

Rexroth Rho 4 BAPS3 Short description

1070072177
Edition 08

Software manual



Title Rexroth Rho 4
BAPS3 Short description

Type of Documentation Software manual

Document Typecode DOK-RHO*4*-BAPSK*SOFTH-PR08-EN-P

Purpose of Documentation The present manual informs about:

- language commands available in BAPS3 and
- available BAPS3 functions

Record of Revisions

Description	Release Date	Notes
DOK-RHO*4*-BAPSK*SOFTH-PR07-EN-P	10.2003	Valid from VO07
DOK-RHO*4*-BAPSK*SOFTH-PR08-EN-P	01.2005	Valid from VO08

Copyright © Bosch Rexroth AG, 1998 – 2005

Copying this document, giving it to others and the use or communication of the contents thereof without express authority, are forbidden. Offenders are liable for the payment of damages. All rights are reserved in the event of the grant of a patent or the registration of a utility model or design (DIN 34–1).

Validity The specified data is for product description purposes only and may not be deemed to be guaranteed unless expressly confirmed in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.

Published by Bosch Rexroth AG
Postfach 11 62
D-64701 Erbach
Berliner Straße 25
D-64711 Erbach
Tel.: +49 (0) 60 62/78-0
Fax: +49 (0) 60 62/78-4 28
Abt.: BRC/ESH (KW)

Overview of all manuals

Overview of all manuals

Manual	Contents
Connection conditions Rho 4.0	2 System overview
	3 Installation
	4 Electrical connection
	5 Interfaces
	6 LED display
	7 Maintenance and replacement
	8 Order numbers
System description Rho 4.0	2 System overview
	3 Structure of the rho4.0
	4 PCLrho4.0
	5 CAN-Bus peripheral unit
	6 SERCOS interface
	7 Software
	8 File management
Connection conditions Rho4.1, Rho 4.1/IPC300	2 System overview
	3 Security functions
	4 Installation
	5 Electrical connection
	6 Interfaces
	7 LED display
	8 Maintenance and replacement
	9 Software
	10 Order numbers
Connection conditions Rho 4.1/BT155, Rho 4.1/BT155T, Rho 4.1/BT205	2 System overview
	3 Security functions
	4 Installation
	5 Electrical Connections
	6 Interfaces
	7 Display and Operating Controls
	8 Maintenance and Replacemant
	9 Software
	10 Order numbers
System description Rho 4.1	2 Structure of the rho4.1
	3 PCL
	4 CAN-Bus peripheral unit
	5 SERCOS interface

Overview of all manuals

Manual	Contents
	6 Software
	7 File management
	8 Scope of the rho4.1 Software

Manual	Contents
Control functions	2 Survey of special functions
	3 Accurate position switching
	4 Setting the machine position
	5 Calling operating system functions
	6 Parameterization of the belt characteristic
	7 Selecting a point-file
	8 Mirroring
	9 Belt type
	10 System date and time
	11 System counter
	12 WC main range
	13 Setting the belt counter
	14 Recording of reference path
	15 Flying measurement (rho4.1 only)
	16 MOVE_FILE
	17 Setting the block preparation
	18 Exception-Handling
	19 Belt counter current value
	20 Automatic velocity adjustment for PTP movements
	21 Belt-synchronous working area belt kind 4
	22 Current belt speed
	23 Changing the belt simulation speed
	24 General functions
	25 Process-oriented functions
	26 BAPS3 keywords
	Machine parameters
3 Application of the machine parameters	
4 General system parameters	
5 Speeds	
6 Positions	
7 Kinematic parameters	

