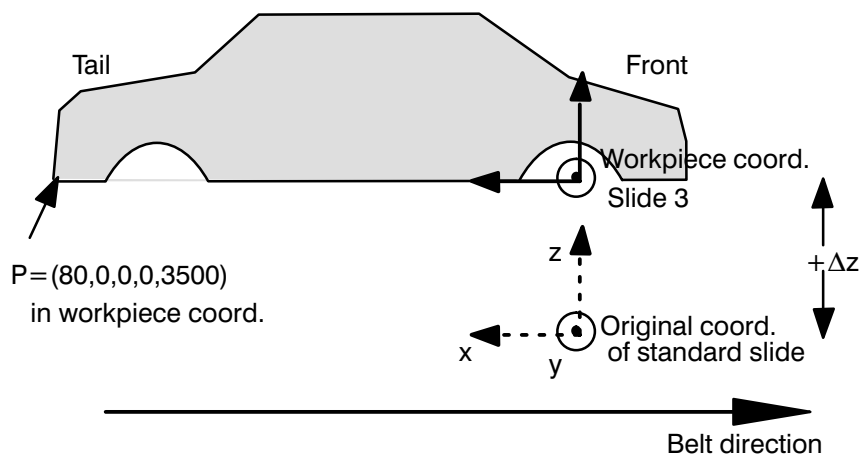
## 1.7 Examples of special workpiece coordinates

The command `WC_SYSTEM` is used to inform the control of the zero point offset of the local workpiece coordinate system with respect to the zero point of the original coordinate system (robot coordinate system), i.e. in this concrete case the offset of slides 1..3 from the standard slide. Slides 1..3 correspond to the local workpiece coordinate system, and the standard slide corresponds to the original coordinate system.

### 1.7.1 Example 1: Pure height offset

Example 1 shows the case of a mere height offset of  $+\Delta z$ , i.e. on slide 3 the body is higher than on the standard slide. The workpiece coordinate system of slide 3 would have to be defined as follows:

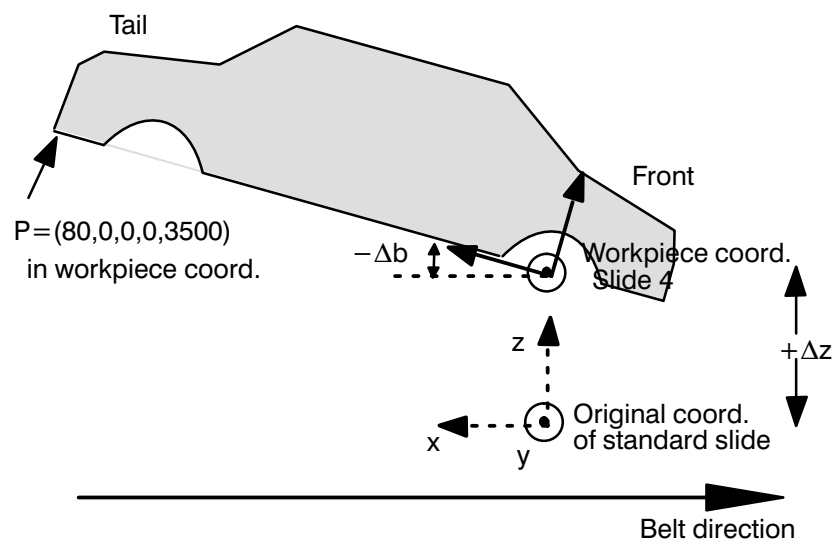
$$\text{SLIDE3} = \text{WC\_FRAME} ( 0 , 0 , +\Delta z , 0 , 0 , 0 )$$



## 1.7.2 Example 2: Height offset and rotation

In this example, the body is higher than with the standard slide and rotated by  $-\Delta b$  by the y axis. The workpiece coordinate system of slide 2 would have to be defined as follows:

**SLIDE2 = WC\_FRAME ( 0 , 0 , + $\Delta z$  , 0 , - $\Delta b$  , 0 )**



## **2 rho 3.0**

The additional functions of the version TO06 basically refer to the support of the rho3.0. In detail, the following functions are available:

### **2.1 Software PLC for the rho3.0**

For the rho3.0 the PIC functions are available in the form of a software PLC. The scope of functions is fully compatible with the PIC 250 of the rho3.1. However, the software PLC requires additional processing time.

The PIC signal assignment is not included in the manual '*Signal description and error messages*' No. 1070 073029.

### **2.2 Runtime display of the software PLC**

The runtime of the software PLC can be displayed on the PHG by selecting MODE 3.2.5. The display shows the minimum, maximum, current and average runtime.

Also refer to '*rho3.0 Required Connections and Project Development Information*' No. 1070 072149, Chapter '*Configuration*'.

### **2.3 Communication between rho3.0 – PLC**

If the rho3.0 is used together with a PLC central unit in one PLC rack, data exchange between the rho3.0 – Central Unit is effected via the I/O bus using the KOMFIFO communication module and the DBLOAD module.

Also refer to '*rho3.0 Required Connections and Project Development Information*' No. 1070 072 149, Chapter '*Interfacing with the PLC*'.

### **2.4 Using the high-speed inputs and outputs of the rho3.0**

The rho3.0 has (optionally) 4 high-speed inputs and 4 high-speed outputs at connector X21. A probe input is additionally available.

Also refer to '*rho3.0 Required Connections and Project Development Information*' No. 1070 072 149, Chapter '*High-speed inputs/outputs*'.

## 2.5 Double assignment of the PHG interface

In the rho3.0, the PHG interface can be operated as a **PHG interface** or as an active **V24/20mA interface**.

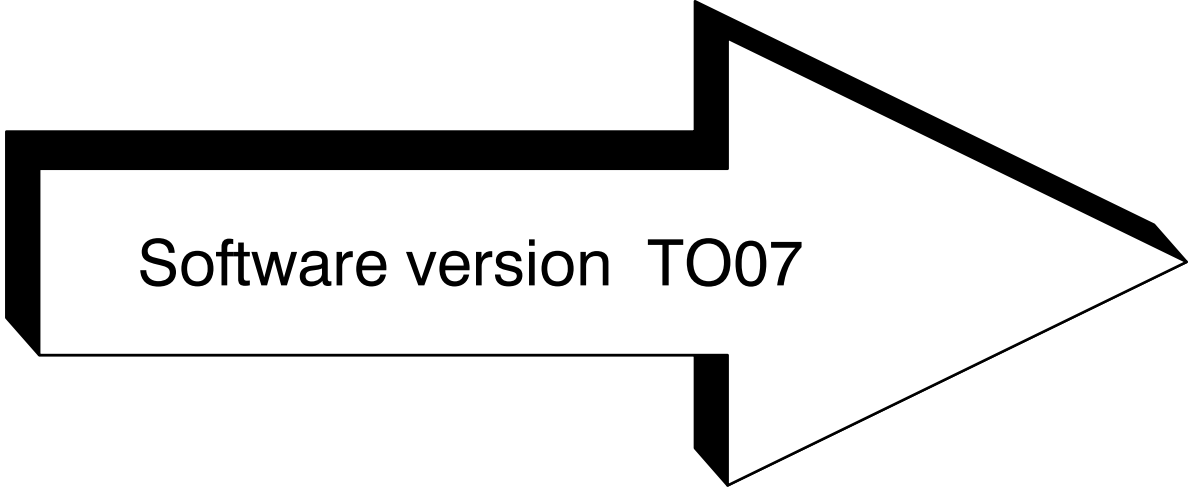
If a PHG is connected, it is automatically recognized (jumper in PHG connector), and the interface is operated as PHG interface.

If the PHG is not connected but rather another device, the interface will be operated as assigned.

## 2.6 New or expanded machine parameters

P11:	expanded	Numer of high-speed outputs (max. 4)
P15:	expanded	Servo board type 9 = rho 3.0
P33:	new	Hardware version of the rho 3.0 1 SoftPIC version (stand-alone) 0 CL version (module in PLC rack)
P34:	new	Number of characters which are maximally output if the handshake signal is set.

Also refer to '*rho3.0 Required Connections and Project Development Information*' No. 1070 072 149, Chapter '*Configuration*' and/or '*Description of machine parameters*' No. 1070 073 027.



Software version TO07



# 1 PHG 2000

The new features of the software version TO07x basically refer to the support of the PHG 2000. Furthermore, the IAT module is now supported as higher-level control, and coupling of two rho's via the CAN bus is supported.

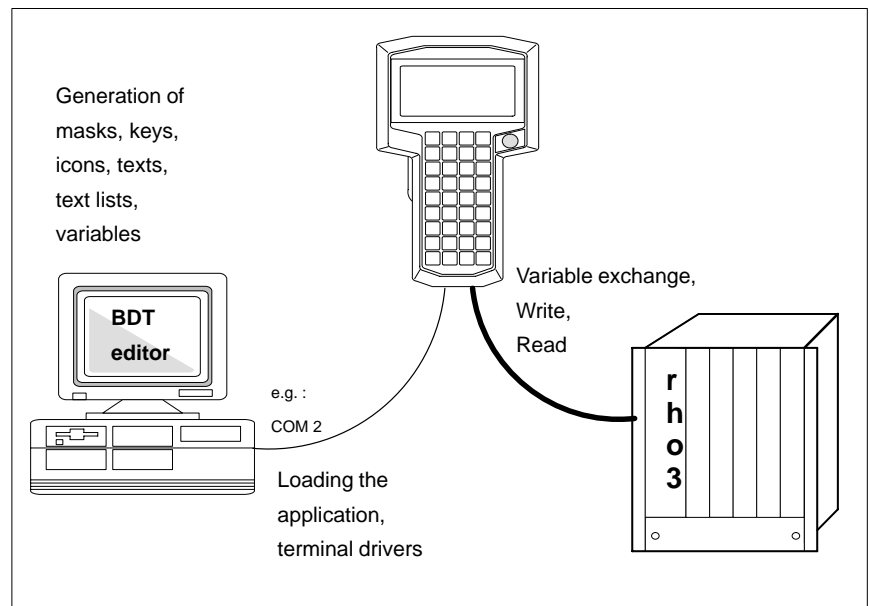


Fig. 7 – 3 PHG2000 Interfacing options

The PHG 2000 has a 256 x 128 pixel full-graphics display. Currently, two character sets (6\*8 and 9\*11 pixels) are available.

The user can create so-called masks for building up a menu structure with subbranches in order to obtain a user-specific interface that suits his requirements.

Icons can be included in these masks for operator guidance which provide for the selection of the individual menu levels with the assignment:

**icon** – **key (K1 – K36)** – **mask**

Texts, text lists and variables may be edited in the masks. Controlled by the PHG 2000, variables from the control are displayed for the text descriptions. Alternatively, variables are copied to the control by a simple key operation. Furthermore, a rho response may be initiated by the feedback value of the variable text list (depending on the cursor position).

The assignment of the PHG keys can be adapted to customer-specific requirements by loading a different key assignment (KEY.BNR in the rho or LAYOUT.LAY in the PHG 2000).



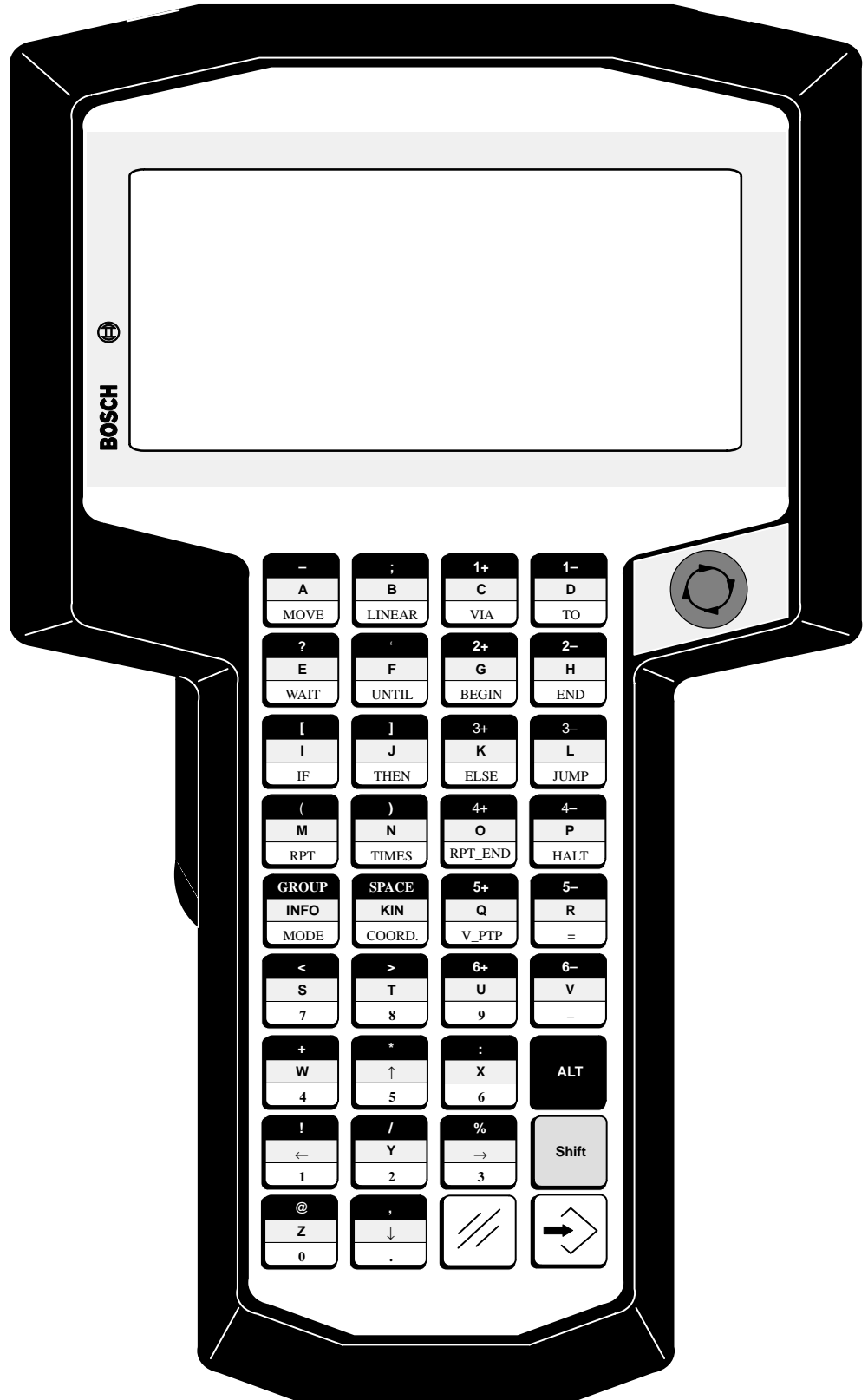


Fig. 7 - 4 General view of the PHG 2000

## 1.1 PHG 2000 modes

### 1.1.1 PHG 3 compatible mode

Machine parameter **P35 (PHG-Mode)** is set to **1**.  
The rho controls the data transmission (4800 baud).

- ★ Menu structure and operation with mode key identical to PHG 3
- ★ Display as in PHG 3 – 4 lines with 20 characters each (characters are bigger)
- ★ Write PHG / Read PHG as with PHG 3
- ★ Define / Teach In as with PHG 3
- ★ Redefine keys using the **KEY.BNR** file

### 1.1.2 Transparent mode

Machine parameter **P35 (PHG-Mode)** is set to **0** (default setting upon delivery). Data transmission with 9600 baud.

- ★ Define (mode 4.1) with new display and operation
- ★ Teach in (mode 4.2) with new display and operation
- ★ Redefine keys using the **KEY.BNR** file
- ★ Write PHG / Read PHG as with PHG 3 (through PHG interface or V24x with protocol 6), Display: 4 lines with 20 characters each (characters are bigger)
- ★ Write V24x / Read V24x (through V24x interface, setting of V24x: interface 2 with protocol 2 (with echo) or protocol 5 (without echo)). For new functions for 256 x 128 pixels (in 6\*8 or 9\*11 character set) please refer to chapter Controlling the PHG 2000 output

#### 1.1.2.1 Transparent mode with existing menu structure

The **PHG keys** and the **PHG display** are **not disabled** in the PLC.  
The rho controls the data transmission (9600 baud).

- ★ Menu structure and operation with mode key identical to PHG 3
- ★ Display as in PHG 3 – 4 lines with 20 characters each (characters are bigger)

### 1.1.2.2 Transparent mode with user-defined menu structure

The **PHG display** (and maybe also the **PHG keys**) are **disabled** in the PLC. The PHG 2000 controls data transmission, the rho only supplies the data.

- ★ User masks were generated using the BDT editor and have been loaded  
 —> **individual menu structure, user-defined key functions**

## 1.2 Key assignment

The standard key assignment will remain valid unless a new rho start-up has been performed using the KEY.BNR file, thus activating a new key assignment, or the PHG keys and the PHG display were disabled in transparent mode with a user-defined menu structure, thus activating the mask-related key assignment (as defined in LAYOUT.LAY).

### 1.2.1 Mask-specific key assignment

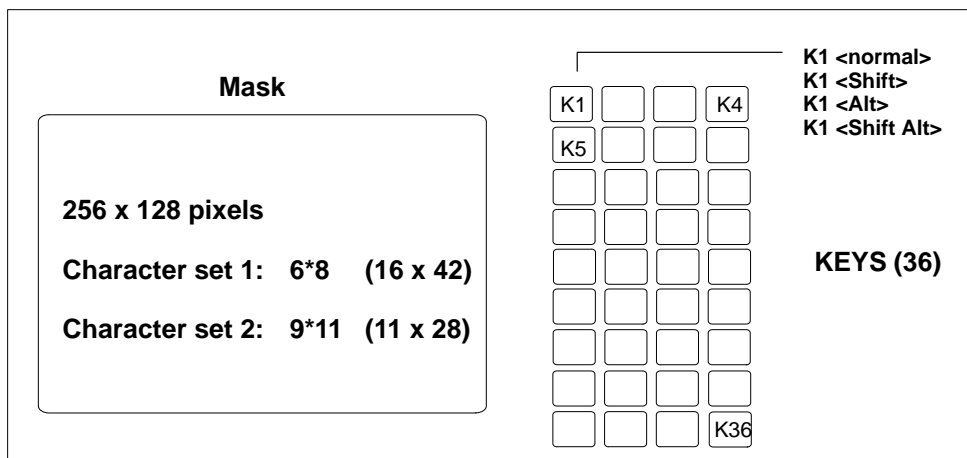


Fig. 7 – 5 PHG display and key assignment

A user-specific, freely definable key assignment is permanently assigned to each mask. Each of the 36 keys K1 – K36 (cf. Fig. 7 – 5) can have 4 assignments (normal, SHIFT+key, ALT+key, SHIFT+ALT+key).

Each key combination can be linked to the selection of a new mask or the description of variables to build up the user-defined menu structure.

A maximum of 128 KBytes are available in the FLASH-EPROM memory for user-defined masks and variables.

### 1.3 BDT editor

The BDT editor tool is used to define the PHG 2000 masks and the links between masks via keys.

The following mask elements are available:

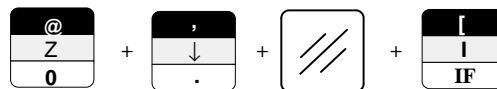
- ★ constant **texts**
- ★ display **variables** (cf. list of system variables)  
e.g. AUTO\_MN\_RCI, position of axis 3
- ★ display constant texts **with variable portions**  
e.g. "Number: **47** pieces"
- ★ display **variables as texts**  
e.g. OFF, ON, BLACK, BROWN, RED
- ★ display / scroll through **lists with constant texts**  
selection can be written to rho variables (feedback value)
- ★ display / scroll through **lists with variable texts**  
selection can be written to rho variables (feedback value)
- ★ write a **variable when calling up the mask**
- ★ write a **rho variable by depressing / releasing a key**  
e.g. F1 gripper=1 F5 gripper = 0
- ★ **edit rho variables on display**
- ★ **call up other masks by pressing a key**
- ★ call up other masks by pressing a key **depending on variables**  
(e.g. a key-operated switch)

## 1.4 Configuration and test

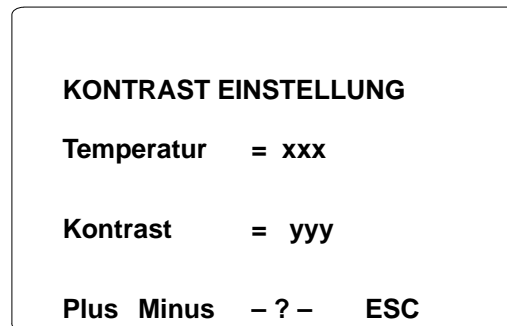
Only possible if no communication with the rho is active.

—> Remove the PHG connector from the rho so that only the 24V supply is still connected.

Press the key combination:



to select the contrast setting of the PHG 2000.



- ★ The contrast value can be set in the range 0 .. 127 using the key assignments **Plus** (K1) and **Minus** (K2).
- ★ The version statuses of all PHG drivers are displayed with the **?** key (K3).
- ★ The key combination **Shift+?** calls up the test mode where the keys of the PHG 2000 can be checked. When the key is depressed, the display shows the following:

- |   |                   |
|---|-------------------|
| 1 | Key alone         |
| 2 | SHIFT + key       |
| 3 | ALT + key         |
| 4 | SHIFT + ALT + key |

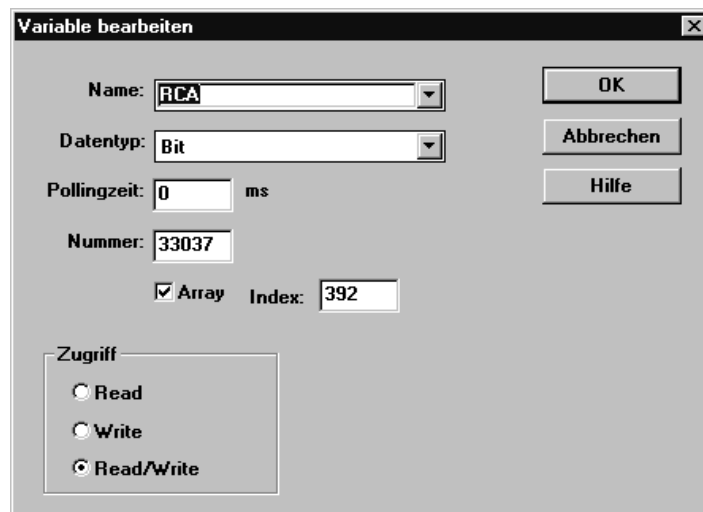
## 2 PHG 2000 System Variables

Data transmission to and from the rho and the PHG 2000 is handled by system variables. These system variables are declared in the BDT editor. They have read access, and in some cases also write access, to the rho variables (not to the user variables declared in the program).

The following definitions must be made for declaring the system variables:

- ★ Name (variable name): max. 30 characters
- ★ Datentyp (data type): Bit,  
Byte, Byte (unsigned)  
Word, Word (unsigned)  
Double, Double (unsigned)  
String, String (variable text list)  
Real (IEEE format)
- ★ Nummer (variable number): cf. tables below
- ★ Array: If the variable number does not clearly point to a single variable but rather to a variable array, Array must be selected. As a result, an entry in the Index field is also necessary.
- ★ Index (variable index): Serial number (only for Array, cf. tables) coded index (for multi-dimensional array)

(For the PIC signals, the index is derived from the standard signal assignment of the PIC interface in use. Cf. manual *Signal description and error messages*, Part No. 1070 073 029).



Description		Base type	Index	Code	R/W
<b>SIGNALS</b>				<b>33024</b>	
PIC inputs	bit	Bit	0 ... 1023	33025	R
PIC outputs (RCI)	bit	Bit	0 ... 1023	33026	R
PIC flags	bit	Bit	0 ... 1023	33027	R
PIC inputs	byte	Byte	0 ... 127	33028	R
PIC outputs (RCI)	byte	Byte	0 ... 127	33029	R
PIC flags	byte	Byte	0 ... 127	33030	R
PIC inputs	word	Word	0 ... 126	33031	R
PIC outputs (RCI)	word	Word	0 ... 126	33032	R
PIC flags	word	Word	0 ... 126	33033	R
PIC inputs	double	Double	0 ... 124	33034	R
PIC outputs	double	Double	0 ... 124	33035	R
PIC flags	double	Double	0 ... 124	33036	R
RC outputs	bit	Bit	64 ... 511	33037	R/W
RC outputs	byte	Byte	8 ... 63	33038	R/W
RC outputs	word	Word	8 ... 62	33039	R/W
RC outputs	double	Double	8 ... 60	33040	R/W
MTB PIC inputs	bit	Bit	640 ... 703	33041	R/W
MTB PIC inputs	byte	Byte	80 ... 87	33042	R/W
MTB PIC inputs	word	Word	80 ... 86	33043	R/W
MTB PIC inputs	double	Double	80 ... 84	33044	R/W
SB1 inputs	bit	Bit	0 ... 15	33045	R
SB2 inputs	bit	Bit	0 ... 15	33046	R
SB3 inputs	bit	Bit	0 ... 15	33047	R
SB1 outputs	bit	Bit	0 ... 15	33048	R
SB1 inputs	byte	Byte	0 ... 1	33049	R
SB2 inputs	byte	Byte	0 ... 1	33050	R
SB3 inputs	byte	Byte	0 ... 1	33051	R
SB1 outputs	byte	Byte	0 ... 1	33052	R
SB1 inputs	word	Word	0	33053	R
SB2 inputs	word	Word	0	33054	R
SB3 inputs	word	Word	0	33055	R
SB1 outputs	word	Word	0	33056	R
Number of analog inputs		Byte	–	33057	R
Number of analog outputs		Byte	–	33058	R

Description	Base type	Index	Code	R/W
<b>SIGNALS</b>			<b>33024</b>	
Analog inputs	Real	1 ... 32	33059	R
Analog outputs	Real	1 ... 32	33060	R

Description	Base type	Index	Code	R/W
<b>POSITIONS</b>			<b>33280</b>	
JC position Set	Real	1 ... 32	33281	R
JC position Actual	Real	1 ... 32	33282	R
JC position Set	Real	1 ... 32	33283	R
Spare			33284	
OC position	Real	1 ... 32	33285	R
GC position	Real	1 ... 32	33286	R
SW limit switch positive WC	Real	1 ... 32	33287	R
SW limit switch negative WC	Real	1 ... 32	33288	R
SW limit switch positive JC	Real	1 ... 32	33289	R
SW limit switch negative JC	Real	1 ... 32	33290	R
Axis names	String	1 ... 32	33291	R
Coordinate names	String	1 ... 32	33292	R
Lag	Real	1 ... 32	33293	R
Inpos ( '*' or ' ' )	String	1 ... 32	33294	R
Offset	Real	1 ... 32	33295	R
End point	Real	1 ... 32	33296	R
Number of axes	Byte	–	33297	R
Number of belts	Byte	–	33298	R
Belt	Real	1 ... 8	33299	R
Belt names	String	1 ... 8	33300	R

All position variables can be addressed with:

simple index : current axis number (total system) or  
 coded index : kinematics number / axis number of the kinematics

For a coded index, the value to be entered comprises the identification for a coded index in the uppermost bit (15), the kinematics number and the number of the component (axis).

1	Kinematics 1 – 127	Component 1 – 255
---	--------------------	-------------------

Calculation: **32768 + Kinematics no. \* 256 + Component no.**



Description	Base type	Index	Code	R/W
<b>OTHER KIN-INFO</b>			<b>33536</b>	
No. of axes kinematics-dependent	Byte	1 ... 16	33537	R
No. of belts kinematics-dependent	Byte	1 ... 16	33538	R
Tool name	String	1 ... 16	33539	R
Tool <sup>1</sup>	Real	coded	33540	R
WC system <sup>2</sup>	Real	coded	33541	R

<sup>1</sup> only with coded index: Kinematics 1...16, component 1...6

<sup>2</sup> only with coded index: Kinematics 1...16, component 1...6

Description	Base type	Index	Code	R/W
<b>A, V, D</b>			<b>33792</b>	
VFACTOR global	Real	–	33793	R/W
AFACTOR global	Real	–	33794	R/W
DFACTOR global	Real	–	33795	R/W
VFACTOR globale limit min	Real	–	33796	R
VFACTOR globale limit max	Real	–	33797	R
AFACTOR globale limit min	Real	–	33798	R
AFACTOR globale limit max	Real	–	33799	R
DFACTOR globale limit min	Real	–	33800	R
DFACTOR globale limit max	Real	–	33801	R
VFACTOR kin	Real	1 ... 16	33802	R/W
AFACTOR kin	Real	1 ... 16	33803	R/W
DFACTOR kin	Real	1 ... 16	33804	R/W
VFACTOR kin limit min	Real	1 ... 16	33805	R
VFACTOR kin limit max	Real	1 ... 16	33806	R
AFACTOR kin limit min	Real	1 ... 16	33807	R
AFACTOR kin limit max	Real	1 ... 16	33808	R
DFACTOR kin limit min	Real	1 ... 16	33809	R
DFACTOR kin limit max	Real	1 ... 16	33810	R

Description	Base type	Index	Code	R/W
<b>SYSTEM-INFO</b>			<b>34048</b>	
Clock time [ms]	Word	–	34049	R
Time	String	–	34050	R/W
Date	String	–	34051	R/W
Date/time	String	–	34052	R/W
Version P1	String	–	34053	R
Version P2	String	–	34054	R
Version SB1	String	–	34055	R
Version SB2	String	–	34056	R
Version SB3	String	–	34057	R
Version 8085	String	–	34058	R
PIC program name	String	–	34059	R
Number of warnings	Word	–	34060	R
Numer of errors	Word	–	34061	R
Warning ASCII axes + code	String	1 ... 32	34062	R
Warning ASCII + axes	String	1 ... 32	34063	R
Warning ASCII	String	1 ... 32	34064	R
Warning axis-coded	Word	1 ... 32	34065	R
Warning code	Word	1 ... 32	34066	R
Error ASCII axes + code	String	1 ... 32	34067	R
Error ASCII + axes	String	1 ... 32	34068	R
Error ASCII	String	1 ... 32	34069	R
Error axis-coded	Word	1 ... 32	34070	R
Error code	Word	1 ... 32	34071	R

Description	Base type	Index	Code	R/W
<b>PROCESS-INFO</b>			<b>34304</b>	
Number of normal processes	Word	–	34305	R
Number of permanent processes	Word	–	34306	R
Number of sub-processes	Word	–	34307	R
Number of incorrect processes	Word	–	34308	R
Process list (listbox)	String	1 ... 100	34309	R
Selected process	Word	–	34310	R/W
Process type	String	–	34311	R
Process priority	Word	–	34312	R
Process status	String	–	34313	R
External program	String	–	34314	R
Program level	Byte	–	34315	R
Process name	String	–	34316	R
QLL line	Word	–	34317	R
QLL INCLUDE line	Word	–	34318	R
Kinematics	Word	–	34319	R
Process error code	Word	–	34320	R
Process error text	String	–	34321	R
Stop process	Byte	–	34322	W

Description	Base type	Index	Code	R/W
<b>FILE-INFO</b>			<b>34560</b>	
List file – name/size/date	String	1 ... 1000	34561	R
List file – name/size	String	1 ... 1000	34562	R
List file – name/date	String	1 ... 1000	34563	R
List file – name	String	1 ... 1000	34564	R
Cursor Index	Double	–	34565	R
Control variable	Double	–	34566	R
Memory allocation	Double	–	34567	R
Memory available	Double	–	34568	R
Memory used	Double	–	34569	R
Free memory	Double	–	34570	R

Description	Base type	Index	Code	R/W
<b>VARIABLE TEXT LISTS</b>			<b>34816</b>	
VTL000	String	1 ... 999	34816	R
VTL001	String	1 ... 999	34817	R
VTL002	String	1 ... 999	34818	R
. . . . . 256 DAT files . . . . . . . . . .	one file line	index corresponds to line number		
VTL253	String	1 ... 999	35068	R
VTL254	String	1 ... 999	35069	R
VTL255	String	1 ... 999	35070	R

Your notes:

## 3 Controlling the PHG 2000 output

### 3.1 General information

- ★ The subroutine calls described in chapter 3.3, "Calling subroutines in the BAPS2 main program section" do not constitute complete or useful programs, but rather a description of the function, transfer parameters and call of the subroutines.
- ★ The interface (e.g. Ser\_4) to which the PHG 2000 is connected must be set (e.g. with MODE 9.1.1.x) to interface 2 (V24\_2) protocol 2 (data with echo) or 5 (data without echo). The connector remains at X32 (rho3.0) or X22 (rho3.1).
- ★ The PHGAUSG file in the rho directory of the mailbox (phone xx49-6062-7217) can be used as model program.
- ★ Mask positions are usually specified in pixels.  
The display size is 256x128 pixels.  
The coordinate origin (x=0, y=0) is the upper left corner.
- ★ The character attributes and character set have to be selected only once if it is ensured that they have not been changed by other processes or system outputs.

#### 3.1.1 Examples of cursor positioning for different character sets

##### **1st character set                      6\*8 (6 pixels wide, 8 pixels high)**

The 7th character in the 3rd line has the pixel position **x=36, y=16**

The following formula should be applied:

$$\begin{array}{rcl} \text{x-pixel position} & = & (\text{column}-1) * \text{horizontal character size} \\ 36 & = & (7-1) * 6 \end{array}$$

$$\begin{array}{rcl} \text{y-pixel position} & = & (\text{line}-1) * \text{vertical character size} \\ 16 & = & (3-1) * 8 \end{array}$$

##### **2nd character set 9\*11                      (9 pixels wide, 11 pixels high)**

The 5th character in the 7th line has the pixel position **x=36, y=66**

The following formula should be applied:

$$\begin{array}{rcl} \text{x-pixel position} & = & (\text{column}-1) * \text{horizontal character size} \\ 36 & = & (5-1) * 9 \end{array}$$

$$\begin{array}{rcl} \text{y-pixel position} & = & (\text{line}-1) * \text{vertical character size} \\ 66 & = & (7-1) * 11 \end{array}$$

## 3.2 BAPS2 declaration section

Program TRANSPAR

----- **Beginning of declaration section** -----

**; Commands of transparent mode**

; Low-Byte = command; High-Byte=command length

```
CONST :                ; low  high
  CD_CLS                = 1 + (0 * 256),
  CD_GOTOXY             = 2 + (4 * 256),
  CD_ATTRIB            = 3 + (1 * 256),
  CD_TEXT              = 5,                ; fixed length of 80 characters
  CD_REFRESH           = 7 + (4 * 256),
  CD_ENDE              = 8 + (0 * 256),
  CD_LINIE             = 12 + (8 * 256),
  CD_N_MASKE           = 15 + (2 * 256),
  CD_CHARSET           = 18 + (1 * 256),
  CD_ZEI_REL           = 22 + (2 * 256)
```

----- Display attributes -----

```
CONST :
  NORMAL                = 0,                ; characters are displayed normally
  BLINK                 = 1,                ; characters flash
  INVERS                = 2,                ; characters are displayed in inverse video
  DOPPEL_BREIT         = 4,                ; characters are displayed in double width
  DOPPEL_HOCH          = 8,                ; characters are displayed in double height
```

----- Character sets -----

```
CONST :
  CS_6_MAL_8            = 0,                ; max. 42x16 characters on display
  CS_9_MAL_11           = 1,                ; max. 28x11 characters on display
```

----- Mask selection -----

```
CONST :
  BEL_MASKE             = -1,               ; -1= always show PHG outputs
  START_MASKE           = 1,               ; Start mask
  MK_IST_POS            = 2,               ; Mask for actual position display of axes in JC
```

----- System variables -----

```
CONST :
  VTL000_NR             = 34816            ; Basic number of the VTL files
```

-----  
FILE : PHG2000

----- **End of declaration section** -----

## 3.3 Calling subroutines in the BAPS2 main program section

```

;----- Beginning of main program section -----
BEGIN

ASSIGN PHG2000, 'V24_4. '           ; The file variable "PHG 2000" is assigned the interface "V24_4".
                                   ; Three blanks are used as extension.

SC_FUER_MASK ( BEL_MASKE )         ; Each PHG 2000 output must begin with the mask number
                                   ; for which the output is intended. If another mask number is just
                                   ; being shown, the output is discarded. If "-1" is transferred as
                                   ; value, the output will always be made, regardless of the active
                                   ; mask. The output must be concluded by "SC_ENDE".

SC_NEUE_MASK ( START_MASKE )      ; Select start mask
SC_NEUE_MASK ( MK_IST_POS )       ; Select mask for actual position display of the axes in JC
SC_CLS                             ; The PHG display is completely cleared.
SC_LINIE ( 0, 0, 255, 127 )       ; A line is drawn from the upper left corner (x=0, y=0)
                                   ; to the bottom right corner (x=255, y=127).

SC_CHARSET ( CS_9_MAL_11 )        ; Select the character size (in this case character matrix
                                   ; of 9x11 pixels).

SC_ATTRIB ( NORMAL )              ; Set attribute for normal display.
SC_ATTRIB ( BLINK+INVERS )        ; Set attribute for flashing and inverse display (constants may
                                   ; be added).

SC_GOTOXY ( 30, 40 )              ; Position cursor to x=30, y=40 pixels.
SC_TEXT ( 'HALLO WORLD' )         ; Display character sequence "HALLO WORLD" on the PHG.
SC_GOTOXY ( 30, 60 )              ; Position cursor to x=30, y=60 pixels.
SC_DEZ ( -1234.45 )               ; The decimal number "-1234.45" is displayed on the PHG.
SC_GOTOXY ( 30, 80 )              ; Position cursor to x=30, y=80 pixels.
SC_GANZ ( 1234, 8 )               ; Display integer "1234". The second transfer parameter ("8")
                                   ; indicates the max. length. The output is right-aligned; empty
                                   ; places are filled up with blanks.

SC_REFRESH ( VTL000_NR+5,3 )      ; Refresh VTL005.DAT with line cursor in line 3.
SC_REFRESH ( 34068,1 )            ; Refresh error display with line cursor in line 1.
SC_ZEI_REL ( -1,0 )               ; Moves the write position to the left by one character width.
SC_ZEICHEN ( ':' )                ; Display character ":" on the PHG.
SC_ENDE                            ; This command initiates the output to the PHG 2000. All
                                   ; outputs between "SC_FUER_MASK" and "SC_ENDE" are
                                   ; transmitted to the PHG.

PROGRAM_END
;----- End of main program -----

```



## 3.4 Outputs to the PHG 2000 in subroutine technique

The subroutines listed below are available in the **PHGAUSG.EXE** file in the **rho** directory of the mailbox, phone number xx49–6062–7217.

\*\*\*\*\* **Decoding and output of the high and low control byte** \*\*\*\*\*

- ; The transfer parameter "PARAM" contains the value of the command to be transmitted to the PHG 2000.
- ; The sequence of characters ".." has the effect that the output is stored in an intermediate memory and is not transmitted to the PHG 2000 before the "SC\_ENDE" command.
- ; This subroutine is needed by other SR's for transmitting the command to the PHG 2000..

```
SUBROUTINE SCHR_GNZ_WRD ( VALUE INT : PARAM )
BEGIN
  PARAM = PARAM MOD 65536
  WRITE PHG2000, CHR ( PARAM MOD 256 ), CHR ( PARAM / 256 ), ..
SUB_END
```

\*\*\*\*\*

\*\*\*\*\* **Definition of the mask for the output** \*\*\*\*\*

- ; The variable "MASK\_NO" contains the number of the mask for which the output is designed.

```
SUBROUTINE SC_FUER_MASK ( VALUE INT : MASK_NO )
BEGIN
  SCHR_GNZ_WRD ( MASK_NO )
SUB_END
```

\*\*\*\*\*

\*\*\*\*\* **Select mask for PHG display** \*\*\*\*\*

- ; The variable "MASK\_NO" contains the number of the mask to be displayed on the PHG 2000.

```
SUBROUTINE SC_NEUE_MASK ( VALUE INT : MASK_NO )
BEGIN
  SCHR_GNZ_WRD ( CD_N_MASKE )
  SCHR_GNZ_WRD ( MASK_NO )
SUB_END
```

\*\*\*\*\*

.\*\*\*\*\* Output of the command to the PHG 2000\*\*\*\*\*  
,

; The WRITE command without ".." starts transmission to the PHG 2000.

```
SUBROUTINE SC_ENDE
BEGIN
  WRITE PHG2000, CHR(CD_ENDE), CHR(0)
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\*Select display design\*\*\*\*\*  
,

; The transfer parameter "ATTRIBUT" contains the value for the display attribute.  
; The available attributes and their values are described in the declaration of the constant.  
; For activating several attributes simultaneously, the constants may be summed  
; (cf. example in main program).

```
SUBROUTINE SC_ATTRIB ( VALUE INT : ATTRIBUT )
BEGIN
  SCHR_GNZ_WRD ( CD_ATTRIB )
  WRITE PHG2000, CHR(ATTRIBUT), ..
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\*Select point matrix\*\*\*\*\*  
,

; The variable "CHAR\_SET" contains the value for the point matrix of the characters to be displayed.  
; The available character sets and their values are described in the constant declaration.

```
SUBROUTINE SC_CHARSET ( VALUE INT : CHAR_SET )
BEGIN
  SCHR_GNZ_WRD ( CD_CHARSET )
  WRITE PHG2000, CHR(CHAR_SET), ..
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\* Define write position on the display \*\*\*\*\*  
,

; The transfer parameter "X\_PIXEL" contains the value for the horizontal pixel position for positioning  
; the cursor on the display. "Y\_PIXEL" contains the value for the vertical position.

```
SUBROUTINE SC_GOTOXY ( VALUE INT : X_PIXEL, Y_PIXEL )
BEGIN
  SCHR_GNZ_WRD ( CD_GOTOXY )
  SCHR_GNZ_WRD ( X_PIXEL )
  SCHR_GNZ_WRD ( Y_PIXEL )
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\* **Line draw** \*\*\*\*\*  
,

; The transfer parameters "FROM\_X" and "FROM\_Y" contain the values (in pixels) for the starting position,  
; the parameters "TO\_X" and "TO\_Y" the values for the end position (in pixels) of the line.

```
SOUBROUTINE SC_LINIE ( VALUE INT : FROM_X, FROM_Y, TO_X, TO_Y )
BEGIN
  SCHR_GNZ_WRD ( CD_LINIE )
  SCHR_GNZ_WRD ( FROM_X )
  SCHR_GNZ_WRD ( FROM_Y )
  SCHR_GNZ_WRD ( TO_X )
  SCHR_GNZ_WRD ( TO_Y )
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\* **Output a single character** \*\*\*\*\*  
,

; The transfer parameter "CHAR\_VAL" contains the character to be displayed. Its length is 2 ("CHR(2)"),  
; because the end-of-transmission sign ("CHR(0)") also has to be counted.

```
SUBROUTINE SC_ZEICHEN ( VALUE CHR : CHAR_VAL )
BEGIN
  WRITE PHG2000, CHR(CD_TEXT), CHR(2), CHAR_VAL, CHR(0), ..
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\* **Output text string** \*\*\*\*\*  
,

; The variable "DISPLAY" contains the character string to be displayed (in this example the character string  
; "HALLO WORLD").  
; The line length (80 + 1 for "CHR(0)") may be specified for the string length.  
; "CHR(0)" is the end-of-transmission identification of the output.

```
SUBROUTINE SC_TEXT ( VALUE TEXT : DISPLAY )
BEGIN
  WRITE PHG2000, CHR(CD_TEXT), CHR(81), DISPLAY, CHR(0), ..
SUB_END
```

.\*\*\*\*\*  
,

.\*\*\*\*\* **Update display** \*\*\*\*\*  
;

; The transfer parameter "LINE\_NO" contains the line to be updated, the parameter "VAR\_NO" the number of  
; the system variable.

```
SUBROUTINE SC_REFRESH ( VALUE INT : VAR_NO,LINE_NO )
BEGIN
  SCHR_GNZ_WRD (CD_REFRESH)
  SCHR_GNZ_WRD (VAR_NO)
  SCHR_GNZ_WRD (LINE_NO)
SUB_END
```

.\*\*\*\*\*  
;

.\*\*\*\*\* **Move write position by n characters** \*\*\*\*\*  
;

; The transfer parameters "DELTA\_COLUMN" and "DELTA\_LINE" contain the new column and line numbers  
; (relative to the current positions).

```
SUBROUTINE SC_ZEI_REL ( VALUE INT : DELTA_COLUMN, DELTA_LINE )
BEGIN
  SCHR_GNZ_WRD (CD_ZEI_REL)
  WRITE PHG2000, CHR(DELTA_COLUMN), CHR(DELTA_LINE), ..
SUB_END
```

.\*\*\*\*\*  
;

.\*\*\*\*\* **Clear PHG-Display** \*\*\*\*\*  
;

; Clears the entire PHG display.

```
SUBROUTINE SC_CLS
BEGIN
  SCHR_GNZ_WRD ( CD_CLS )
SUB_END
```

.\*\*\*\*\*  
;

.\*\*\*\*\* **Display real value** \*\*\*\*\*  
;

; The transfer parameter "REAL\_VAL" contains the real value to be displayed. The length of this value (in this  
; case "CHR(9)") must also be transmitted to the PHG.  
; "CHR(0)" is the end-of-transmission identification of the output.

```
SUBROUTINE SC_DEZ ( VALUE REAL : REAL_VAL )
BEGIN
  WRITE PHG2000, CHR(CD_TEXT), CHR(9), REAL_VAL, CHR(0), ..
SUB_END
```

.\*\*\*\*\*  
;

```
.***** Display integer number*****
;
; The transfer parameter "INT_VAL" contains the integer value to be displayed.
; Before the value is displayed, the system checks whether an error (W_ERR<>0) has occurred during
; conversion (INT —> ASCII).
; In this case, only hyphens ("-") are displayed instead of the value.
; "CHR(0)" is the end-of-transmission sign of the output.

SOBROUTINE SC_GANZ ( VALUE INT : INT_VAL, LENGTH )
  TEXT : DISPLAY
  INT : W_ERR, I
BEGIN
  INT_ASC (INT_VAL, DISPLAY, 1, LENGTH, W_ERR)
  DISPLAY[LENGTH+1] = CHR(0)
  IF W_ERR<>0 ; <>0= an error has occurred during conversion
    THEN BEGIN
      I=1
      RPT LENGTH TIMES
        DISPLAY[I]='-' ; only display hyphens in the event of an error
        I = I + 1
      RPT_END
    END
  SC_TEXT ( DISPLAY )
SUB_END
.*****
```

## Example:

```
SC_NEUE_MASK (mask2) ;Select mask
SC_ENDE

SC_FUER_MASK (mask2)
SC_CLS ;Clear Screen
SC_ATTRIB (NORMAL) ;Reset attributes
SC_CHRSET (CS_9_MAL_11) ;Character set 9*11
SC_ENDE

SC_FUER_MASK (mask2)
SC_GOTOXY (10,20) ;Cursor position x=10, y=20
SC_TEXT ('Meas.val=') ;Text output
SC_GOTOXY (25,20) ;Cursor position x=25, y=20
SC_GANZ (MV) ;Variable output in INTEGER format
SC_ENDE
```

If the BAPS2 subroutines are not available, the PHG 2000 output can also be initiated directly, i.e. with the commands used in the subroutines.