Overview of all manuals

Manual	Contents
	8 Measuring system parameters
	9 Belt parameters
	10 Drive parameters Servodyn-GC
	11 Drive parameter Servodyn-D
	12 Table of parameters
Manual	Contents
BAPS3 Programming manual	2 Program structure
	3 Constants
	4 Variables
	5 Program control
	6 Value assignments and combinations
	7 Functions
	8 Movement statement
	9 Write/read functions
	10 BAPS3 keywords
	BAPS3 Short description
3 Constants and variables	
4 Program structure	
5 Value assignments and combinations	
6 Standard functions	
7 Movements and speeds	
8 Belt synchronous	
9 Workspace limitation	
10 Write/read functions	
11 Special functions	
12 Library functions	
13 Fix files	
14 BAPS3 keywords	
Signal descriptions	
	3 Signal description of PCL inputs
	4 Signal description of PCL outputs
Status messages and warnings	2 rho4 status messages
	3 Warnings
	4 CANopen error codes
ROPS4/Online	2 General information
	3 Activation and functions of Online
	4 The function key box

Overview of all manuals

Manual	Contents
	5 Function key assignment
	6 The marker box
	7 File ROPS4WIN.ini
	8 Selection of a file
	9 TCP/IP settings for ROPS4
Manual	Contents
DLL library	2 Library functions
	3 Calling library functions in BAPS
	4 Block structure of the rho4.1
	5 Library server
	6 Application development
	7 rho4 library functions
	8 Variable access per DLL
	PHG2000
3 PHG2000 system variables	
4 Selection of PHG functions	
5 Info function of the PHG	
6 Controlling the PHG2000 output	
7 Define/Teach	
8 SRCAN functions	
9 File and User Memory Functions	
10 File list	
11 Process info	
12 Restoring the PGH display	
13 Variable assignment of PHG keys	
14 Select point file and point name	
15 BDT editor	
Connection conditions Rho 4.1/IPC 40.2	
	3 Security Functions
	4 Installation
	5 Eelectrical Connections
	6 Interface Ports & Connectors
	7 Display- and Operating Components
	8 Maintenance and Replacement
	9 Software
	10 Ordering Informations

Overview of all manuals

Manual	Contents
DDE-Server	2 Introduction
	3 Hardware and Software
	4 Operation
	5 Items of Server 4
	6 Scope of function

Overview of all manuals

Notes:

Contents

Contents

	page
1	Safety Instructions 1-1
1.1	Intended use 1-1
1.2	Qualified personnel 1-2
1.3	Safety markings on products 1-3
1.4	Safety instructions in this manual 1-4
1.5	Safety instructions for the described product 1-5
1.6	Documentation, software release and trademarks 1-7
2	Program structure 2-1
2.1	Program composition 2-1
2.2	Compiler statements 2-2
2.3	Program declarations 2-4
2.4	Subroutine declarations 2-5
3	Constants and variables 3-1
3.1	Standard constants 3-1
3.2	File type 3-2
3.3	User-defined types 3-3
3.4	Record types 3-4
3.5	Point declarations 3-5
3.6	Global variables 3-6
3.7	Array declarations 3-8
3.8	I/O array declarations 3-9
3.9	Channel declarations 3-10
4	Program control 4-1
4.1	Program flow control 4-1
4.2	Parallel processes 4-3
4.3	Software-PLC functions 4-5
5	Value assignments and links 5-1
5.1	Mathematical functions 5-1
5.2	Value assignments 5-2
5.3	ACT/measuring position 5-3
6	Standard functions 6-1
7	Movements and speeds 7-1
7.1	Movement instructions 7-1
7.2	Speeds / accelerations 7-3
7.3	V/A/D-factors 7-5
7.4	Slope speed types 7-6

Contents

8	Belt-synchronous	8-1
9	Workspace limitation	9-1
10	Write/read functions	10-1
10.1	Interfaces	10-1
10.2	I/O files	10-4
11	Special functions	11-1
12	Library functions	12-1
13	Fix files	13-1
13.1	EXPROG.DAT	13-1
13.2	MSD.DAT	13-1
13.3	TEXT.DAT	13-1
13.4	TOOLS.DAT	13-2
14	BAPS3 keywords	14-1
A	Appendix	A-1
A.1	Abbreviations	A-1
A.2	Index	A-2

Safety Instructions

1 Safety Instructions

Please read this manual before you startup the rho4.
Store this manual in a place to which all users have access at any time.