## 4 Define / Teach In

### 4.1 Introduction

Using the "DEFINE" and "TEACH IN" modes it is possible to display several points from a point file simultaneously on the PHG 2000.

### 4.2 Display layout

#### 4.2.1 "Define" layout

Define Programe (7) Robot 1 (1)	X	Y	Z (8)
Startpos	100.000	100.000	50.000
Pickuppos	-1234.56	-1234.56	-1234.56
Removalpos	100.000	20.000	55.000
Depositpos	1000.05	0.000	66.000
Palletpos	300.000	200.000	70.000
Intermediatepos	11.000	30.005	4.000
Distribpos	400.000	300.000	0.000
Targetpos	50.000	40.000	0.001
Testpos (2)	—	—	—
POS (3)	-1111.11	330.000	0.000
Insert : # (4) (5)			

Fig. 7 – 6 Define layout with three axes

Notes:

- (1) Active kinematics display, in this case **Robot 1**.
- (2) The point **Testpos** has not yet been defined. Therefore, the point components X,Y,Z are represented by "—" characters. For a point array, the array index is shown in the line below the point name.
- (3) The **current position** is displayed according to the selected point (@point/point).

- (4) Value entry for the selected point.

Additional functions:

**Mode + Shift S** : search desired point

**Mode + Shift E** : Exit **Define** mode.

Alternatively, the mode can be exited with **Shift Del**.

- (5) Status messages
- (6) Due to the separating lines in the display, maximally 15 lines can be displayed.
- (7) Name of the PKT file.
- (8) For 10 to 12 axes 4 header lines,  
for 13 to 15 axes 5 header lines, and  
for 16 to 18 axes 6 header lines are needed to display the axis or coordinate names.

For @ (JC) and WC points, the axis and coordinate names are displayed for the selected point.

Define Programme Robot 1	X A	Y B	Z
Startpos	100.000 90.000	400.000 30.000	50.000
Pickuppos	-1234.56 555.000	-1234.56 0.000	-1234.56
Removalpos	100.000 76.005	20.000 54.008	55.000
Depositpos	1000.05 20.000	0.000 30.000	66.000
POS	-1111.11 90.000	330.000 40.000	0.000
Insert : #			

Fig. 7 – 7 Define layout with five axes

**4.2.2 "Teach In" layout**

In addition to the point display, the Teach In mode requires the assignment of axis keys. This results in the following display for three-axis systems in Teach In mode:

Define	X	Y	Z
Programe Robot 1			
Startpos	100.000	100.000	50.000
Pickuppos	-1234.56	-1234.56	-1234.56
Removalpos	100.000	20.000	55.000
Depositpos	1000.05	0.000	66.000
Palletpos	300.000	200.000	70.000
Intermediatepos	11.000	30.005	4.000
Distribpos     Y plus	400.000	300.000	0.000
Targetpos	50.000	40.000	0.001
Testpos	—	—	—
POS (WC) (3)	-1111.11	330.000	0.000
X plus	Y plus	Z plus	(1)
X minus	Y minus	Z minus	(2)

*Fig. 7 – 8 Teach In layout with three axes*

(1),(2) These lines must be generated by the BDT editor. For this purpose, a mask is to be created which contains exactly these two lines. The rho3 enters the positions in the free field of this mask.

(3) The actual position is displayed for the selected coordinate system.

In the example, world coordinates (WC) are active.

The notes on Define apply analogously to Teach In except notes 3 to 5.



Systems with five axes will result in the following layout:

Define Programe Robot 1	X A	Y B	Z
Startpos	100.000 90.000	400.000 30.000	50.000
Pickuppos	-1234.56 555.000	-1234.56 0.000	-1234.56
Removalpos	100.000 76.005	20.000 54.008	55.000
Depositpos	1000.05 20.000	0.000 30.000	66.000
POS (WC)	-1111.11 90.000	330.000 40.000	0.000
X plus	Y plus	Z plus	
X minus	Y minus	Z minus	

Fig. 7 – 9 Teach In layout with five axes

Note:

For systems with four and six axes, the display layout will have the same format as for five axes.

## 4.3 Operation

### 4.3.1 Operation of "Define"

In the "Define" mode the point values are defined by a numerical input. Positioning to the desired point component is by means of the cursor keys. The selected component is displayed in inverse video on the PHG. Point components that have not yet been declared are marked by "—" characters.

Press the "Enter" key to save the new point value.

**4.3.1.1 Operation of "Define" with three axes**

Define Progname Robot 1	X	Y	Z
Startpos	100.000	100.000	50.000
Pickuppos	-1234.56	-1234.56	-1234.56
Removalpos	100.000	20.000	55.000
Depositpos	1000.05	0.000	66.000
Palletpos	300.000	200.000	70.000
Intermediatepos	11.000	30.005	4.000
Distribpos	400.000	300.000	0.000
Targetpos	50.000	40.000	0.001
Testpos	—	—	—
POS	-1111.11	330.000	0.000
Insert : #			

Fig. 7 – 10 Operation of Define with three axes

In the example, the Y component of the point "Removalpos" is to be changed.

Operation steps:

- 1.) Move the cursor with the **cursor keys** to the desired component *Removalpos.Y*.
- 2.) **Numerical input** of the point value in the display next to *Insert: #*.
- 3.) Press the **Enter** key to save the new point value.
- 4.) Use the same sequence of operations to change more point values.

**Mode + Shift S** : search other point

**Mode + Shift E** : exit **Define** mode.

Note:

Point values that have not yet been defined, e.g. for the point Testpos, are defined with the same method.

## 4.3.2 Operation of "Teach In"

### 4.3.2.1 Operation of "Teach In" with three axes

The following example shows a system with three axes. The point to be taught in can be selected by scrolling with the cursor keys.

Define Programe Robot 1	X	Y	Z
Startpos	100.000	100.000	50.000
Pickuppos	-1234.56	-1234.56	-1234.56
Removalpos	100.000	20.000	55.000
Depositpos	1000.05	0.000	66.000
Palletpos	300.000	200.000	70.000
Intermediatepos	11.000	30.005	4.000
Distribpos    Y plus	400.000	300.000	0.000
Targetpos	50.000	40.000	0.001
Testpos	—	—	—
POS (WC)	-1111.11	330.000	0.000
X plus	Y plus	Z plus	
X minus	Y minus	Z minus	

Fig. 7 – 11 Operation of Teach In with three axes

Operation steps:

- 1.) Select the desired point "Removalpos" with the cursor keys.
- 2.) Teaching-in of the new position is done in jogging mode with the axis keys shown in the display. The actual position is shown dynamically in the lower part of the display.
- 3.) Press the "Enter" key to save the new point value.
- 4.) Other point values can be changed through analogous operation.
- 5.) Use the "Move" or the "Move" and the "Linear" keys to select a direct approach to the taught-in points. Movement will be carried out when the "Life man switch" and the "Enter" key are depressed.
- 6.) Press "Shift right-arrow" to search a desired point.
- 7.) Press "Shift left-arrow" to exit this mode.

Note: Points that have not yet been defined, e.g. "Testpos", can be taught in with the same procedure as described in steps 1. to 4.

**4.3.2.2 Operation of "Teach In" with five axes**

The following example shows a system with five axes.

Operation is initially the same as for three axes.

Define Programe Robot 1	X A	Y B	Z
Startpos	100.000 90.000	400.000 30.000	50.000
Pickuppos	-1234.56 555.000	-1234.56 0.000	-1234.56
Removalpos	100.000 76.005	20.000 54.008	55.000
Depositpos	1000.05 20.000	0.000 30.000	66.000
POS (WC)	-1111.11 90.000	330.000 40.000	0.000
X plus	Y plus	Z plus	
X minus	Y minus	Z minus	

*Fig. 7 – 12 Operation of Teach In with five axes*

Operation steps:

- 1.) Select the desired point **Pickuppos** with the cursor keys.
- 2.) Approach the new position in Manual mode using the **axis keys** (X, Y, Z, etc.) shown on the display.  
The actual position is shown dynamically in the POS field in the lower part of the display.
- 3.) For moving axes A and B, first change over the axis key assignment with the **Group** key or another key to be declared.
- 4.) Press the **Enter** key to save the new point value.
- 5.) For defining more points, this process is repeated analogously.

**Shift right-arrow** : search desired point  
**Shift left-arrow** : Exit this mode.

Note:

For systems with several kinematics, change over is done as before with the **KIN** key.

For systems with 13 or more axes the actual position is not displayed.

Your notes:



## 5.2 Creating the objects in the BDT editor

For generating the mask shown in Fig. 7 – 13, you first have to create all required objects by first editing the required texts.

In a second step, the variables marked by jokers are declared in accordance with the PHG 2000 variable code using code **numbers** (Nummer) 34561 or 34567. The **data types** (Datentyp) entered in the BDT editor must be identical with the basic types described for PHG 2000 variable coding.

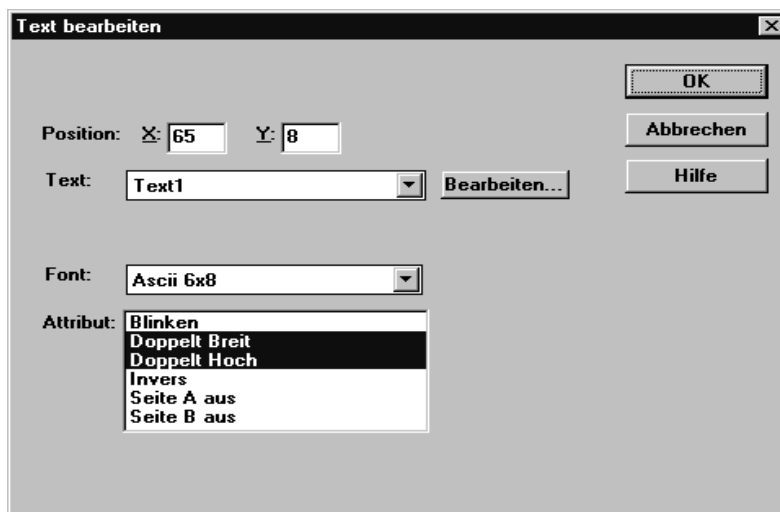
Furthermore, the entry in the **Array** field must specify whether the variable is indexed or simple, and the **Zugriff** (access) field contains the information whether the variable is to be read or written (read/write access).

### 5.2.1 Heading "File list" on the PHG 2000

For the heading, a text variable must be declared so that the text "File list" is displayed. In the BDT editor, the following declaration must be made.

The **TEXT VARIABLE** has the following properties:

Name	: FText1
Text	: Text1
Attribut	: Doppelt hoch (Double height), Doppelt breit (Double width)



The text "File list" may be entered directly in the mask.

### 5.2.2 Listbox for the file list

The variable for 'DATEILISTE' (file list) (no. 34561) must be declared in order to display the file names within the listbox. Once this variable has been declared, you can generate the **variable text list**. The following declarations must be made in the BDT editor.

The two variables have the following properties:

'DATEILISTE' (no. 34561):

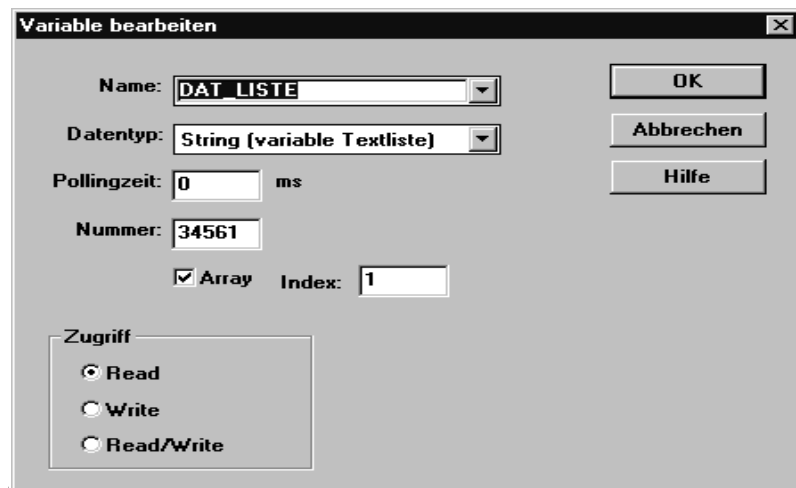


Fig. 7 – 14 Editing variables with 'Variable bearbeiten'

The **TEXT LIST** then has the following properties:



Fig. 7 – 15 Editing the text list with 'Textliste bearbeiten'



### 5.3 Displaying the memory allocation

The variable 'SPEICHER\_BEL' (memory allocation) (no. 34567) must be declared in order to display the memory allocation information of the user memory. Once this variable has been declared, you can generate the text variable. The following declarations must be made in the BDT editor.

**'SPEICHERBELEGUNG' (no. 34567):**

Name : SPEICHER\_BEL  
Datentyp : String  
Nummer : 34567  
Array : no  
Index : -1  
Zugriff : Read access

The **TEXT VARIABLE** then has the following properties:

Name : FVariable2  
Variable : SPEICHER\_BEL  
Länge (length) : 38  
Nachkommastellen  
(decimal places) : 0

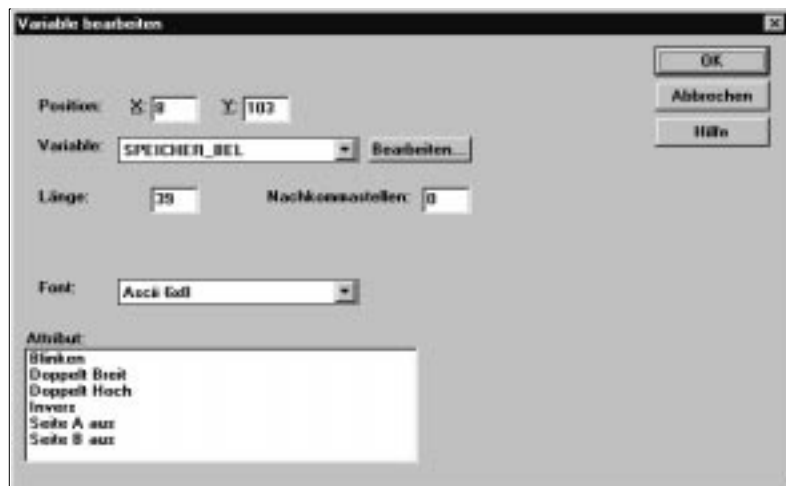


Fig. 7 – 16 Variable for memory allocation

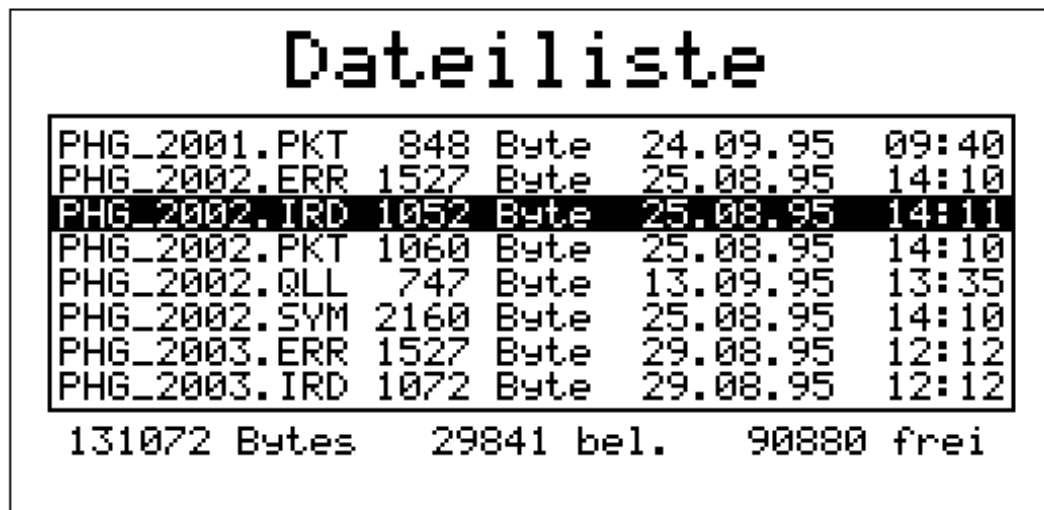
**Note:** The names of the variables, text list and text variable can be freely selected by the user and have no effect on the function.

**5.4 PHG 2000 display**

Fig. 7 – 17 shows the PHG 2000 display of a file list (entitled "Dateiliste") and the memory allocation data. The file display includes the name, size and creation date. The current cursor position is shown in inverse video (in this example, the cursor is on the file "PHG\_2002.IRD" in the listbox). Scrolling within the listbox is with the PHG keys <Shift> and <up-arrow> or <Shift> and <down-arrow>. Thus, you can also list files which currently do not fit in the listbox.

Below this listbox the user memory information is displayed as

- Total memory size
- Memory used
- Free memory.



The screenshot shows a monochrome display with the title "Dateiliste" at the top. Below it is a list of files with columns for filename, size, unit, date, and time. The file "PHG\_2002.IRD" is highlighted with an inverse video effect. At the bottom of the listbox, memory statistics are shown: "131072 Bytes", "29841 bel.", and "90880 frei".

Filename	Size	Unit	Date	Time
PHG_2001.PKT	848	Byte	24.09.95	09:40
PHG_2002.ERR	1527	Byte	25.08.95	14:10
<b>PHG_2002.IRD</b>	<b>1052</b>	<b>Byte</b>	<b>25.08.95</b>	<b>14:11</b>
PHG_2002.PKT	1060	Byte	25.08.95	14:10
PHG_2002.OLL	747	Byte	13.09.95	13:35
PHG_2002.SYM	2160	Byte	25.08.95	14:10
PHG_2003.ERR	1527	Byte	29.08.95	12:12
PHG_2003.IRD	1072	Byte	29.08.95	12:12

131072 Bytes    29841 bel.    90880 frei

Fig. 7 – 17 PHG2000 display

Your notes:

## 6 Process Information

Using the PHG 2000, the user can design the process display according to his needs. The process information display described below is offered as a standard solution and is included as a BDT project in the ROPS package as of version W3C. The control provides the information for the individual processes as coded system variables. All user processes are represented in a listbox of a user-defined size. A process in the list can be selected by scrolling within the listbox. This selected process is shown in inverse video in the listbox. The individual information displayed in the right-hand half of the PHG display, such as priority and QLL line, refer to this selected process.

### 6.1 Mask in the BDT editor

The BDT mask for the process display comprises fixed text objects, variables and a variable text list (listbox). Fig. 7 – 18 shows the display of the BDT editor. The texts are identical to the subsequent PHG display. Variables are represented by the joker "0001" of different length. The length is set in the BDT editor. For the different variable types as well as the variable numbers please refer to the description "PHG 2000 – coding of system variables". The listbox containing the processes on the PHG display is represented by the window marked with '#'.

```

PROZESS-INFO
Anzahl Prozesse      Prozess-Info
normale      : 01      0000000000000001
permanente   : 01      Prioritaet      :0001
sub          : 01      Zustand        : 000001
fehlerhafte  : 01      Prozess-Art    :000001
Prozess-Liste      Kinematik      :0001
#####           QLL - Zeile    :0001
#####           QLL-Einf.Zeile:0001
#####           Ext. Programm :00000001
#####           Ebene        :0001
#####           Prozess-Fehler:0001
#####           00000000000000000001
    
```

Fig. 7 – 18 Process information: Mask in BDT editor

## 6.2 Generating the objects in the BDT editor

For generating the mask shown in Fig. 7 – 18, you first have to create all required objects by first editing the necessary **16 texts**.

<b>Anzahl Prozesse</b>	1	<b>Prozess Info</b>		
<b>normale</b>	: 2			
<b>permanente</b>	: 3			
<b>sub</b>	: 4			
<b>fehlerhafte</b>	: 5		<b>Prioritaet</b>	: 8
<b>Prozess-Liste</b>	6		<b>Zustand</b>	: 9
		<b>Prozess-Art</b>	: 10	
		<b>Kinematik</b>	: 11	
		<b>QLL-Zeile</b>	: 12	
		<b>QLL-Einf.Zeile</b>	: 13	
		<b>Ext. Programm</b>	: 14	
		<b>Ebene</b>	: 15	
		<b>Prozess-Fehler</b>	: 16	

Fig. 7 – 19 Process information texts

In a second step, the **variables** (cf. chapter 3) marked by jokers are declared in accordance with the PHG 2000 variable code using code numbers 34304..34322. The data types (Datentyp) entered in the BDT editor must be identical with the basic types described for PHG 2000 variable coding. Furthermore, it must be specified whether the variable is indexed or simple, and whether the variable is to be read or written (read/write access). For the complete process display, 17 variables must be declared. Using the variable (no. 34322) it is possible to stop a selected process.





## 6.2.2 Global process information

The variables with the numbers 34305–34308 serve for the display of global process information such as the number of normal and permanent processes. The declarations in the BDT editor are to be made as shown in the following two examples.

Name	Data type	Polling	Code no.	Array	Index	Access
ANZ_NORM_PROZ	Word	0	34305	No	-1	R
ANZ_PERM_PROZ	Word	0	34306	No	-1	R

R = Read, W = Write

## 6.2.3 Information on the selected process

All other variables refer to the selected process, e.g. process priority (Prioritaet) or QLL line (QLL-Zeile). It is not necessary to use all variables. For example, if a user does not use any external programs or INCLUDE instructions, he may skip the texts 'Ext. Programm :', 'Ebene :' (level) and 'QLL-Einf.Zeile :' (QLL INCLUDE line) and their related variables. The BDT variable declarations will be explained by the two examples of process priority and QLL line.

Name	Data type	Polling	Code no.	Array	Index	Access
PROZ_PRIO	Word	0	34312	No	-1	R
QLL_ZEILE	Word	0	34317	No	-1	R

R = Read, W = Write



## 6.3 PHG 2000 display

Fig. 7 – 22 shows the PHG 2000 display for 6 active user processes. The process 'ROBOT\_A' has been selected. The information in the right-hand half of the display refers to this process. As mentioned above, scrolling within the process listbox is with the PHG keys <Shift> and <up-arrow> or <Shift> and <down-arrow> for selecting, e.g., the previous process "PAR\_EA S03" or the subsequent process "ROBOT\_B".

The variable 'Prozessname' (process name) (no. 34316) is used in this example (Fig. 7 – 22) to display the name of the selected process in the right-hand display field under the heading "Prozess-Info". If no process is active, this field as well as the listbox are empty.

Process errors are displayed including an error code and an error text 20 characters in length. In the example for the process display (Fig. 7 – 22) no process error has occurred. The error code displayed is 0, and 20 blanks are shown as error text. The location of the error text displayed in the event of an error instead of the joker '00000000000000000001' can be recognized in Fig. 7 – 18.

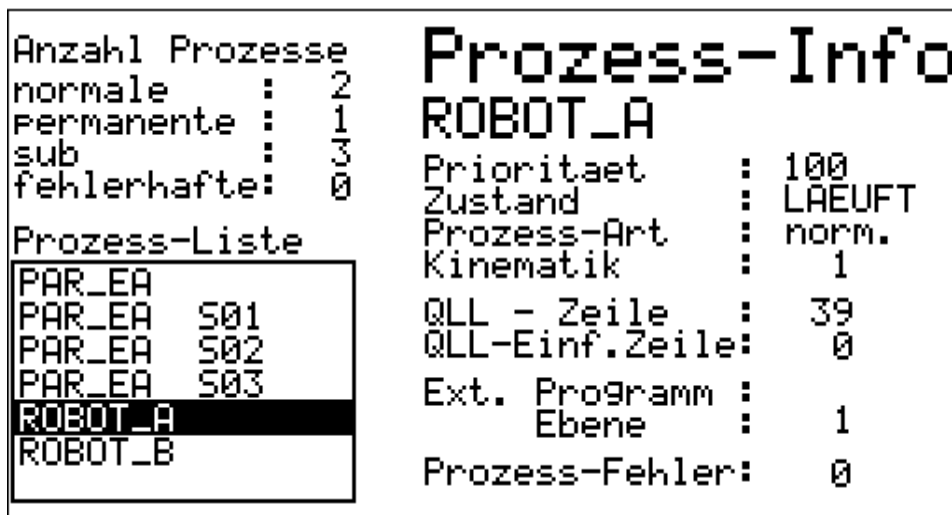


Fig. 7 – 22 PHG2000 display

## 6.4 Stopping processes

The system variable with the code number 34322 can be used to stop processes easily from the process display. If any PHG key (e.g. the <Delete> key) is linked to this variable, the currently active process can be stopped from the process display using this key. The effect is similar to the operation of "PHG mode 10.2: Stop processes" of the PHG3. The PHG 2000 function is different in that there is no verification such as "stop processes?", and furthermore individual SUB processes can be stopped, which is not possible in PHG mode 10.2.

Due to the higher risk of unintentional stopping of processes, this function is not active in the standard process information display, however, it can be implemented by the user in the **BDT editor** as follows:

- 1) Declare system variable no. 34322 ( '**Stop process**' ):

Name	Data type	Polling	Code no.	Array	Index	Access
STP_ANGEW_PROZESS	Byte	0	34322	No	-1	W

- 2) Assign the desired key in the BDT editor, '**Maske / Taste**' (mask / key) menu:  
(In the example, the <Delete> key K35 is used <normally>)

Name	Key	Icon	Action	Value (Press)	Value (Release)
STP_ANGEW_PROZESS	K35 norm.	no icon	Variable	1	0

This declaration means that the currently selected process is stopped when the <Delete> key is depressed.

More protection against unintentional stopping of processes can be achieved by selecting a more complicated key combination (e.g. <Shift> + <Delete>) instead of the <Delete> key.

Your notes:

## 7 Variable assignment of PHG keys

### 7.1 Description

The PHG has 36 keys with a standard assignment on each of the following levels:

- Normal level
- 'ALT' level (press 'ALT' key simultaneously)
- 'SHIFT' level (press 'SHIFT' key simultaneously)
- Level for PIC monitor and PIC editor (MODE 3.2.1 and MODE 6.1)

The standard key assignment can be modified with the **KEY.BNR** file. This file name is fixed. The file length is exactly 256 bytes (character number 0 to character number 255). The second column of tables 2–5 in the annex contains the standard assignment (key code). Table 1 shows the indices of the individual keys which correspond to tables 2–5. The key indices in table 1 apply to each of the four levels.

**If a file named KEY.BNR is available in the user memory when the rho3 starts up, key assignment will be as defined by this file.** If this file is incorrect or unavailable, the standard assignment will be used. The rho3 will not generate an error message if the KEY.BNR file is incorrect.

The KEY.BNR file is created with a commercially available HEX editor. The standard assignment stored in the file **STA\_TAST.BNR** file is taken as a basis (STA\_TAST.BNR is included in the delivery as of ROPS version W3C). Table 1 shows the key indices of the individual keys. They indicate the sequence of entries in the KEY.BNR file on the respective level.

The following character numbers are fixed and must not be modified:

Character number	Contents (key code)	
	HEX	ASCII
0	5A	'Z'
1–7	20	' '
44–63	20	' '
64	5A	'Z'
65–71	20	' '
108–127	20	' '
128	5A	'Z'
129–135	20	' '
172–191	20	' '
192	5A	'Z'
193–199	20	' '
236–255	20	' '

If a key is to remain without an effect, code 00H (hex) must be entered for this key.

When the **ALT** and **SHIFT** levels are used, the keys ALT and SHIFT must be in the same location as in the standard assignment.

If the keys ALT and/or SHIFT are assigned a code other than 00H, they will produce the corresponding effect, and the ALT and/or SHIFT level no longer exists.

Please refer to the standard key assignment and the control response for the effect on the individual key codes.

If, due to an incorrect KEY.BNR file, the control can no longer be operated from the PHG, the KEY.BNR file can be deleted with the help of ROPS3. An EPROM backup will also restore the standard key assignment and provide for control operation from the PHG.

The multiple key operations for initiating an EPROM BACKUP, EEPROM BACKUP, etc. always remain on the same physical keys as in the standard assignment, even if a KEY.BNR file specifies different key codes than the standard assignment.

## Example 1:

### Requirement:

The K1 key in the normal level (cf. table 1) shall have the same effect as the K36 key in the normal level (ENTER key in standard assignment).

### How to proceed:

- Derive the key index for ENTER, 23, from table 1.
- The key index 23 (column 6) results in the key code for ENTER, i.e. 0A H, as shown in table 2/1 (column 3).
- This key code must be entered for the K1 key in KEY.BNR.
- The K1 key has key index 40 (cf. table 1).
- As shown in table 2/2 (column 6), the key index 40 results in the character number (index in KEY.BNR, column 1) 40.
- Enter the code for ENTER, 0A H, for character number 40 (instead of 74H, 't', in standard assignment) in KEY.BNR.

## Example 2:

### Requirement:

The K10 key in the ALT level (cf. table 1) shall have the same effect as the K29 key in the SHIFT level (SHIFT LEFT-ARROW key in standard assignment).

### How to proceed:

- Derive the key index for the K29 key, 9, from table 1.
- The key index 9 (column 7) results in the key code for LEFT-ARROW (SHIFT level), i.e. 18 H, as shown in table 3/1 (column 3). This key code must be entered for the K10 key in the ALT level in KEY.BNR.
- The K10 key has key index 37 (cf. table 1).
- As shown in table 4/2 (column 8), the key index 37 results in the character number in the ALT level (index in KEY.BNR) 165.
- Enter the code for LEFT-ARROW, 18 H, for character number 165 (instead of 5 D H, ']', in standard assignment) in KEY.BNR.

## Example 3:

### Requirement:

The K28 key (ALT) (cf. table 1) shall have the same effect as the K33 key in the normal level ('0' in standard assignment). This will cancel all functions of the ALT level.

### How to proceed:

- Enter the code for '0', 30 H, for character number 19 (instead of 00 H in standard assignment).

## Example 4:

### Requirement:

The keys K1 to K8 (cf. table 1) shall have no effect in the normal level.

### How to proceed:

- Enter code 00 H for the following character numbers in KEY.BNR:
- K1 key, character number 40 (00 H instead of 74 H, 't' in standard assignment)
- K2 key, character number 41 (00 H instead of 72 H, 'r' in standard assignment)
- K3 key, character number 42 (00 H instead of 69 H, 'i' in standard assignment)
- K4 key, character number 43 (00 H instead of 6B H, 'k' in standard assignment)
- K5 key, character number 38 (00 H instead of 6D H, 'm' in standard assignment)
- K6 key, character number 39 (00 H instead of 66 H, 'f' in standard assignment)
- K7 key, character number 30 (00 H instead of 73 H, 's' in standard assignment)
- K8 key, character number 31 (00 H instead of 67 H, 'g' in standard assignment)



## CAUTION

### 7.5

**The machine manufacturer bears the sole responsibility for unintended movements possibly caused by the selection of an incorrect version of the KEY.BNR file (keyboard assignment). It is recommended that any modification of the standard keyboard assignment be performed subject to prior consultation with BOSCH.**

A MOVE -	B LINEAR ;	1+ VIA	C TO	1- D		
<b>40</b> <i>K1</i>	<b>41</b> <i>K2</i>	<b>42</b> <i>K3</i>	<b>43</b> <i>K4</i>			
E WAIT ?	F UNTIL '	2+ BEGIN	G END	2- H		
<b>38</b> <i>K5</i>	<b>39</b> <i>K6</i>	<b>30</b> <i>K7</i>	<b>31</b> <i>K8</i>			
I IF [	J THEN ]	3+ ELSE	K JUMP	3- L		
<b>36</b> <i>K9</i>	<b>37</b> <i>K10</i>	<b>28</b> <i>K11</i>	<b>29</b> <i>K12</i>			
M RPT (	N TIMES )	4+ RPT_END	O HALT	4- P		
<b>34</b> <i>K13</i>	<b>35</b> <i>K14</i>	<b>26</b> <i>K15</i>	<b>27</b> <i>K16</i>			
INFO GROUP MODE	KIN SPACE COORDINATES	5+ V_PTP	Q =	5- R		
<b>32</b> <i>K17</i>	<b>33</b> <i>K18</i>	<b>24</b> <i>K19</i>	<b>25</b> <i>K20</i>			
S 7 <	T 8 >	6+ 9	U -	6- V		
<b>15</b> <i>K21</i>	<b>16</b> <i>K22</i>	<b>17</b> <i>K23</i>	<b>18</b> <i>K24</i>			
W 4 +	UP-ARROW * 5	X 6 :	ALT			
<b>12</b> <i>K25</i>	<b>13</b> <i>K26</i>	<b>14</b> <i>K27</i>	<b>19</b> <i>K28</i>			
LEFT-ARROW ! 1	Y 2 /	RIGHT-ARROW % 3	SHIFT			
<b>9</b> <i>K29</i>	<b>10</b> <i>K30</i>	<b>11</b> <i>K31</i>	<b>20</b> <i>K32</i>			
Z 0 @	DOWN-ARROW ,	DELETE	ENTER			
<b>8</b> <i>K33</i>	<b>21</b> <i>K34</i>	<b>22</b> <i>K35</i>	<b>23</b> <i>K36</i>			

Abb. 7 – 23 Table 1: PHG key index



Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
0	00	5A	'Z'	–	1	–	–	–
1	01	20	' '	–	–	–	–	–
2	02	20	' '	–	–	–	–	–
3	03	20	' '	–	–	–	–	–
4	04	20	' '	–	–	–	–	–
5	05	20	' '	–	–	–	–	–
6	06	20	' '	–	–	–	–	–
7	07	20	' '	–	–	–	–	–
8	08	30	'0'	0	8	–	–	–
9	09	31	'1'	1	9	–	–	–
10	0A	32	'2'	2	10	–	–	–
11	0B	33	'3'	3	11	–	–	–
12	0C	34	'4'	4	12	–	–	–
13	0D	35	'5'	5	13	–	–	–
14	0E	36	'6'	6	14	–	–	–
15	0F	37	'7'	7	15	–	–	–
16	10	38	'8'	8	16	–	–	–
17	11	39	'9'	9	17	–	–	–
18	12	2D	'_'	–	18	–	–	–
19	13	00		ALT	19	–	–	–
20	14	00		SHIFT	20	–	–	–
21	15	2E	'.'	.	21	–	–	–
22	16	10		DELETE	22	–	–	–
23	17	0A		ENTER	23	–	–	–
24	18	6F	'o'	V_PTP	24	–	–	–
25	19	3D	'='	=	25	–	–	–
26	1A	65	'e'	RPT_END	26	–	–	–
27	1B	68	'h'	HALT	27	–	–	–
28	1C	64	'd'	ELSE	28	–	–	–
29	1D	6A	'j'	JUMP	29	–	–	–
30	1E	73	's'	BEGIN	30	–	–	–
31	1F	67	'g'	END	31	–	–	–

Abb. 7 – 24 Table 2/1: Standard key assignment on the normal level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
32	20	7B	{	MODE	32	–	–	–
33	21	20	“	COORDINATES	33	–	–	–
34	22	6C		RPT	34	–	–	–
35	23	6E	n	TIMES	35	–	–	–
36	24	71	q	IF	36	–	–	–
37	25	70	p	THEN	37	–	–	–
38	26	6D	m	WAIT	38	–	–	–
39	27	66	f	UNTIL	39	–	–	–
40	28	74	t	MOVE	40	–	–	–
41	29	72	r	LINEAR	41	–	–	–
42	2A	69	i	VIA	42	–	–	–
43	2B	6B	k	TO	43	–	–	–
44	2C	20	“	–	–	–	–	–
45	2D	20	“	–	–	–	–	–
46	2E	20	“	–	–	–	–	–
47	2F	20	“	–	–	–	–	–
48	30	20	“	–	–	–	–	–
49	31	20	“	–	–	–	–	–
50	32	20	“	–	–	–	–	–
51	33	20	“	–	–	–	–	–
52	34	20	“	–	–	–	–	–
53	35	20	“	–	–	–	–	–
54	36	20	“	–	–	–	–	–
55	37	20	“	–	–	–	–	–
56	38	20	“	–	–	–	–	–
57	39	20	“	–	–	–	–	–
58	3A	20	“	–	–	–	–	–
59	3B	20	“	–	–	–	–	–
60	3C	20	“	–	–	–	–	–
61	3D	20	“	–	–	–	–	–
62	3E	20	“	–	–	–	–	–
63	3F	20	“	–	–	–	–	–

Abb. 7 – 25 Table 2/2: Standard key assignment on the normal level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
64	40	5A	'Z'	–	–	1	–	–
65	41	20	''	–	–	–	–	–
66	42	20	''	–	–	–	–	–
67	43	20	''	–	–	–	–	–
68	44	20	''	–	–	–	–	–
69	45	20	''	–	–	–	–	–
70	46	20	''	–	–	–	–	–
71	47	20	''	–	–	–	–	–
72	48	5A	'Z'	Z	–	8	–	–
73	49	18		Left-arrow	–	9	–	–
74	4A	59	'Y'	Y	–	10	–	–
75	4B	0E		Right-arrow	–	11	–	–
76	4C	57	'W'	W	–	12	–	–
77	4D	0F		Up-arrow	–	13	–	–
78	4E	58	'X'	X	–	14	–	–
79	4F	53	'S'	S	–	15	–	–
80	50	54	'T'	T	–	16	–	–
81	51	55	'U'	U	–	17	–	–
82	52	56	'V'	V	–	18	–	–
83	53	00		ALT	–	19	–	–
84	54	00		SHIFT	–	20	–	–
85	55	0C		Down-arrow	–	21	–	–
86	56	10		DELETE	–	22	–	–
87	57	0A		ENTER	–	23	–	–
88	58	51	'Q'	Q	–	24	–	–
89	59	52	'R'	R	–	25	–	–
90	5A	4F	'O'	O	–	26	–	–
91	5B	50	'P'	P	–	27	–	–
92	5C	4B	'K'	K	–	28	–	–
93	5D	4C	'L'	L	–	29	–	–
94	5E	47	'G'	G	–	30	–	–
95	5F	48	'H'	H	–	31	–	–

Abb. 7 – 26 Table 3/1: Standard key assignment on the SHIFT level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
96	60	7A	'Z'	INFO	—	32	—	—
97	61	07		KIN	—	33	—	—
98	62	4D	'M'	M	—	34	—	—
99	63	4E	'N'	N	—	35	—	—
100	64	49	'I'	I	—	36	—	—
101	65	4A	'J'	J	—	37	—	—
102	66	45	'E'	E	—	38	—	—
103	67	46	'F'	F	—	39	—	—
104	68	41	'A'	A	—	40	—	—
105	69	42	'B'	B	—	41	—	—
106	6A	43	'C'	C	—	42	—	—
107	6B	44	'D'	D	—	43	—	—
108	6C	20	' '	—	—	—	—	—
109	6D	20	' '	—	—	—	—	—
110	6E	20	' '	—	—	—	—	—
111	6F	20	' '	—	—	—	—	—
112	70	20	' '	—	—	—	—	—
113	71	20	' '	—	—	—	—	—
114	72	20	' '	—	—	—	—	—
115	73	20	' '	—	—	—	—	—
116	74	20	' '	—	—	—	—	—
117	75	20	' '	—	—	—	—	—
118	76	20	' '	—	—	—	—	—
119	77	20	' '	—	—	—	—	—
120	78	20	' '	—	—	—	—	—
121	79	20	' '	—	—	—	—	—
122	7A	20	' '	—	—	—	—	—
123	7B	20	' '	—	—	—	—	—
124	7C	20	' '	—	—	—	—	—
125	7D	20	' '	—	—	—	—	—
126	7E	20	' '	—	—	—	—	—
127	7F	20	' '	—	—	—	—	—

Abb. 7 – 27 Table 3/2: Standard key assignment on the SHIFT level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
128	80	5A	'Z'	–	–	–	1	–
129	81	20	' '	–	–	–	–	–
130	82	20	' '	–	–	–	–	–
131	83	20	' '	–	–	–	–	–
132	84	20	' '	–	–	–	–	–
133	85	20	' '	–	–	–	–	–
134	86	20	' '	–	–	–	–	–
135	87	20	' '	–	–	–	–	–
136	88	40	'@'	@	–	–	8	–
137	89	21	'!'	!	–	–	9	–
138	8A	2F	'/'	/	–	–	10	–
139	8B	25	'%'	%	–	–	11	–
140	8C	2B	'+'	+	–	–	12	–
141	8D	2A	'*'	*	–	–	13	–
142	8E	3A	':'	:	–	–	14	–
143	8F	3C	'<'	<	–	–	15	–
144	90	3E	'>'	>	–	–	16	–
145	91	06		6+	–	–	17	–
146	92	20	' '	6–	–	–	18	–
147	93	00		ALT	–	–	19	–
148	94	00		SHIFT	–	–	20	–
149	95	2C	','	,	–	–	21	–
150	96	10		DELETE	–	–	22	–
151	97	0A		ENTER	–	–	23	–
152	98	20	' '	5+	–	–	24	–
153	99	20	' '	5–	–	–	25	–
154	9A	20	' '	4+	–	–	26	–
155	9B	20	' '	4–	–	–	27	–
156	9C	20	' '	3+	–	–	28	–
157	9D	20	' '	3–	–	–	29	–
158	9E	20	' '	2+	–	–	30	–
159	9F	20	' '	2–	–	–	31	–

Abb. 7 – 28 Table 4/1: Standard key assignment on the ALT level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
160	A0	08		GROUP	–	–	32	–
161	A1	20	' '	SPACE	–	–	33	–
162	A2	28	'('	(	–	–	34	–
163	A3	29	')'	)	–	–	35	–
164	A4	5B	'['	[	–	–	36	–
165	A5	5D	']'	]	–	–	37	–
166	A6	3F	'?'	?	–	–	38	–
167	A7	27	'	'	–	–	39	–
168	A8	5F	'_'	_	–	–	40	–
169	A9	3B	','	;	–	–	41	–
170	AA	20	' '	1+	–	–	42	–
171	AB	20	' '	1–	–	–	43	–
172	AC	20	' '	–	–	–	–	–
173	AD	20	' '	–	–	–	–	–
174	AE	20	' '	–	–	–	–	–
175	AF	20	' '	–	–	–	–	–
176	B0	20	' '	–	–	–	–	–
177	B1	20	' '	–	–	–	–	–
178	B2	20	' '	–	–	–	–	–
179	B3	20	' '	–	–	–	–	–
180	B4	20	' '	–	–	–	–	–
181	B5	20	' '	–	–	–	–	–
182	B6	20	' '	–	–	–	–	–
183	B7	20	' '	–	–	–	–	–
184	B8	20	' '	–	–	–	–	–
185	B9	20	' '	–	–	–	–	–
186	BA	20	' '	–	–	–	–	–
187	BB	20	' '	–	–	–	–	–
188	BC	20	' '	–	–	–	–	–
189	BD	20	' '	–	–	–	–	–
190	BE	20	' '	–	–	–	–	–
191	BF	20	' '	–	–	–	–	–