1.1 Intended use


This instruction manual presents a comprehensive set of instructions and information required for the standard operation of the described products. The described products are used for the purpose of operating with a robot control rho4.

The products described

- have been developed, manufactured, tested and documented in compliance with the safety standards. These products normally pose no danger to persons or property if they are used in accordance with the handling stipulations and safety notes prescribed for their configuration, mounting, and proper operation.
- comply with the requirements of
 - the EMC Directives (89/336/EEC, 93/68/EEC and 93/44/EEC)
 - the Low-Voltage Directive (73/23/EEC)
 - the harmonized standards EN 50081-2 and EN 50082-2
- are designed for operation in industrial environments, i.e.
 - no direct connection to public low-voltage power supply,
 - connection to the medium- or high-voltage system via a transformer.

The following applies for application within a personal residence, in business areas, on retail premises or in a small-industry setting:

- Installation in a control cabinet or housing with high shield attenuation.
- Cables that exit the screened area must be provided with filtering or screening measures.
- 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.

 **This is a Class A device. In a residential area, this device may cause radio interference. In such case, the user may be required to introduce suitable countermeasures, and to bear the cost of the same.**

The faultless, safe functioning of the product requires proper transport, storage, erection and installation as well as careful operation.

Safety Instructions

1.2 Qualified personnel

The requirements as to qualified personnel depend on the qualification profiles described by ZVEI (central association of the electrical industry) and VDMA (association of German machine and plant builders) in:

Weiterbildung in der Automatisierungstechnik
edited by: ZVEI and VDMA
MaschinenbauVerlag
Postfach 71 08 64
D-60498 Frankfurt.

The present manual is designed for RC technicians. They need special knowledge on handling and programming robots.

Interventions in the hardware and software of our products, unless described otherwise in this manual, are reserved to specialized Rexroth personnel.

Tampering with the hardware or software, ignoring warning signs attached to the components, or non-compliance with the warning notes given in this manual may result in serious bodily injury or damage to property.

Only electrotechnicians as recognized under IEV 826-09-01 (modified) who are familiar with the contents of this manual may install and service the products described.

Such personnel are

- those who, being well trained and experienced in their field and familiar with the relevant norms, are able to analyze the jobs being carried out and recognize any hazards which may have arisen.
- those who have acquired the same amount of expert knowledge through years of experience that would normally be acquired through formal technical training.

With regard to the foregoing, please note our comprehensive range of training courses. Please visit our website at <http://www.boschrexroth.com>

for the latest information concerning training courses, teachware and training systems. Personal information is available from our Didactic Center Erbach,

Telephone: (+49) (0) 60 62 78-600.

Safety Instructions

1.3 Safety markings on products

Warning of dangerous electrical voltage!



Warning of danger caused by batteries!



Electrostatically sensitive components!



Warning of hazardous light emissions
(optical fiber cable emissions)!



Disconnect mains power before opening!



Lug for connecting PE conductor only!



Functional earthing or low-noise earth only!



Connection of shield conductor only

Safety Instructions

1.4 Safety instructions in this manual



DANGEROUS ELECTRICAL VOLTAGE

This symbol is used to warn of a **dangerous electrical voltage**. The failure to observe the instructions in this manual in whole or in part may result in **personal injury**.



DANGER

This symbol is used wherever insufficient or lacking compliance with instructions may result in **personal injury**.



CAUTION

This symbol is used wherever insufficient or lacking compliance with instructions may result in **damage to equipment or data files**.

☞ This symbol is used to draw the user's attention to special circumstances.

★ This symbol is used if user activities are required.

Safety Instructions

1.5 Safety instructions for the described product

**DANGER**

Danger of life through inadequate EMERGENCY-STOP devices! EMERGENCY-STOP devices must be active and within reach in all system modes. Releasing an EMERGENCY-STOP device must not result in an uncontrolled restart of the system! First check the EMERGENCY-STOP circuit, then switch the system on!