Abb. 7 – 29 Table 4/2: Standard key assignment on the ALT level

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
192	C0	5A	'Z'	–	–	–	–	1
193	C1	20	' '	–	–	–	–	–
194	C2	20	' '	–	–	–	–	–
195	C3	20	' '	–	–	–	–	–
196	C4	20	' '	–	–	–	–	–
197	C5	20	' '	–	–	–	–	–
198	C6	20	' '	–	–	–	–	–
199	C7	20	' '	–	–	–	–	–
200	C8	30	'0'	0	–	–	–	8
201	C9	31	'1'	1	–	–	–	9
202	CA	32	'2'	2	–	–	–	10
203	CB	33	'3'	3	–	–	–	11
204	CC	34	'4'	4	–	–	–	12
205	CD	35	'5'	5	–	–	–	13
206	CE	36	'6'	6	–	–	–	14
207	CF	37	'7'	7	–	–	–	15
208	D0	38	'8'	8	–	–	–	16
209	D1	39	'9'	9	–	–	–	17
210	D2	2D	'_'	–	–	–	–	18
211	D3	00		ALT	–	–	–	19
212	D4	00		SHIFT	–	–	–	20
213	D5	2E	'.'	.	–	–	–	21
214	D6	10		DELETE	–	–	–	22
215	D7	0A		ENTER	–	–	–	23
216	D8	6F	'o'		–	–	–	24
217	D9	3D	'='		–	–	–	25
218	DA	1F		EP	–	–	–	26
219	DB	1E		NOP1	–	–	–	27
220	DC	4D	'M'	M	–	–	–	28
221	DD	6A	'j'		–	–	–	29
222	DE	41	'A'	O	–	–	–	30
223	DF	67	'g'		–	–	–	31

Abb. 7 – 30 Table 5/1: Standard key assignment on the level for PIC monitor and PIC editor

Index in KEY.BNR (character number)		Contents of KEY.BNR with standard key assignment (key code)		Inscription on and meaning of the PHG keys with standard assignment	PHG key index in the normal level	PHG key index in the SHIFT level	PHG key index in the ALT level	PHG key index in the level for PIC monitor and PIC editor
DEC	HEX	HEX	ASCII					
224	E0	7B	{	MODE	–	–	–	32
225	E1	20	' '		–	–	–	33
226	E2	3D	'='	=	–	–	–	34
227	E3	62	'b'	JC	–	–	–	35
228	E4	73	's'	S	–	–	–	36
229	E5	1D		R	–	–	–	37
230	E6	1C		O	–	–	–	38
231	E7	1A		ON	–	–	–	39
232	E8	1B		A	–	–	–	40
233	E9	75	'u'	AN	–	–	–	41
234	EA	45	'E'	I	–	–	–	42
235	EB	6B	'k'		–	–	–	43
236	EC	20	' '	–	–	–	–	–
237	ED	20	' '	–	–	–	–	–
238	EE	20	' '	–	–	–	–	–
239	EF	20	' '	–	–	–	–	–
240	F0	20	' '	–	–	–	–	–
241	F1	20	' '	–	–	–	–	–
242	F2	20	' '	–	–	–	–	–
243	F3	20	' '	–	–	–	–	–
244	F4	20	' '	–	–	–	–	–
245	F5	20	' '	–	–	–	–	–
246	F6	20	' '	–	–	–	–	–
247	F7	20	' '	–	–	–	–	–
248	F8	20	' '	–	–	–	–	–
249	F9	20	' '	–	–	–	–	–
250	FA	20	' '	–	–	–	–	–
251	FB	20	' '	–	–	–	–	–
252	FC	20	' '	–	–	–	–	–
253	FD	20	' '	–	–	–	–	–
254	FE	20	' '	–	–	–	–	–
255	FF	20	' '	–	–	–	–	–

Abb. 7 – 31 Table 5/2: Standard key assignment on the level for PIC monitor and PIC editor



Your notes:

## 8 Selecting a PKT file or point name

### 8.1 General

With the help of special function 16, a special **PKT file** can be selected for modes **DEFINE** (Mode 4.1), **TEACH IN** (Mode 4.2) and **PRINT PKT FILE** (Mode 9.4.2), which is immediately offered for display when the corresponding mode is selected and without having to press any other key. For the modes **DEFINE/TEACH IN**, a **point name** can be additionally preselected within the selected PKT file.

### 8.2 Declaration

Special function 16 is declared as follows in the BAPS program:

```
SPC_FCT : 16 = PNT_SELECT (TEXT: PNT_FNAM  
                          TEXT: PNT_PNAM  
                          INT: RET_CODE)
```

Whereby

- **PNT\_SELECT** is a user-defined name of the special function
- **PNT\_FNAM** is the name of the PKT file displayed for **DEFINE/TEACH IN**. The file name must be entered with its extension 'PKT'.
- **PNT\_PNAM** is the name of the point offered within the file.
- **RET\_CODE**

Return code = 0	everything o.k.
= 1	incorrect file name
= 2	incorrect extension (PKT required)
= 3	file does not exist
= 4	file cannot be opened
= 5	file is empty
= 11	incorrect point name
= 12	point not found in file
= 13	index not found (for PKT array)

It is also possible to enter an 'empty name' as point name, e.g. PNAM = ' '. In this case, the first point of the PKT file is displayed for 'DEFINE' / 'TEACH IN'.

The point name has no effect on mode 'PRINT PKT FILE', instead, the entire file will always be printed.

The selected names are reset when the corresponding mode is selected.

### 8.3 Example

```
PROGRAM SPCF16

SPC_FCT : 16 = PNT_SELECT ( TEXT: PNT_FNAM
                           TEXT: PNT_PNAM
                           INT: RET_CODE)

TEXT    : PNTFNAM,PNTPNAM
INT     : RET

BEGIN
  PNTFNAM = 'TEST.PKT'
  PNTPNAM = 'STARTPOS'
  RET = 0
  PNT_SELECT (PNTFNAM,PNTPNAM,RET)
PROGRAM_END
```



## 9 Reversing an absolute measuring system

### 9.1 General

The direction of rotation of absolute measuring systems is usually defined by a suitable circuitry or by setting the corresponding encoder parameters.

The reversing function of the rho3 provides for the use of encoders or converters (e.g. resolver ==> abs.) which do not offer this feature.

### 9.2 Function

The direction of rotation of absolute measuring systems is set with the "Direction" subparameter of **P401**.

Direction = 1: Positive direction of the encoders results in positive direction within the rho3.

Direction = -1: Positive direction of the encoders results in negative direction within the rho3.

The sign of the measuring system evaluation has no effect on the direction of rotation of the absolute encoders.

### 9.3 Compatibility

The default value of the Direction is 0 for machine parameter sets of old SW versions. This means that the direction is not reversed. This is to ensure that a software update of existing plants does not result in a functional change of the absolute encoders.

Your notes:

# 10 IAT Programming

## 10.1 General

In order to ensure that the IAT module can be used as a PLC central unit in the PLC rack together with the rho3.0, it is now possible to create executable PLC programs for the rho3.0 on the IAT module.

## 10.2 Requirements

### 10.2.1 Hardware

- PLC basic unit rack
- IAT module
- rho3.0

#### 10.2.1.1 Switch setting on the IAT module

The switches on the IAT module must be set as follows:



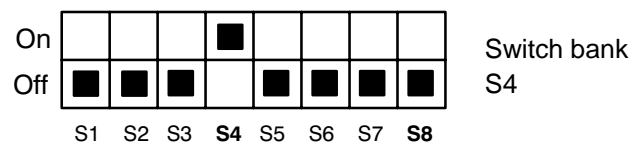
S1, S2 : **Base address**

S1 = OFF, S2 = OFF: Base address = C800

S1 = OFF, S2 = ON : Base address = D800

The set base address must be identical with the base address defined in **IATEA.H**. The default is **D800**.

S4 : ON **Activate Reset on front cover**



S4 : ON **IRQ 15** for module interrupt **on**

S8 : OFF for a **386** processor module

ON for a **486** processor module



## 10.3 Creating PLC programs

For creating system-specific PLC programs, the C program (basis: R3\_vv\_D.C) and maybe also the symbol file (basis: R3S\_vv\_D.H) must be changed on the basis of the standard PLC program provided. Programming knowledge of BORLAND C++ (or Microsoft Visual C++) is indispensable.

### 10.3.1 Recommended procedure

- Copy the files to be changed  
**R3\_vv\_D.C** ==> program name.C  
**R3S\_vv\_D.H** ==> symbol file.H
- Call up BORLANDC (or Microsoft Visual C ++)
- Create a new project  
Copy **R3\_vv\_D.PRJ** ==> projectname.PRJ  
The default settings made under R3\_vv\_D.PRJ must not be changed.
- Create the symbol file  
On the basis of the standard symbol file all additional I/O signals and/or flags, if necessary, are declared in the symbol file. For a description of the macros used, please refer to the IAT manual (Software section).
- Create the PLC program  
On the basis of the standard PLC program you can create a system-specific PLC program. All functions of the C programming language are available for this purpose. For implementing PLC-specific functions, the macros described in the IAT manual (Software section) are additionally available.
- Compile and link the finished programs.
- Exit BORLANDC (or Microsoft Visual C++) and start the new program.  
  
vv = rho 3 operating system version



## 10.3.2 Special features

### 10.3.2.1 Data types

All data types admissible in C can be used for internal calculations within the PLC program. For data transmission to the rho3 and the I/O boards, the macros described in the IAT manual (Software section) must be used.

## 10.4 Executing the PLC program

The new program is run **as TSR in the background**. The program is **started in a fixed time scale** by a timer interrupt. This time can be set by the user. For this purpose, the constant ZYKLUS\_ZEIT (in [ms]) was defined in the symbol file (R3S\_6C\_D). The default time is **20ms**.

The cycle time must be longer than the maximum processing time of the PLC program. If other applications run in parallel to the PLC program, suitable time reserves should be provided for.

The processing time can be estimated according to the following formula:

$$\begin{aligned} & 1.2 \text{ ms} \\ + & 100 \mu\text{s per input or output board} \\ + & 6 \mu\text{s per C instruction in the PLC program} \\ & \text{(valid for 386 processor, 25 MHz)} \end{aligned}$$

### 10.4.1 Displays

- When the *IAT has been started* the **red STOP/RUN LED** at the front side of the IAT module is lighted.  
The status display shows **.8**.
- An *error-free execution* of the PLC program will (if IAT\_R30.C is installed) **clear both displays**.
- When the rho3 *no longer transmits data*, the status display shows the value **8** (timeout).

It is possible to generate user-specific displays. For this purpose, IAT\_R30.C must be changed accordingly. The procedures for addressing the displays are described in the IAT manual (Software section).

## 10.5 Automatic start-up

In order to ensure automatic start-up of the system, the PLC program call should be included in the AUTOEXEC.BAT file.

**Since the IAT module generally has a longer start-up time than the rho3.0, measures must be taken to ensure that the start-up of the IAT module is completed earlier than the start-up of the rho3.0, e.g. do not switch on the rho3.0 before the READY relay on the PLC power supply is closed (IAT start-up complete).**

Your notes:

# 11 rho3 coupling via CAN bus

## 11.1 Coupling of several rho3 controls

The digital I/O signals via the CAN bus are used for coupling several rho3 controls. Since each control generates its own cycle, it is not possible to synchronize all controls to a common cycle. For this reason, it cannot be guaranteed that an input telegram can be received in each cycle. In order to ensure data transmission to and from rho3 controls, time monitoring was therefore slowed down specifically for this application.

For further information, please refer to chapter *Variable assignment of the CAN interface* (version TO05).

### 11.1.1 Linking the controls

A CAN bus must be reserved for linking the rho3 controls. No other devices may be connected to this bus.

### 11.1.2 Setting the identifiers

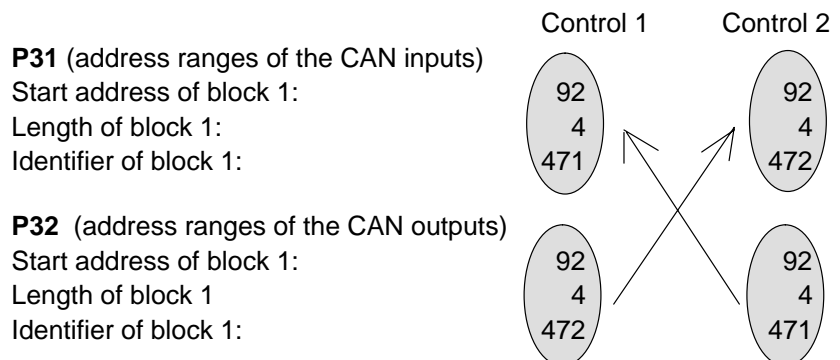
Reserved identifiers for coupling digital I/O modules:

- Outputs:: 471..480
- Inputs: 541..550

Of these, only identifiers 471..480 are used for coupling rho3 controls. For performing data transmission, the same identifiers must be set on the controls in question.

#### Example 1:

Data transmission to and from 2 controls



In example 1, the output signal bytes 92..95 of control 1 are transmitted to the input bytes 92..95 of control 2 via identifier 472.

In the opposite direction, the output signal bytes 92..95 of control 2 are transmitted to input bytes 92..95 of control 1 via identifier 471.

As shown above, identifiers of one control, which were originally reserved for outputs, are used as input identifiers. This setting automatically has the effect that time monitoring for these inputs is set to 300 ms.

If necessary, this monitoring time can be set to different values with an OPTION flag, or deactivated entirely.

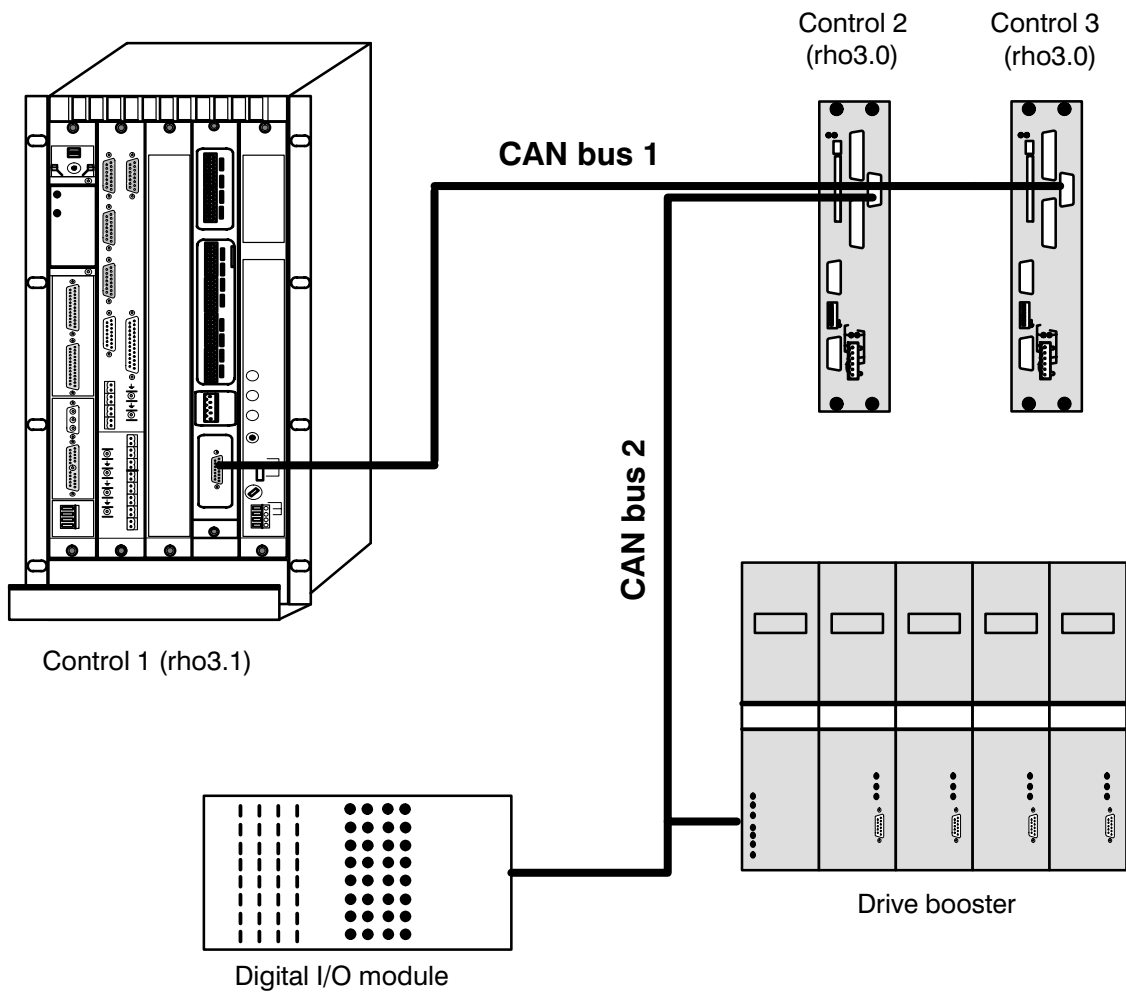
When the controls start up, time monitoring initially remains inactive. It is not activated until all controls participating in the data transmission have responded. If a control has not responded 60 sec. after start-up of the fastest control, transmission is aborted with the error message

**no transm. CANx Iy**      ( $x$  = number of the CAN bus,  
    $y$  = number of the input block)

The identifiers must be set so as to ensure that 2 controls connected to the same bus do not transmit via the same identifier.

**11.1.3 Structure**

(Example 2)



**Note:**

The CAN bus is suitable for coupling rho3.1 and/or rho3.0 controls in any configuration. All controls connected to the bus have equal access rights.

## Identifier (example 2):

### CAN bus 1 – data transmission between rho3 controls

Control 1	=>	Control 2:	471
		Control 3:	471
Control 2	=>	Control 1:	472
		Control 3:	472
Control 3	=>	Control 1:	473
		Control 2:	473

### CAN bus 2 – control 2 is master

Control 2	=>	Drive booster	
		SYNC telegram:	100
		Set value telegrams:	210,220,230,240
		Actual value telegrams:	610,620,630,640

Control 2	=>	Digital I/O module:	474
Digital I/O module:	=>	Control 2:	544

## 12 Memory expansion with SRAM board

### 12.1 General

In its maximum configuration, the rho3.0 currently offers a 512 kByte user RAM. If this memory range is too small, the memory can be expanded by inserting an SRAM memory board into the existing slot for PCMCIA boards. The RAM user memory (512 kBytes) is no longer available in this case. Currently, SRAM boards with 1 MB, 2 MB or 4 MB are commercially available.

### 12.2 Formatting the SRAM board

Only suitably formatted SRAM boards may be inserted into the rho3.0. At the moment, the SRAM boards are formatted by BOSCH. If the rho3.0 is started with an unformatted SRAM board, the control will not start up. In this case, no messages will be output on the PHG.

**Attention:** An EPROM backup may possibly be initiated, i.e. the user memory is deleted.

### 12.3 Inserting the SRAM board into the rho3.0

- ★ Insert board.
- ★ Start up the control with EPROM backup (press 'ALT', 'MODE', 'DELETE' and 'DEAD MAN' key at the PHG simultaneously).

==> User memory is completely created on the SRAM board. The size of the available memory is not changed. All existing user programs will be deleted!

### 12.4 Removing the SRAM board

After activation, the SRAM board is treated like a fixed component. Therefore, an active SRAM board must not be removed. Temporary removal of the SRAM board (e.g. in order to backup the data on the board or to use the slot for loading a new operating system) is permitted.



## 12.5 Restrictions

The SRAM board is treated like a fixed component of the control. Therefore, when the SRAM board has been activated, operation is subject to insertion of the board. During access, no check is carried out whether the SRAM board is still available, or if the desired access is permitted.

The following conditions may cause undefined error messages (SYSTEM errors):

- ★ Access to SRAM board if it is not inserted.
- ★ Write access to SRAM board if write protection is active.

## 12.6 Data protection

When the control is switched on, the SRAM board receives its power supply from the rho3.0, i.e. the buffer battery installed on the board is not discharged. When the control is switched off, the board is only buffered by the battery included on the board.

**The battery voltage is not monitored by the rho3.0.** Therefore, it is the responsibility of the user to replace the buffer battery regularly as specified by the board supplier.

## 13 PLC Interface expansion

### 13.1 General

The CL400 PLC is capable of addressing 256 bytes as inputs and 256 bytes as outputs. So far, only the reduced CL300 interface was supported by the CL400–PLC coupling, because the MTB inputs and outputs were not needed.

In connection with PHG 2000 applications, however, the MTB–I/O’s are increasingly used as a variable address range outside the rho standard signals.

### 13.2 Expanded PLC interface for rho3.0 withdrawable module version

For transmitting the complete rho3 system signals including the MTB signals, the **Expanded PLC interface** is necessary; i.e. 88 bytes of signal data are transmitted from/to rho3.0. The following figure shows the grouping of the signals and their flag subdivision in the PLC.

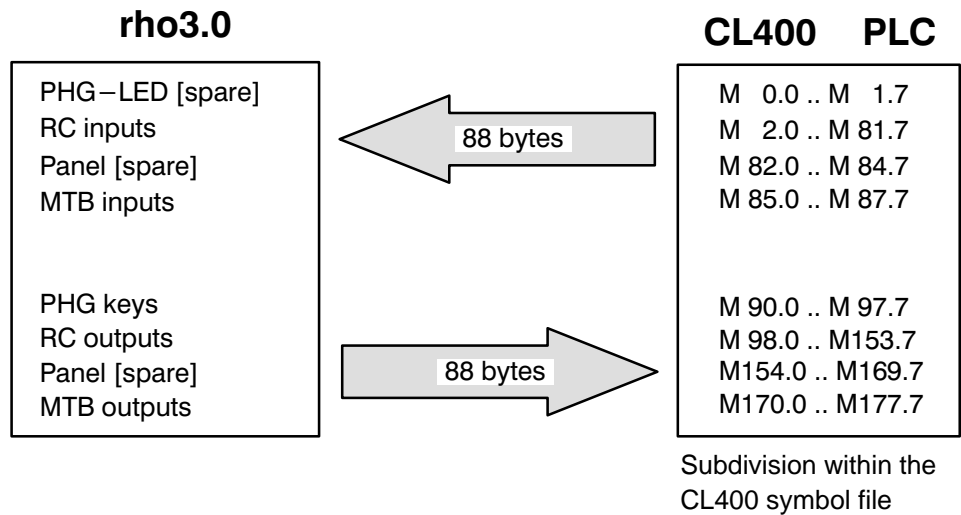


Fig. 7 – 32 Expanded PLC interface for rho3.0



**For saving flags, it is also possible to program auxiliary registers (cf. Chapter Interfacing with the PLC in manual 'rho3.0 Required Connections and Project Development Information' No. 1070 072149).**

## 13.2.1 PLC programming

In the ROPS3 subdirectory R3\_4.C00 you will find finished CL400 standard modules for the normal PLC interface. The standard modules for the expanded PLC interface are located in the ROPS3 subdirectory **R3\_6.C00**.

The **READPLC.TXT** file in the ROPS3 directory contains a list of the modules required for each configuration.

The **KOMFIFO** functional module controls communication between the PLC and the rho3.0. Enter K88 (expanded interface) in parameter P5, number of data bytes. Otherwise, the parameters are the same as for the CL300 interface.

The **DBLOAD\_6** program module copies the signal data from the data module for rho3.0 I/O's to the interface range specified by the user. The module now has four parameters. Their meaning is described in the following example.

The standard PLC module **R3\_7A** can be used for all configurations.

The program in the **R30\_OB1** organization module, for example, has the following structure:

```
CM  -RHO3DAT           ;Data module for rho3.0 I/O's

CM  -DBLOAD_6,4       ;DBLOAD_6 provides the I/O data in the
                       ;previously called data module (RHO3DAT)
P0  W  K15AH          ;Index address for PLC input array as of M 90
P1  W  K100H          ;Index address for PLC output array as of M 0
P2  W  K88D           ;Length of the data received
P3  W  K88D           ;Length of the data transmitted

CM  -KOMFIFO,7        ;Communication module for rho3.0
P0  W  K1              ;Start input
P1  W  K0              ;Central use
P2  W  K0              ;Module address of rho3.0
P3  W  K0              ;Busmaster FIFO no. (under preparation)
P4  W  K48D           ;FIFO depth (under preparation)
P5  W  K88D           ;Number of interface data bytes (expanded PLC
                       ;interface)
P6  W  D412           ;Status and acknowledgement word

CM  -R3_7A            ;Standard PLC module

CM  -ZEITPB0          ;Program module for cycle time adjustment
```

### 13.2.2 Setting the rho3 machine parameters

For defining the **Expanded PLC interface** on the rho3.0, the following parameters must be set.

#### 13.2.2.1 P20 I/O module configuration

I/O-HW-CONFIG. = 19

**Expanded PLC interface** for withdrawable rho3.0 module.

#### 13.2.2.2 P21 address range for PLC bit coupling

I.START : 0

Start address of the RC outputs in the PLC input range

I.END : 87

End address of the RC outputs in the PLC input range

O.START : 0

Start address of the RC inputs in the PLC output range

O.END: 87

End address of the RC inputs in the PLC output range

No value > 0 may be selected for the start addresses, because this would mean an additional shift of the signal data. Furthermore, the number of the data to be transmitted results from the difference between the end address and the start address.

### 13.3 Expanded PLC interface for rho3.0 (bit coupling)

For using the **Expanded PLC interface** with a bit coupler, you only have to change the configuration in the rho3 machine parameters and make adjustments in the PLC program. The division of the signals within the CL400 symbol file can be structured as follows, for example.

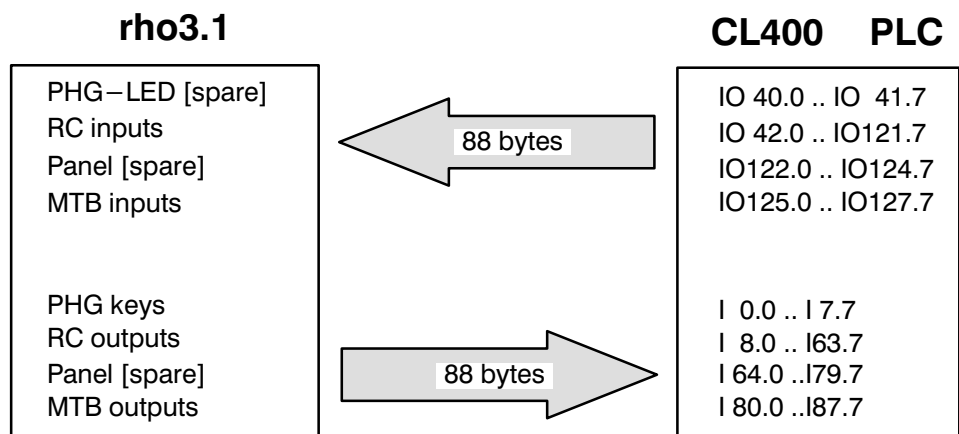


Fig. 7 – 33 Expanded PLC interface for rho3.1 (bit coupler in I/O array)

## 13.3.1 PLC programming

The **A\_LOAD\_6** program module copies the signal data from the flag range to the rho3.1 inputs. Since the output field of the CL400 can be directly addressed only up to O63.7, the data is copied to the interface output field. Since the interface output field changes the image of the CL400, the **A\_LOAD\_6** module **must be called up immediately before the program end** (cf. programming example).

The module has three parameters. The rho3 outputs are directly in the input field of the CL400 and can also be addressed by symbols in the actual program. The meaning of the parameters is explained in the following example.

The **R3\_7A** standard PLC module is used for all configurations.

The program in the **R31OB1\_6** organization module has the following structure, for example:

```
CM  -R3_7A           ; rho3 standard PLC module

CM  -ZEITPB0        ;Program module for cycle time adjustment

CM  -A_LOAD_6,3     ;Provides the output data for rho3
P0  W  K2C8H        ;Index address of PLC interface output field
                        ;as of IO 40
P1  W  K100H        ;Index address of flags as of M 0
P2  W  K88D         ;Length of data to be transmitted
```

The output addresses O0.0 to O39.7 were reserved for (external) digital outputs. In the standard PLC program, the digital outputs are programmed between O0.0 and O4.7.

The digital inputs can be used as of I88.0. In the standard PLC program, the digital inputs between I88.0 and I95.7 are assigned as for the PIC250 or SoftPic programs.

## 13.3.2 Setting the rho3 machine parameters

For defining the **Expanded PLC interface** on the rho3.1, the following parameters must be set.

**13.3.2.1 P20 I/O module configuration**

I/O–HW–CONFIG. = 2

**Expanded PLC interface** for rho3.1 bit coupling

**13.3.2.2 P21 address range for PLC bit coupling**

I.START :	<b>0</b>	Start address of the RC outputs in the PLC input range
I.END :	<b>87</b>	End address of the RC outputs in the PLC input range
O.START :	<b>40</b>	Start address of the RC inputs in the PLC output range
O.END:	<b>127</b>	End address of the RC inputs in the PLC output range

A value > 0 may be selected for the start addresses. This will mean a shift of the signal data. The number of data to be transmitted results from the difference between the end address and the start address.

**13.4 Expanded PLC interface for rho3.1 bit coupling in the extended I/O field**

Analogous to the standard bit coupler, the bit coupler can also be operated in the extended I/O field (NC–PLC(BZ)–coupler no. 077085) with the **Expanded PLC interface**. The figure shows the assignment of this bit coupling option.

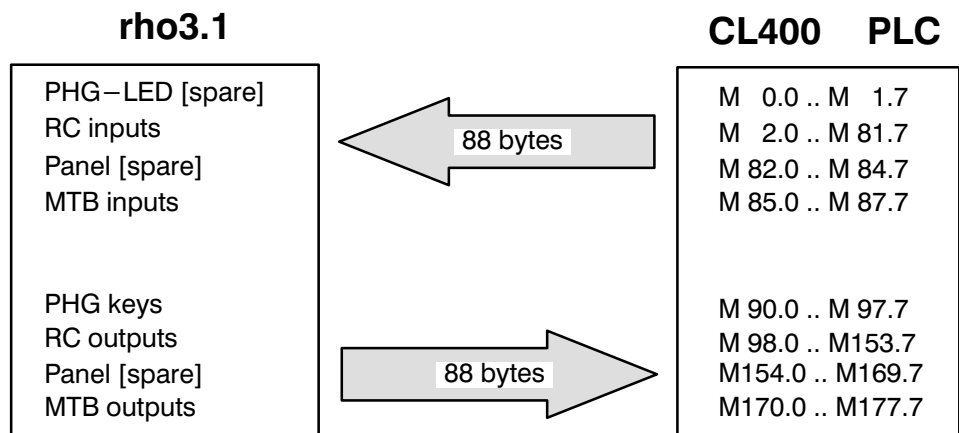


Fig. 7 – 34 Expanded PLC interface for rho3.1 (bit coupler in the EI/EO field)

## 13.4.1 PLC programming

The PLC program communicates with the rho3 bit coupler addressed in the extended I/O field.

The **EZAZLD\_6** program modules copies the signal data from the flag range to the extended output field EO and reads from the extended input field EI.

The module uses four parameters. The meaning of the parameters is explained in the following example.

The **R3\_7A** standard PLC module is used for all configurations.

The program in the **R31OB1\_6** organization module has the following structure, for example:

```
CM  -R3_7A           ; rho3 standard PLC module

CM  -ZEITPB0        ;Program module for cycle time adjustment

CM  -EZAZLD_6,4     ;Provides the input and output data of rho
                    ;in the extended I/O field (EI/EO) of the PLC
P0   W   K15AH      ;Index address of PLC extended input field
                    ;as of M 90
P1   W   K100H     ;Index address of the PLC extended output field
                    ;as of M 0
P2   W   K88D      ;Length of data to be received
P3   W   K88D      ;Length of data to be transmitted
```

## 13.4.2 Setting the rho3 machine parameters

### 13.4.2.1 P20 I/O module configuration

I/O-HW-CONFIG. = 2                      rho3.1 bit coupling with **Expanded PLC interface**

### 13.4.2.2 P21 address range for PLC bit coupling

I.START :	2	Start address of the RC outputs in the PLC input range
I.END :	89	End address of the RC outputs in the PLC input range
O.START :	2	Start address of the RC inputs in the PLC output range
O.END:	89	End address of the RC inputs in the PLC output range

A value > 0 must be selected for the start addresses. This will mean a shift of the signal data. The number of data to be transmitted results from the difference between the end address and the start address.

### 13.5 Overview of the programs for the Expanded PLC interface

The following modules are used for the **Expanded PLC interface**:

<b>R3_xx.PCO</b>	Standard PLC module
<b>OB2.PCO</b>	System settings
<b>ZEITPB0.PCO</b>	Program module for cycle time adjustment

**xx** denotes the version, e.g. 7A

#### 13.5.1 Programs for withdrawable rho3.0 module

<b>DBLOAD_6.PCO</b>	Load data module for rho3.0 coupling
<b>KOMFIFO.PCO</b>	Functional module for rho3.0 coupling
<b>R30_xx_6.PCA</b>	Program to be loaded ( <b>Expanded PLC interface</b> )
<b>R30_xx_6.SCS</b>	rho3.0 symbol file ( <b>Expanded PLC interface</b> )
<b>R30OB1_6.PCO</b>	OB1 organization module for <b>Expanded PLC interface</b>

#### 13.5.2 Programs for rho3.1 bit coupler

<b>A_LOAD_6.PCO</b>	Coupling for bit coupler in I/O field
<b>R31_xx_6.PCA</b>	Program to be loaded ( <b>Expanded PLC interface</b> )
<b>R31_xx_6.SCS</b>	rho3.1 symbol file ( <b>Expanded PLC interface</b> )
<b>R31_OB5.PCO</b>	Start-up module
<b>R31OB1_6.PCO</b>	OB1 organization module for bit coupler with <b>Expanded PLC interface</b>

#### 13.5.3 Programs for rho3.1 bit coupler in the extended I/O field

<b>EZAZLD_6.PCO</b>	Coupling for EI/EO extended I/O field
<b>R31Zxx_6.PCA</b>	Program to be loaded for the <b>Expanded PLC interface</b> with bit coupler in the extended I/O field
<b>R31Zxx_6.SCS</b>	rho3.1 symbol file for the <b>Expanded PLC interface</b> with bit coupler in the extended I/O field
<b>R31_OB5.PCO</b>	Start-up module
<b>R31OB1Z6.PCO</b>	OB1 organization module for bit coupler in the extended I/O field with <b>Expanded PLC interface</b> .



Your notes:

## 14 Expansion of the BAPS2 function CONDITION

In all earlier operating system versions including TO06J it is not possible to use IRDATA programs including the BAPS function **CONDITION <system signal>** in another language version without changing the QLL program and compiling the program again.

Example:

**ZUSTAND ( 'E88.0' )**

must be

**CONDITION ( 'I88.0' )**

in the English version !

As of operating system version TO07F, the identification letter of the **German** and the **English** language version is admissible for this function in addition to the **national language**.

In any case, the national language has priority. If the identification letter does not correspond to the respective national language, the system checks whether the letter fits the German or English language version.

As of version TO07F, BAPS programs compiled in German or English can be run in all other language versions as far as the BAPS function **CONDITION (<system signal>)** is concerned!

Example:

**ZUSTAND ( 'E88.0' ) = ZUSTAND ( 'I88.0' )**

**CONDITION ( 'I88.0' ) = CONDITION ( 'E88.0' )**

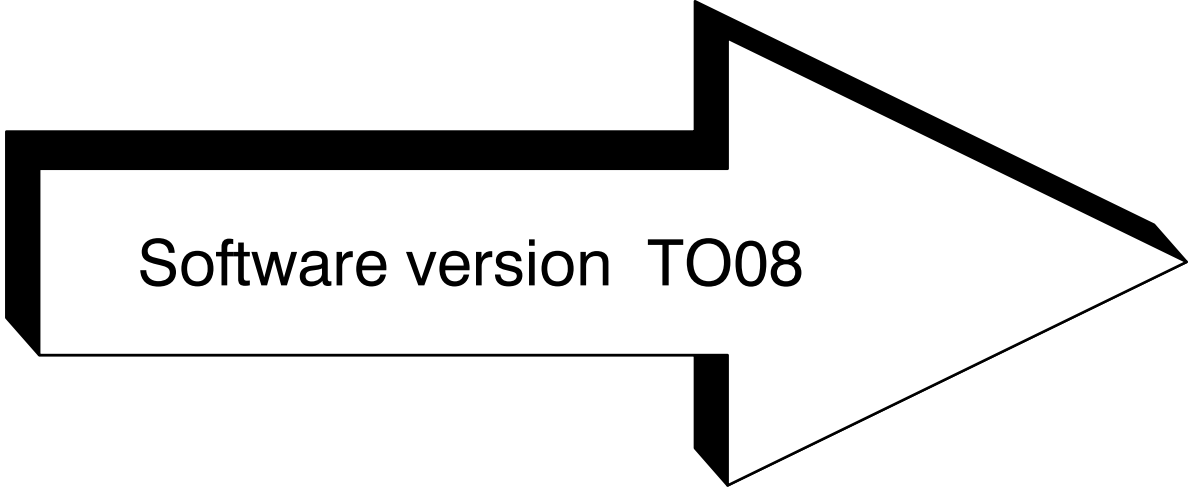
---

# Expansion of the BAPS2 function CONDITION

---



Your notes:

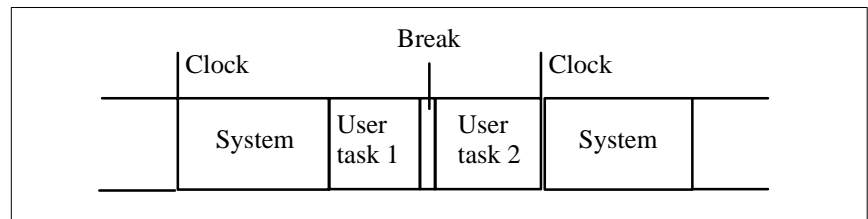


Software version TO08



# 1 BAPS FUNCTION "BREAK"

Normally, a BAPS program occupies the CPU until no more blocks have to be prepared (11 blocks, or the number of blocks specified with the BLOCK SEARCH special function have been prepared) or until the clock cycle requests the CPU for position control.



*Fig. 8 – 1 CPU occupation with Break command*

Using the BAPS function **BREAK** the user can force an interruption of the active user process at any point of an active user process, thus making the CPU available to other user and/or system processes with the same or a higher priority during the same clock time. The interrupted process is added to the end of the queue of all processes with the same priority. When all processes with equal priority waiting for CPU availability have been active once (Round Robin algorithm) the interrupted process becomes active again. Processes with a higher priority may become active several times before the interrupted process is activated again.

## 1.1 BAPS syntax

The BAPS function **BREAK** is realized as a BAPS standard subroutine without parameters.

Call in BAPS (German): **UNTERBRECHE**

Call in BAPS (English): **BREAK**

EXAMPLE: PROGRAM EXMP\_1

```
INPUT      : I=I1
OUTPUT     : O=O1

BEGIN
LOOP:
    IF I1=1
        THEN O1=1
        BREAK
        JUMP LOOP
PROGRAM_END
```

The process shown in the example releases the CPU prematurely whenever the loop has been processed. The remaining time until the next clock interrupt is made available to other processes.

## 1.2 Special features

Since the BAPS function **BREAK** is implemented as a BAPS standard subroutine, no BAPS language symbol has been reserved. Therefore, it should be ensured that the name **BREAK** is not used in other BAPS declarations (e.g. as a variable name or a name of another subroutine). Existing BAPS programs containing the BAPS name **BREAK** must be changed.

## 2 Multi–function I/O’s

### 2.1 General

The rho3.0 is equipped with **4 MF I/O’s** each. These I/O’s can be used in BAPS in conjunction with a special function (inputs only) with channel numbers 611–614 and as high–speed inputs/outputs with channel numbers 801–804.

For saving digital I/O modules, these MF I/O’s are also provided at the interface to the **PIC** or the **PLC**.

#### 2.1.1 General conditions

- Only **free** address ranges may be set at the interface.
- A whole byte is assigned, however, only the 4 lower–value bits will be used.
- In the **stand–alone** version, **free** addresses between **88–127** may also be selected.
- In the **withdrawable module version** only the addresses **up to 88** are available.
- If parameter **P36** (address) is **–1**, only BAPS will be effective. If **P36** is  $\neq -1$ , the conditions of the multi–function I/O’s will be overwritten by PIC/PLC.

### 2.2 Multi–function I/O’s via SoftPIC for stand–alone rho3.0

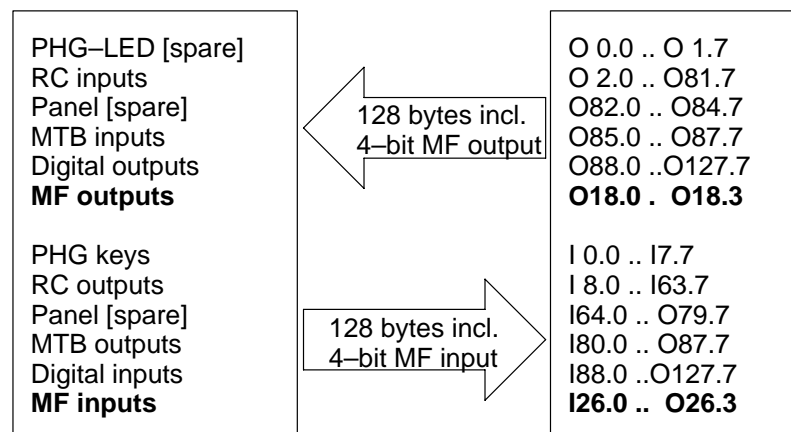


Fig. 8 – 2 Standard interface assignment with MF I/O’s for SoftPIC



## 2.3 Multi–function I/O's via PLC with withdrawable rho3.0 version

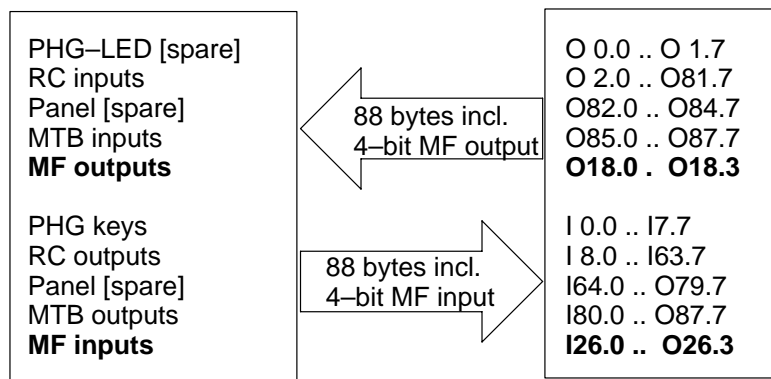


Fig. 8 – 3 Standard interface assignment with MF I/O's for withdrawable version

## 2.4 Machine parameters

The addresses for the multi–function I/O's are set in machine parameter 36.

Parameter 36: Address for **multi–function inputs**

<b>0</b>	<b>I26</b>	<b>Basic setting</b>
<b>1..87</b>	<b>I1 .. I87</b> e.g. I24,I26,I27	<b>Address setting</b> free addresses
<b>88..127</b>	<b>I28..127</b>	<b>only available in stand–alone version</b> (if not occupied)
<b>–1</b>	<b>–</b>	MF I/O's are <b>not active</b> –> only available in BAPS

Parameter 36: Address for **multi–function outputs**

<b>0</b>	<b>O18</b>	<b>Basic setting</b>
<b>1..87</b>	<b>O1 .. O87</b> O18–20, O23, O32, O34, O35, O42, O43–46, O48, O55–57, O73–75	<b>Address setting</b> free addresses
<b>88..127</b>	<b>O88 .. O127</b>	<b>only available in stand–alone version</b> (if not occupied)
<b>–1</b>	<b>–</b>	MF I/O's are <b>not active</b> –> only available in BAPS

## 3 Initializing SRAM boards

### 3.1 General

In its maximum configuration, the rho3.0 currently offers a 512 kByte internal user RAM. This memory range can be expanded by inserting an SRAM memory board into the PCMCIA slot of the rho3.0.

The option "Memory expansion for rho3.0 on SRAM board" is available as of operating system version **TO08E**.

#### 3.1.1 General conditions

- The SRAM board must have at least 1 MByte RAM.
- The SRAM board is considered as fixed component of the control. Therefore, it must always be inserted once it has been initialized (in operation). Removing an initialized SRAM board while the control is running or activating write protection will cause the rho3 to output **System error 39**.
- If the control starts up with the SRAM board initialized but not inserted, or active write protection, **System error 300** will be output.
- For removing an initialized SRAM board, the memory size must be reset to its original value (max. 512 k). This can be achieved by loading the old machine parameters or by an EEPROM backup.
- An SRAM board can only be initialized for the rho3.0. For the rho3.1, the error message "Option not active" (cf. section 3.2.2) will be displayed if MODE 9.8 is selected.
- When a new operating system version has been loaded (from flash board), the SRAM board must be re-initialized. Initializing the board will delete all data on the SRAM board.
- The use of the SRAM board is interlocked with an option bit.

## 3.2 Formatting and initializing the SRAM board

During initialization, the SRAM board is formatted, i.e. the user programs must be saved beforehand.

### 3.2.1 Procedure for initializing an SRAM board

- Start up the control without SRAM board.
- Insert SRAM board.
- Produce EMERGENCY–STOP condition.
- Stop all active user processes (including permanent processes), if any.
- Select **MODE 9.8** on the PHG.

PHG display

```
INIT SRAM
SRAM–Card
Initialisation
execute with:ENTER
```

Since any data that may be available on the board or in the internal user memory of the rho3.0 will be destroyed by formatting, the following verification is displayed when ENTER has been pressed:

```
INIT SRAM
SRAM–Card
Initialisation
sure? (yes): ENTER
```

Initialization is not performed unless this verification is confirmed. If the SRAM board shall not be initialized, press 'SHIFT <' to exit MODE 9.8 at this point. The user memory size is automatically set to the memory available on the SRAM board. Then the rho3.0 is automatically restarted. When the new start–up is complete, the SRAM board has been completely initialized in the control.

## 3.2.2 Error messages

PHG display:

**Option not active  
any key --> continue**

Cause:

- Function was selected on rho3.1 control.
- Option was not enabled in the rho3.0.

PHG display:

**EMERGENCY STOP input  
any key --> continue**

Cause: Control is not in EMERGENCY STOP condition.

PHG display:

**Prog. is active  
any key --> continue**

Cause: At least 1 (permanent) process is still active.

PHG display:

**No SRAM-Card  
any key --> continue**

Cause: No, or an incorrect, PCMCIA board was inserted (e.g. FLASH board).

PHG display:

**Write protection  
any key --> continue**

Cause: Write protection of the SRAM board is active.

PHG display:

**SRAM – Card too small  
any key --> continue**

Cause: An SRAM board with less than 1 MByte memory was inserted.

PHG display:

**PCMCIA – Card defective  
any key --> continue**

Cause: Memory chip on the SRAM board is defective.

PHG display: (as of version TO08E)

**System error 39  
Error SRAM – Card  
any key --> restart**

Cause:

- SRAM board was removed while the control was running.
- Write protection was activated while the control was running.

PHG display:

<b>XX</b>
<b>CHECKSUM</b>
<b>XX</b>
<b>SYSTEM ERROR 300</b>

Cause:

- SRAM board was removed before the control started up.
- Write protection of the SRAM board is active.

Note:

If **System error 300** is displayed, press the 'ALT', 'MODE', 'DELETE' and 'DEAD MAN' (permission) keys at the PHG simultaneously to initiate an EPROM backup directly (i.e. without toggling the rho3 off/on). The control is started up again once in order to eliminate the cause of the error (e.g. reload machine parameters). If the error is not eliminated, **System error 300** will be displayed again when the control starts up again.

### 3.3 Data protection

When the control is switched on, the SRAM board receives its power supply from the rho3.0, i.e. the buffer battery installed on the board is not discharged. When the control is switched off, the board is only buffered by the battery included on the board. **The battery voltage is not monitored by the rho3.0.** Therefore, it is the responsibility of the user to replace the buffer battery regularly as specified by the board supplier.

## 3.4 Replacing the SRAM board

When the SRAM board is replaced, the entire user memory is deleted. Therefore, all necessary user programs must be saved before the board is replaced.

For replacing an SRAM board installed in a rho3.0, first the procedure described below must be observed:

- **Save all user programs** available in the rho (machine parameters and PIC programs are retained).
- Switch control off.
- Remove SRAM board from PCMCIA slot.
- Switch control on.  
The control starts up until the RAM test.  
Then start-up is aborted with system error 300.
- Insert new SRAM board while the control is **still switched on**.
- Press the

### **ALT + MODE + DEL + Permission keys**

simultaneously to perform an EPROM backup.  
The control starts up again.  
The new SRAM board is initialized.  
The user memory is empty.

- Reload user programs that have previously been saved.

## 4 Belt synchronization with endless belt

### 4.1 General

Some applications require an endless axis moving permanently synchronously to another (uncontrolled) axis (belt). For this purpose, the master axis (belt) must be cyclically reset in order to avoid an internal counter overflow.

### 4.2 Function

The BAPS **SYNC** *belt name* command resets the belt counter. So far, this command could only be programmed **outside** the synchronous movement.

Using this option, the **SYNC** *belt name* command can also be programmed **within** the synchronization condition. The belt counter is reset without disturbing the synchronous movement.

### 4.3 Restrictions

For resetting the belt counter within the synchronous belt movement, the following restrictions must be observed:

- Programming of **absolute positions** on the belt, analogous to the "Belt synchronization" option, is **not possible** because the belt counter value is no longer available as an absolute value. It is not possible to teach in positions relative to the belt.
- The belt-parallel axis must be declared as an **endless axis**.
- Within the synchronous belt movement, **only MOVE REL commands** make sense for the belt-parallel axis.
- Within the synchronous movement, the belt value **must not be reset conditionally depending on an input signal** (e.g. SYNC BN1,I1=1).



## 4.4 Program examples

### 4.4.1 Example 1

```
;;CONTROL=rho3
;;KINEMATICS:(1=ROBI1)

;;ROBI1.JC_NAMES=  A_1,BN1
;;ROBI1.WC_NAMES=  K_1,BN1

PROGRAM BS_VS0

INPUT:          1=  E1
ROBI1.BELT:    501= BN1
INT:           MOD_VAL

;;KINEMATICS=ROBI1
;;INT=LINEAR

BEGIN

AFACTOR=1;DFACTOR=1;VFACTOR=1;V=100
MOD_VAL = 1000

SYNC BN1,I1=1
MOVE LINEAR TO POS
SYNCHRON ROBI1 BN1

LOOP:
SYNC BN1 >= MOD_VAL
WAIT1
IF I1=1 THEN JUMP LOOP

SYNCHRON_END BN1

PROGRAM_END
```

When the programmed limit value (in this example MOD\_VAL = 1000) is exceeded, the command **SYNC BN1 >= 1000** is used to reduce the current belt counter value by this limit value. The belt-parallel axis (A\_1) continuously follows the belt even when the belt counter is reset.

The program is stopped in the SYNC block until the programmed SYNC condition is met. The program does not continue until the SYNC command has been executed.

#### 4.4.2 Example 2

```
.  
. .  
. .  
SYNC BN1,I1=1  
MOVE LINEAR TO POS  
SYNCHRON ROBI1 BN1  
  
LOOP:  
IF @POS.BN1 <MOD_VAL THEN JUMP NO_SYNC  
  
SYNC BN1 >= MOD_VAL  
NO_SYNC:  
MOVE REL LINEAR (10,0)  
MOVE REL LINEAR (-10,0)  
IF I1=1 THEN JUMP LOOP  
SYNCHRON_END BN1  
  
PROGRAM_END
```

In example 2, the command **SYNC BN1 >= 1000** only becomes active when the programmed limit value for the belt counter has been exceeded. Thus, it is possible to execute more BAPS command within the loop. In this example, the belt- parallel axis moves forward and backward relative to the belt.

Your notes:

## 5 Remote use of the rho3.0

### 5.1 General

As of operating system version TO08E, remote operation of the rho3.0 in an expansion unit coupled with PROFIBUS/DP is supported. The KOMFIFO module (coupling between rho3.0–PLC) was expanded accordingly.

On the PLC side, this feature is supported by the modules:

Busmaster **BM DP12** in the higher–level PLC

Remote module **RM4 DP12** in the remote expansion unit

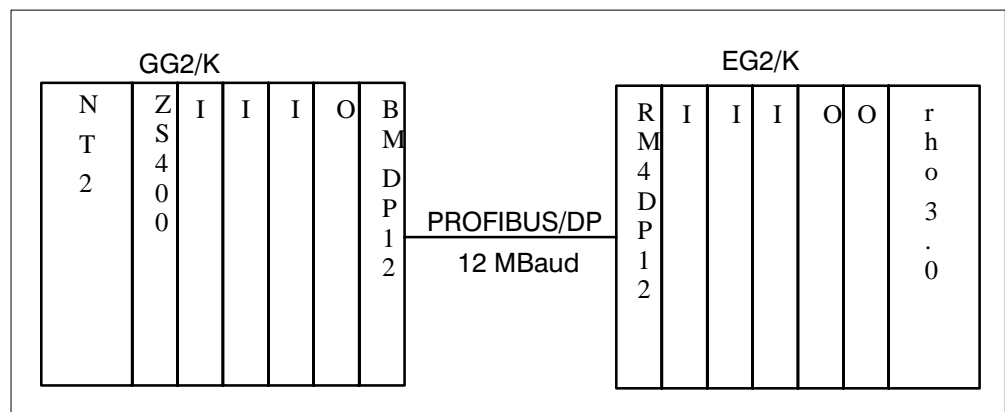


Fig. 8 – 4 Remote operation of the rho3.0 (PLC coupling with PROFIBUS/DP)

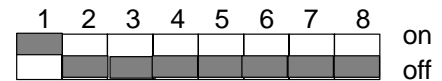
#### Busmaster (BM DP12) setting:

		1	2	3	4	5	6	7	8	
Station address (S3) = 1										on off
Coupling field address (S4) e.g. EI/EO 4										on off
Baudrate (S5) + Transmission format at X71 (Profibus) = 12 MBaud										on off
Baudrate (S6) + Transmission format at X31 (PG interface)										on off

The busmaster can only be addressed in the extended I/O field (EI/EO). As of the selected start address, it occupies a 6 byte address range in the extended input and the extended output field.

## Remote module (RM4 DP12) setting:

Bus station address (S1)  
(must be identical with the  
setting in the bus master file)



## 5.2 PLC program examples for remote use of the rho3.0

The directory **\ROPS3\DEZENTRL.C00** contains two PLC program examples (normal interface with 46 bytes and expanded interface with 88 bytes).

When using the expanded interface, transmission must be by blocks, because max. 48 bytes can be transferred with each transmission.

Another example in the directory **\ROPS3\ZEN\_DEZ.C00** describes mixed operation with one central and one remote rho3.0.

## 6 Start-up behavior of multiple rho3.0's (in PLC)

Data transmission between the rho3.0 and the central unit cannot be initiated unless start-up of both controls has been completed. The rho3.0 (as well as other intelligent peripheral bus modules, such as CC10.3, type 1 osa or the busmaster BM DP) uses the I/OINIT signal to inform the ZS400 that the module is in the start-up phase. When the I/OINIT signal is active, the PLC waits until the I/OINIT's of all modules become "0". Then the PLC starts up, and communication is initiated. For rho version TO071 this means a PLC timeout of the rho that was first started up because it first does not receive a response from the PLC.

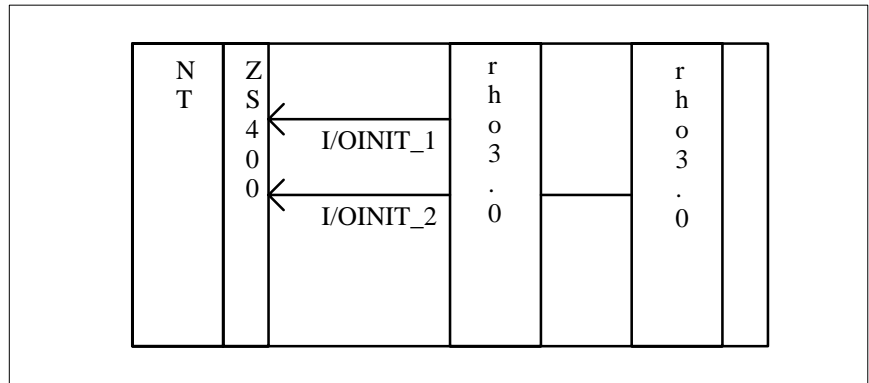


Fig. 8 - 5 Start-up signal rho3.0-PLC

As of rho version TO07J, the PLC timeout response during start-up is delayed by 80s. After the first transmission (KOMFIFO), the old timeout value becomes active again.

If the rho3.0 is operated remotely, the PLC timeout still responds in the version TO08E in the case of different start-up behaviors.

As of ZS400 operating system version V2.3, the I/OINIT signal can be deactivated in OB2.

If the I/OINIT signal is not active, the PLC starts up at once. The rho's are immediately served as soon as they have started up (KOMFIFO).

For enabling the rho-specific signals, bit 10 (RC ready) can be used by P6 (KOMFIFO) (also refer to 'rho3.0 Required Connections and Project Development Information' No. 1070 073 149).

---

# Start-up behavior of multiple rho3.0's (in PLC)

---



Your notes:

## **A Annex**

### **A.1 Abbreviations**

C:	Drive name in this case drive C (fixed disk drive)
EI	Extended input field
EO	Extended output field
e.s.d.	electrostatic discharge Abbreviation for all terms referring to electrostatic discharge, e.g. e.s.d. protection, e.s.d. compo- nent.
e.s.d. component	Component subject to electrostatic discharge
PE	Protective Earth



## A.2 Safety instructions

### A.2.1 Dansk

#### Sikkerhedshenvisningerne i denne brugsanvisning



Disse symboler anvendes i den foreliggende brugsanvisning i følgende tilfælde:



**FORSIGTIG**

Dette symbol benyttes, hvis der skal advares mod **farlig elektrisk spænding**. Hvis advarslen ikke følges nøjagtigt eller ignoreres kan det medføre **personskader**.



**FORSIGTIG**

Dette symbol benyttes, hvis en unøjagtig eller manglende overholdelse af anvisningerne kan medføre beskadigelser af **personer**.



**VIGTIGT**

Dette symbol benyttes, hvis en unøjagtig eller manglende overholdelse af anvisningerne kan medføre beskadigelser af **apparater eller filer**.



Dette symbol benyttes for at gøre Dem opmærksom på noget særligt.



**FORSIGTIG**

0.1

Risiko for personer og ting!

Prøv hvert nyt program, inden De tager et anlæg i drift!



**VIGTIGT**

0.2

Risiko for modulet!

Modulet må ikke sættes i eller trækkes ud af stikket, når der er tændt for styringen! Modulet kan blive ødelagt. Der skal først slukkes for styringens netdelmodul, den eksterne spændingsforsyning og signalspændingen eller disse skal trækkes ud af stikket, inden modulet må sættes i eller trækkes ud af stikket!



**VIGTIGT**

0.3

Risiko for modulet!

Ved omgang med modulet skal alle forholdsregler til ESD-beskyttelse iagttages!

Undgå elektrostatiske udladninger!



**VIGTIGT**

6.1

Ved program-start skal det ubetinget sikres, at det korrekte emne-koordinatsystem er aktiveret. Hvis der benyttes programmer med forkerte emne-koordinatsystemer, kan der opstå uventede bevægelser.

Den samme effekt kan optræde, hvis der benyttes forskellige emne-koordinatsystemer til teaching og program-gennemarbejdning i forbindelse med et teachtet rumkoordinatpunkt.



**VIGTIGT**

7.1

Ansvar for utilsigtede reaktioner ved en evt. forkert fil TASTEN.BNR bæres af maskinfabrikanten. Ændringer af standardbelægningen må kun ske efter forudgående aftale med BOSCH.

#### Sikkerhedshenvisninger på styrekomponenterne

På styrekomponenterne selv kan der være anbragt følgende advarsler og henvisninger, som skal gøre Dem opmærksom på bestemte ting:



Advarsel mod farlig elektrisk spænding!



Advarsel mod farer fra batterier!



Elektrostatisk udsatte komponenter!



Træk netstikket ud, inden De åbner!



Bolt kun til tilslutning af jordledningen PE!



Kun til tilslutning af en afskærmningsledning!

## A.2.2 Deutsch

### Sicherheitshinweise in dieser Gebrauchsanweisung



Diese Symbole werden in dieser Gebrauchsanweisung unter den folgenden Bedingungen verwendet.



#### VORSICHT

Dieses Symbol wird benutzt, wenn vor einer **gefährlichen elektrischen Spannung** gewarnt werden soll. Durch ungenaues Befolgen oder Nichtbefolgen dieser Anweisung kann es zu **Personenschäden** kommen.



#### VORSICHT

Dieses Symbol wird benutzt, wenn es durch ungenaues Befolgen oder Nichtbefolgen von Anweisungen zu **Personenschäden** kommen kann.



#### ACHTUNG

Dieses Symbol wird benutzt, wenn es durch ungenaues Befolgen oder Nichtbefolgen von Anweisungen zu **Beschädigungen von Geräten oder Dateien** kommen kann.



Dieses Symbol wird benutzt, wenn Sie auf etwas Besonderes aufmerksam gemacht werden sollen.



#### VORSICHT

0.1

Gefahr für Personen und Sachen!  
Testen Sie jedes neue Programm bevor Sie eine Anlage in Betrieb nehmen!



#### ACHTUNG

0.2

Gefahr für die Baugruppe!  
Baugruppe nicht bei eingeschalteter Steuerung stecken oder ziehen! Baugruppe kann zerstört werden. Zuerst Netzteilbaugruppe der Steuerung, externe Spannungsversorgung und Signalspannung ausschalten oder abziehen und erst dann Baugruppe stecken oder ziehen!



#### ACHTUNG

0.3

Gefahr für die Baugruppe!  
Beim Umgang mit der Baugruppe müssen alle Vorkehrungen zum ESD-Schutz eingehalten werden! Elektrostatische Entladungen vermeiden!



#### ACHTUNG

6.1

Bei Programm-Start ist unbedingt sicherzustellen, daß das korrekte Werkstück-Koordinatensystem aktiviert ist. Werden Programme mit falschen Werkstück-Koordinatensystemen abgefahren, so kann es zu unerwarteten Bewegungen kommen. Der gleiche Effekt kann auftreten, wenn bei einem geteachten Raumkoordinatenpunkt unterschiedliche Werkstück-Koordinatensysteme zum Teachen und zur Programm-Abarbeitung verwendet werden.



#### ACHTUNG

7.1

Die Verantwortung für ungewollte Reaktionen bei evtl. falscher Datei TASTEN.BNR liegt beim Maschinenhersteller. Änderungen gegenüber der Standardtastenbelegung sollten nur in Absprache mit Fa. BOSCH erfolgen.

**Sicherheitshinweise an den Steuerungskomponenten**

An den Steuerungskomponenten selbst können folgende Warnungen und Hinweise angebracht sein, die Sie auf bestimmte Dinge aufmerksam machen sollen:



Warnung vor gefährlicher elektrischer Spannung!



Warnung vor Gefahren durch Batterien!



Elektrostatisch gefährdete Bauelemente!



Vor dem Öffnen Netzstecker ziehen!



Bolzen nur für Anschluß des Schutzleiters PE!



Nur für Anschluß eines Schirmleiters!

### A.2.3 Ελληνικά

Υποδείξεις ασφαλείας στις παρούσες οδηγίες χρήσεως



Τα σύμβολα αυτά στις παρούσες οδηγίες χρήσεως χρησιμοποιούνται υπό τους ακόλουθους όρους:



#### ΚΙΝΔΥΝΟΣ

Αυτό το σύμβολο χρησιμοποιείται για να σας προειδοποιήσει από επικίνδυνη ηλεκτρική τάση. Αν δεν τηρούνται με ακρίβεια ή δεν τηρούνται καθόλου οι οδηγίες μπορεί να προκληθούν σωματικές βλάβες.



#### ΚΙΝΔΥΝΟΣ

Το σύμβολο αυτό χρησιμοποιείται, όταν μπορεί να προκληθούν σωματικές βλάβες, αν δεν τηρούνται με ακρίβεια ή δεν τηρούνται καθόλου οδηγίες.



#### ΠΡΟΣΟΧΗ

Το σύμβολο αυτό χρησιμοποιείται, όταν μπορεί να προκληθούν ζημιές σε συσκευές ή σε αρχεία, αν δεν τηρούνται με ακρίβεια ή δεν τηρούνται καθόλου οδηγίες.



Το σύμβολο αυτό χρησιμοποιείται, όταν θα πρέπει να επιστηθεί η προσοχή σας σε κάτι το σημαντικό.



#### ΚΙΝΔΥΝΟΣ

0.1

Κίνδυνος για πρόσωπα και αντικείμενα!

Δοκιμάστε κάθε καινούριο πρόγραμμα πριν θέσετε μια εγκατάσταση σε λειτουργία!



#### ΠΡΟΣΟΧΗ

0.2

Κίνδυνος για το στοιχείο κατασκευής!

Μην αφαιρείτε ή τοποθετείτε το στοιχείο κατασκευής σε κύκλωμα που είναι σε λειτουργία! Το στοιχείο κατασκευής μπορεί να καταστραφεί. Πρώτα αφαιρείτε ή αποσυνδέετε το στοιχείο κατασκευής της ρύθμισης του ηλεκτρικού κυκλώματος, κατόπιν την παροχή τάσης και την τάση σήματος και μετά τοποθετείτε ή αφαιρείτε το στοιχείο κατασκευής.



#### ΠΡΟΣΟΧΗ

0.3

Κίνδυνος για το στοιχείο κατασκευής!

Όταν έχετε στα χέρια σας το στοιχείο κατασκευής πρέπει να τηρείτε όλα τα μέτρα για την ηλεκτροστατική προστασία! Αποφεύγετε ηλεκτροστατικές εκφορτίσεις!



#### ΠΡΟΣΟΧΗ

6.1

Κατά την εκκίνηση του προγράμματος πρέπει οπωσδήποτε να εξασφαλισθεί, ότι είναι ενεργοποιημένο το σωστό σύστημα συντεταγμένων του προς κατεργασία τεμαχίου. Σε περίπτωση που ξεκινήσουν προγράμματα με λάθος σύστημα συντεταγμένων του προς κατεργασία τεμαχίου, μπορεί αυτό να έχει σαν αποτέλεσμα μη αναμενόμενες κινήσεις.

Η ίδιο φαινόμενο μπορεί να παρουσιασθεί και όταν χρησιμοποιούνται σε σημείο συντεταγμένων χώρου που υπάρχει στη μνήμη του μηχανήματος διαφορετικά συστήματα συντεταγμένων του προς κατεργασία τεμαχίου προς εκμάθηση και χρησιμοποιούνται προς επεξεργασία προγράμματος.

**ΠΡΟΣΟΧΗ****7.1**

Την ευθύνη για αθέλητες αντιδράσεις ενδεχομένως σε περίπτωση χρήσης λάθος αρχείου ΤΑΣΤΕΝ.BNP έχει ο κατασκευαστής του μηχανήματος. Αλλαγές σχετικά με την τυποποιημένη κατάληψη πλήκτρων θα πρέπει να γίνονται μόνον κατόπιν συνεννόησης με τον Οίκο ΒΟΣΧΗ.

Υποδείξεις ασφαλείας σε εξαρτήματα ρύθμισης και ελέγχου

Τα εξαρτήματα ρύθμισης και ελέγχου μπορεί να φέρουν τις ακόλουθες προειδοποιήσεις και υποδείξεις, που επιστούν την προσοχή σας σε ορισμένα πράγματα:



Προειδοποίηση σχετικά με επικίνδυνη τάση ηλεκτρικού ρεύματος!



Προειδοποίηση σχετικά με κινδύνους, που προέρχονται από μπαταρίες!



Στοιχεία κατασκευής, για τα οποία υπάρχει ηλεκτροστατικός κίνδυνος!



Πριν από το άνοιγμα βγάλτε το φισ από την πρίζα!



Πείροι μόνο για σύνδεση προστατευτικού αγωγού (γείωσης) PE!



Μόνο για σύνδεση θωρακισμένου αγωγού!

## A.2.4 English

### Safety instructions in this manual



These symbols are used throughout this manual subject to the following conditions.



**DANGER**

This symbol is used to warn of the presence of **dangerous electrical current**. Insufficient or lacking compliance with these instructions can result in **personal injury**.



**DANGER**

This symbol is used wherever an insufficient or lacking compliance with instructions can result in **personal injury**.



**CAUTION**

This symbol is used wherever an insufficient or lacking compliance with instructions can result in **damage to equipment or files**.



This symbol is used to inform the user of special features.



**DANGER**

0.1

Danger to persons and equipment!

New programs must be tested before a system is put into operation!



**CAUTION**

0.2

Danger to the module!

Do not insert or remove module when the control is switched on! This can destroy the module. Switch off or remove control power supply module, external power supply and signal voltage before inserting or removing the module!



**CAUTION**

0.3

Danger for the module!

When handling the module, follow all precautions for e.s.d. protection! Avoid electrostatic discharges!



**CAUTION**

6.1

At the time of program start, it is mandatory to ensure that the proper coordinate system corresponding to the workpiece being used is enabled. In the event that programs are started with incorrect workpiece coordinate systems, unexpected and undesirable movements may result.



**CAUTION**

7.1

The machine manufacturer bears the sole responsibility for unintended movements possibly caused by the selection of an incorrect version of the KEY.BNR file (keyboard assignment). It is recommended that any modification of the standard keyboard assignment be performed subject to prior consultation with BOSCH.

### Safety instructions on the control components

The following warnings and notices may be indicated on the control components themselves and have the following meaning:



Danger: High voltage!



Danger: Battery acid!



Electrostatically-sensitive components!



Disconnect at mains before opening!



Pin for connecting PE conductor only!



For screened conductor only!



## A.2.5 Español

### Indicaciones de seguridad en estas instrucciones de empleo



Estos símbolos se utilizan en estas instrucciones de empleo bajo las siguientes condiciones.



#### PRECAUCION

Este símbolo se utiliza para advertir de una **tensión eléctrica peligrosa**. La ejecución inexacta o la no ejecución de esta instrucción podrá provocar **daños a las personas**.



#### PRECAUCION

Este símbolo se utiliza cuando por una ejecución inexacta o la no ejecución de instrucciones se pueden llegar a producir **daños a las personas**.



#### ATENCION

Este símbolo se utiliza cuando por la ejecución inexacta o la no ejecución de instrucciones se pueden llegar a producir **daños en los aparatos o archivos**.



Este símbolo se utiliza cuando se le debe llamar la atención respecto a algo especial.



#### PRECAUCION

0.1

¡Peligro para personas y bienes materiales!  
¡Compruebe cada nuevo programa antes de poner en funcionamiento una instalación!



#### ATENCION

0.2

¡Peligro para el módulo!  
¡No enchufe ni extraiga el módulo cuando el control está conectado! Puede destruirse el módulo. ¡Desconecte o desenchufe primero el módulo de fuente de alimentación del control, la alimentación de tensión externa y la tensión de señalización y sólo después enchufe o extraiga el módulo!



#### ATENCION

0.3

¡Peligro para el módulo!  
¡Observe en la manipulación del módulo todas las precauciones en cuanto a la protección ESD! ¡Evite descargas estáticas!



#### ATENCION

6.1

Al arrancar el programa hay que prestar atención a que se encuentre activado el programa de coordenadas correcto para la pieza. En caso de activarse programas con un sistema de coordenadas de pieza errónea, pueden producirse movimientos inesperados. El mismo efecto puede producirse si se emplea en un punto de coordenadas espaciales de enseñanza, diferentes sistemas de coordenadas de piezas para la enseñanza y la ejecución del programa.



#### ATENCION

7.1

La responsabilidad de las reacciones imprevistas debidas un fichero erróneo «TASTEN.BNR». Las modificaciones que fuera necesario efectuar con respecto a la ocupación estándar del teclado, sólo deberán efectuarse previa consulta de BOSCH.

**Indicaciones de seguridad en los componentes de control**

En los componentes de control mismos pueden estar dispuestos las siguientes advertencias e indicaciones que le deben llamar la atención sobre determinados temas:



¡Advertencia ante tensión eléctrica peligrosa!



¡Advertencia ante riesgos por baterías!



¡Elementos constructivos con riesgos de descargas electrostáticas!



¡Antes de abrir, desenchufar el conector de la red!



¡Perno sólo para la conexión del conductor protector PE!



¡Sólo para la conexión de un conector blindado!

## A.2.6 Français

### Directives de sécurité relatives au présent mode d'emploi



Ces symboles sont utilisés dans les conditions suivantes:



**DANGER**

Ce symbole est utilisé lorsque l'on veut mettre en garde contre une **tension électrique dangereuse**. Risque de **dommage corporel** si les consignes données ne sont pas respectées ou lorsqu'elles sont mal respectées.



**DANGER**

Ce symbole est utilisé s'il y a un risque de **dommage corporel** si les consignes données ne sont pas respectées ou lorsqu'elles sont mal respectées.



**ATTENTION**

Ce symbole est utilisé s'il y a un risque de dommage matériel ou risque de destruction de fichier si les consignes données ne sont pas respectées ou lorsqu'elles sont mal respectées.



Ce symbole est utilisé lorsqu'il s'agit d'attirer votre attention sur un point particulier.



**DANGER**

0.1

Risque pour les personnes et le matériel !

Testez chaque nouveau programme avant de mettre une installation en service!



**ATTENTION**

0.2

Risque pour l'unité !

Ne branchez ou ne débranchez pas l'unité lorsque la commande est activée ! Risque de destruction de l'unité. Avant de brancher ou de débrancher l'unité, coupez ou déconnectez d'abord le bloc d'alimentation de la commande, l'alimentation en courant électrique externe et la tension de signal !



**ATTENTION**

0.3

Risque pour l'unité !

Respectez toutes les mesures de protection ESD lors du maniement de l'unité ! Evitez les décharges électrostatiques !



**ATTENTION**

6.1

Lors du démarrage du programme, assurez-vous impérativement que c'est le bon système de coordonnées qui a été activé pour la pièce. Si vous exécutez les programmes avec un système de coordonnées inadapté, des mouvements imprévus risquent de se produire.

Le phénomène peut survenir si, le point ayant été enseigné en mode Apprentissage, vous utilisez des systèmes de coordonnées différents pour l'apprentissage et pour l'exécution du programme.



**ATTENTION**

7.1

La responsabilité de ces réactions involontaires en présence du mauvais fichier TASTEN.BNR incombe au fabricant de la machine. Ne modifiez jamais l'affectation standard des touches sans avoir préalablement consulté la société BOSCH.

**Mesures de sécurité relatives aux dispositifs de commande**

Les pictogrammes et messages d'avertissement suivants peuvent se trouver sur les éléments de commande afin d'attirer votre attention sur certains points:



Présence de tension électrique dangereuse



Danger lié à la présence de batteries



Modules sensibles à l'électricité statique



Enlever la fiche secteur avant l'ouverture



Uniquement pour le raccordement de la terre PE !



Uniquement pour le raccordement d'un câble blindé

## A.2.7 Italiano

### Avvertenze per la sicurezza in queste istruzioni per l'uso



Questi simboli vengono impiegati in queste istruzioni per l'uso nelle seguenti condizioni.



#### PERICOLO

Questo simbolo viene impiegato per segnalare la presenza di **tensioni elettriche pericolose**. La mancata osservanza, anche parziale, di queste istruzioni può provocare danni alle **persone**.



#### PERICOLO

Questo simbolo viene impiegato qualora l'osservanza imprecisa o la mancata osservanza delle istruzioni possono provocare danni alle **persone**.



#### ATTENZIONE

Questo simbolo viene impiegato qualora l'osservanza imprecisa o la mancata osservanza delle istruzioni può provocare danni agli **apparecchi o ai file**.



Questo simbolo viene impiegato quando si voglia richiamare l'attenzione su qualcosa di particolare.



#### PERICOLO

0.1

Pericolo per persone ed oggetti!

Provare ogni nuovo programma prima di mettere in funzione l'impianto!



#### ATTENZIONE

0.2

Pericolo per il modulo!

Non innestare o rimuovere il modulo quando il comando è acceso! Il modulo potrebbe venire distrutto. Spegnere prima il modulo d'alimentazione del comando, l'alimentazione esterna di tensione e la tensione del segnale e solo successivamente innestare o rimuovere il modulo!



#### ATTENZIONE

0.3

Pericolo per i moduli!

Durante operazioni con i moduli rispettare tutte le misure di protezione ESD! Evitare scariche elettrostatiche!



#### ATTENZIONE

6.1

All'avvio del programma è indispensabile accertare che sia attivato il corretto sistema di coordinate del pezzo. Se vengono avviati programmi con sistemi di coordinate del pezzo errati, possono verificarsi movimenti imprevisti.

Lo stesso effetto può verificarsi se in un punto di coordinate spaziale, definite mediante colloquio, si utilizzano diversi sistemi di coordinate del pezzo per il colloquio e per l'elaborazione del programma.



#### ATTENZIONE

7.1

La responsabilità per le reazioni indesiderate, in caso di file TASTEN.BNR eventualmente errato è a carico del produttore della macchina. La modifiche rispetto all'assegnazione standard dei tasti devono avvenire solo d'accordo con la Ditta BOSCH.

**Avvertenze per la sicurezza sui componenti di comando**

Sui componenti di comando stessi possono essere applicate le seguenti targhette di avvertimento e di avvertenza, che richiamano l'attenzione su particolari pericoli:



Avvertimento per tensione elettrica pericolosa!



Avvertimento per pericoli dovuti alle batterie!



Elementi costruttivi danneggiabili da cariche elettrostatiche!



Sfilare la spina dalla rete prima di aprire!



Perno solo per il collegamento del conduttore di protezione PE!



Solo per il collegamento di un conduttore schermato!

## A.2.8 Nederlands

### Veiligheidsrichtlijnen in deze gebruiksaanwijzing



Deze symbolen worden in deze gebruiksaanwijzing onder de volgende voorwaarden gebruikt.



ATTENTIE

Dit symbool wordt gebruikt, als de aandacht op een **gevaarlijke elektrische spanning** gevestigd moet worden. Wordt deze aanwijzing niet precies gevolgd of zelfs genegeerd, dan is **lichamelijk letsel** niet uitgesloten.



ATTENTIE

Dit symbool wordt gebruikt wanneer door onnauwkeurige of niet-naleving van aanwijzingen **schade aan personen** kan worden berokkend.



LET OP

Dit symbool wordt gebruikt wanneer door onnauwkeurige of niet-naleving van aanwijzingen **schade aan toestellen of bestanden** kan worden berokkend.



Dit symbool wordt gebruikt wanneer wij u op iets bijzonders willen attent maken.



ATTENTIE

0.1

Gevaar voor lichamelijk letsel en materiële schade!  
Test elk nieuw programma voor u een installatie opstart!



LET OP

0.2

Gevaar voor de module!  
Als de besturing ingeschakeld is, de module niet insteeken of uittrekken! De module kan hierdoor kapot gaan. De module van het netdeel van de besturing, de externe spanningstoevoer en de signaalspanning uitschakelen of aftrekken en pas dan de module insteeken of uittrekken.



LET OP

0.3

Gevaar voor de module!  
In de omgang met de module alle voorschriften m.b.t. de ESD-beveiliging in acht nemen! Elektrostatische ontladingen vermijden!



LET OP

6.1

Bij de programmastart dient men er beslist op te letten dat het juiste werkstuk-coördinatensysteem is geactiveerd. Indien programma's met een onjuist werkstuk-coördinatensysteem worden gestart, kan dit onverwachte bewegingen tot gevolg hebben. Hetzelfde effect kan optreden indien bij een geteached ruimte-coördinatiepunt verschillende werkstuk-coördinatensystemen worden gebruikt voor het teachen en voor de programma-afwerking.



LET OP

7.1

De verantwoordelijkheid voor ongewenste reacties bij een eventueel onjuist bestand TASTEN.BNR ligt bij de machinefabrikant. Wijzigingen in de standaard toetsentoe wijzing dienen uitsluitend te worden verricht in overleg met de firma Bosch.

**Veiligheidsaanwijzingen bij de besturingscomponenten**

Aan de besturingscomponenten zelf kunnen de volgende waarschuwingen en richtlijnen aangebracht zijn. Zij zijn bedoeld om u op bepaalde zaken te attenderen:



Waarschuwing voor gevaarlijke elektrische spanning.



Waarschuwing voor gevaar veroorzaakt door akku's.



Elektrostatisch gevoelige componenten.



Trek de stekker uit alvorens te openen.



Bouten alleen voor aansluiting van de veiligheidsaarding PE.



Alleen voor aansluiting van een afgeschermde kabel.



## A.2.9 Português

### Instruções de segurança contidas nas presentes instruções de serviço



Estes símbolos são utilizados nas presentes instruções de serviço nos seguintes casos:



#### CUIDADO

Este símbolo é utilizado para indicar uma **tensão eléctrica perigosa**. Em caso de não observância ou observância incorrecta desta instrução, existe **perigo de ferimento de pessoas**.



#### CUIDADO

Este símbolo é utilizado quando existe o **perigo de ferimento de pessoas** por observância incorrecta ou não observância das instruções.



#### ATENÇÃO

Este símbolo é utilizado quando existe o perigo de danificação de aparelhos ou ficheiros por observância incorrecta ou não observância das instruções.



Este símbolo é utilizado para chamar a atenção para algo de especial.



#### CUIDADO

0.1

Perigos de ferimentos de pessoas e de danos materiais!

Antes de colocar uma instalação em funcionamento há que experimentar sempre qualquer programa novo!



#### ATENÇÃO

0.2

Perigo para o módulo!

Não retire ou introduza o módulo quando o comando estiver ligado! O módulo poderá ser danificado. Primeiro desligue ou retire o módulo de alimentação do comando, o cabo alimentador da rede e a tensão de sinal, e em seguida, poderá introduzir ou retirar o módulo!



#### ATENÇÃO

0.3

Perigo para o módulo!

Na utilização do módulo, respeitar todas as prescrições para a protecção do ESD! Evitar descargas electrostáticas!



#### ATENÇÃO

6.1

Ao iniciar o programa é indispensável assegurar-se de que se encontra activo o sistema de coordenadas correcto para peças de trabalho. Se forem executados programas com sistemas de coordenadas errados, podem resultar movimentos inesperados.

O mesmo efeito pode ocorrer quando se utiliza, para um ponto de coordenadas ensinado, diversos sistemas de coordenadas de peças de trabalho diferentes para fins de ensino ou execução de programas.



#### ATENÇÃO

7.1

A responsabilidade por reacções acidentais, devido ao ficheiro TASTEN.BNR ser eventualmente incorrecto, é do fabricante da máquina. Quaisquer alterações na ocupação standard das teclas só podem ser realizadas em comum acordo com a firma BOSCH.

**Instruções de segurança nos componentes de comando**

Nos próprios componentes de comando podem estar afixados os avisos ou as instruções seguidamente descritos para chamar à atenção para determinados pontos.



Aviso referente a uma tensão eléctrica perigosa!



Aviso referente a perigos relacionados com baterias!



Módulos em perigo electrostático!



Antes de abrir tirar o cabo alimentador da rede!



Borne apenas para ligação do condutor de protecção à massa PE!



Só para ligação de um condutor blindado!

## A.2.10 Suomi

### Tämän käyttöohjeen turvallisuusohjeet



Näitä symboleja käytetään tässä käyttöohjeessa seuraavasti.



#### VAROITUS

Tätä symbolia käytetään, kun varoitetaan **vaarallisesta sähköjännitteestä**. Seurauksena voi olla **henkilövahinko**, jos ohjetta ei seurata tai sitä ei seurata tarkkaan.



#### VAROITUS

Tätä symbolia käytetään, jos ohjeiden noudattamatta jättäminen voi johtaa **henkilövahinkoihin**.



#### HUOMIO

Tätä symbolia käytetään, jos ohjeiden noudattamatta jättäminen tai niiden epätarkka seuraaminen voi johtaa **laitteiden tai tiedostojen vahingoittumiseen**.



Tätä symbolia käytetään, kun halutaan kiinnittää lukijan huomio johonkin erikoisseikkaan.



#### VAROITUS

0.1

Henkilö- ja tavaravahinkovaara!

Testaa jokainen uusi ohjelma, ennen laitteiston käyttöönottoa!



#### HUOMIO

0.2

Rakennesaryhmä voi vioittua!

Älä liitä tai irrota rakennesaryhmää ohjauksen ollessa päällekytkettynä! Rakennesaryhmä voi tuhoutua. Kytke ensin ohjauksen verkko-osarakenneryhmä, ulkoinen jännitteen tulo ja signaalijännite pois päältä tai irrota ne ja liitä tai irrota rakennesaryhmä vasta sitten!



#### HUOMIO

0.3

Rakennesaryhmä voi vioittua!

Rakennesaryhmän kanssa toimittaessa on kaikkia ESD-suojaan liittyviä toimenpiteitä noudatettava! Elektrostaattista latausta on vältettävä!



#### HUOMIO

6.1

Ohjelmakäynnistyksessä on ehdottomasti varmistettava, että on aktivoitu oikea työkappaleen koordinaattijärjestelmä. Jos ohjelmat ajetaan väärillä työkappaleen koordinaattijärjestelmillä, siitä voi aiheutua odottamattomia liikkeitä.

Näin voi myös käydä, kun tila-koordinaattipisteessä käytetään erilaisia työkappaleen koordinaattijärjestelmiä teach-in-toiminnossa kuin ohjelman läpiviennissä.



#### HUOMIO

7.1

Koneenvalmistaja kantaa vastuun tahattomista reaktioista käytettäessä mahd. väärää tiedostoa TASTEN.BNR. Muutoksia vakionäppäimistöön saa tehdä vain, kun siitä on ensin keskusteltu BOSCHin kanssa.

### Ohjauksen komponenttien turvallisuusohjeet

Ohjauksen komponentteihin voi olla merkittynä seuraavat varoitukset ja ohjeet, joiden tarkoitus on kiinnittää käyttäjän huomio tiettyihin seikkoihin:



Varoitus, vaarallinen sähköjännite!



Varoitus, akkujen aiheuttamat vaarat!



Sähköstaattisesti vaarannetut rakenneosat!



Vedä verkkopistoke irti pistorasiasta ennen avaamista!



Pultti vain suojajohtimen PE liitännälle!



Vain suojajohtimen litäntää varten!

## A.2.11 Svenska

### Säkerhetsanvisningar i denna driftsinstruktion



Dessa symboler används i denna driftsinstruktion för följande förutsättningar.



VARNING

Denna symbol används, vid varning för **farlig elektrisk spänning**. Om denna anvisning inte exakt följs eller inte följs alls kan det medföra **personskador**.



VARNING

Denna symbol används, när **personer kan skadas** om anvisningar inte exakt följs eller inte följs alls.



OBSERVERA

Denna symbol används, när **apparater eller filer kan skadas** om anvisningar inte exakt följs eller inte följs alls.



Denna symbol används, när Ni skall göras uppmärksam på något särskilt.



VARNING

0.1

Fara för person- och saksador!

Prova varje nytt program innan Ni tar en anläggning i drift!



OBSERVERA

0.2

Fara för en komponentgrupp!

Stick inte in och drag inte heller ur en komponentgrupp när styrningen är tillkopplad! Komponentgruppen kan förstöras. Frånkoppla eller drag först ur styrningens nätdelskomponentgrupp, extern spänningsförsörjning och signalspänningen och stick in eller drag först därefter ut komponentgruppen!



OBSERVERA

0.3

Fara för en komponentgrupp!

Vid arbeten med komponentgruppen skall alla åtgärder för ESD-skydd innehållas! Statiska urladdningar skall undvikas!



OBSERVERA

6.1

Vid programstart är det absolut nödvändigt att rätt operatkoordinatsystem är aktiverat. Om program med fel operatkoordinatsystem körs, kan oväntade rörelser uppträda.

Samma effekt kan uppträda om vid en inlärd rumskoordinatpunkt olika operatkoordinatsystem används för inläring och programkörning.



OBSERVERA

7.1

Maskintillverkaren är ansvarig för oönskade reaktioner vid eventuellt felaktig fil TASTEN.BNR. Ändringar av standardtangentbeläggningen får endast göras i samråd med Bosch.

### Säkerhetsanvisningar på styrningskomponenterna

På styrningskomponenterna kan följande varningar och anvisningar vara placerade, som vill göra Er uppmärksam på vissa saker:



Varning för farlig elektrisk spänning!



Varning för faror genom batterier!



Komponenter som kan skadas av elektrostatisk urladdning!



Drag ur kontakten innan öppning!



Bultar endast för anslutning av skyddsledaren PE!



Endast för anslutning av en avskärningsledare!

Your notes:

# Bosch-Automationstechnik

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Industriehydraulik  
Postfach 30 02 40  
D-70442 Stuttgart  
Telefax (07 11) 8 11-18 57

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Fahrzeughydraulik  
Postfach 30 02 40  
D-70442 Stuttgart  
Telefax (07 11) 8 11-17 98

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Pneumatik  
Postfach 30 02 40  
D-70442 Stuttgart  
Telefax (07 11) 8 11-89 17

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Montagetchnik  
Postfach 30 02 07  
D-70442 Stuttgart  
Telefax (07 11) 8 11-77 12

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Antriebs- und Steuerungstechnik  
Postfach 11 62  
D-64701 Erbach  
Telefax (0 60 62) 78-4 28

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Schraub- und Einpreßsysteme  
Postfach 11 61  
D-71534 Murrhardt  
Telefax (0 71 92) 22-1 81

Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Entgrattechnik  
Postfach 30 02 07  
D-70442 Stuttgart  
Telefax (07 11) 8 11-34 75

Technische Änderungen vorbehalten

Ihr Ansprechpartner

# BOSCH



Robert Bosch GmbH  
Geschäftsbereich  
Automationstechnik  
Antriebs- und Steuerungstechnik  
Postfach 11 62  
D-64701 Erbach  
Telefax (0 60 62) 78-4 28