**DANGER**

**Danger for persons and equipment!
Test every new program before starting up a system!**

**DANGER**

**Retrofits or modifications may adversely affect the safety of the products described!
The consequences may include severe injury, damage to equipment, or environmental hazards. Possible retrofits or modifications to the system using third-party equipment therefore have to be approved by Rexroth.**

**DANGER**

Do not look directly into the LEDs in the optical fiber connection. Due to their high output, this may result in eye injuries. When the inverter is switched on, do not look into the LED or the open end of a short connected lead.

**DANGEROUS ELECTRICAL VOLTAGE**

Unless described otherwise, maintenance works must be performed on inactive systems! The system must be protected against unauthorized or accidental reclosing.

Measuring or test activities on the live system are reserved to qualified electrical personnel!

Safety Instructions



CAUTION

Danger to the module!

Do not insert or remove the module while the controller is switched ON! This may destroy the module. Prior to inserting or removing the module, switch OFF or remove the power supply module of the controller, external power supply and signal voltage!



CAUTION

use only spare parts approved by Rexroth!



CAUTION

Danger to the module!

All ESD protection measures must be observed when using the module! Prevent electrostatic discharges!

The following protective measures must be observed for modules and components sensitive to electrostatic discharge (ESD)!

- Personnel responsible for storage, transport, and handling must have training in ESD protection.
- ESD-sensitive components must be stored and transported in the prescribed protective packaging.
- ESD-sensitive components may only be handled at special ESD-workplaces.
- Personnel, working surfaces, as well as all equipment and tools which may come into contact with ESD-sensitive components must have the same potential (e.g. by grounding).
- Wear an approved grounding bracelet. The grounding bracelet must be connected with the working surface through a cable with an integrated 1 MΩ resistor.
- ESD-sensitive components may by no means come into contact with chargeable objects, including most plastic materials.
- When ESD-sensitive components are installed in or removed from equipment, the equipment must be de-energized.

Safety Instructions

1.6 Documentation, software release and trademarks

Documentation

The present manual is a summary of the language commands available in BAPS3. For further details see BAPS3 programming manual and the manual Control functions.

BAPS key words are written in CAPITALS. A list of the keywords is to be found at the end of this manual.

Words written in small letters in the column Syntax of a table may be modified by any user.

Overview of available documentation	Part no.	
	German	English
Rho 4.0 Connectivity Manual	1070 072 364	1070 072 365
Rho 4.0 System description	1070 072 366	1070 072 367
Rho 4.1/IPC 40.2 Connectivity Manual	R911308219	R911308220
Rho 4.1/BT155, Rho 4.1/BT155T, Rho 4.1/BT205 Connectivity manual	1070 072 362	1070 072 363
Rho 4.1, Rho 4.1/IPC300 Connectivity manual	1070 072 360	1070 072 361
Control panels BF2xxT/BF3xxT, connection	1070 073 814	1070 073 824
Rho 4.1 System description	1070 072 434	1070 072 185
ROPS4/Online	1070 072 423	1070 072 180
BAPS plus	1070 072 422	1070 072 187
BAPS3 Short description	1070 072 412	1070 072 177
BAPS3 Programming manual	1070 072 413	1070 072 178
Control functions	1070 072 420	1070 072 179
Signal descriptions	1070 072 415	1070 072 182
Status messages and warnings	1070 072 417	1070 072 181
Machine parameters	1070 072 414	1070 072 175
PHG2000	1070 072 421	1070 072 183
DDE-Server 4	1070 072 433	1070 072 184
DLL-Library	1070 072 418	1070 072 176
Rho 4 available documentation on CD ROM	1070 086 145	1070 086 145

 **In this manual the floppy disk drive always uses drive letter A:, and the hard disk drive always uses drive letter C:.**

Safety Instructions

Special keys or key combinations are shown enclosed in pointed brackets:

- Named keys: e.g., <Enter>, <PgUp>,
- Key combinations (pressed simultaneously): e.g., <Ctrl> + <PgUp>

Release

 **This manual refers to the following versions:**

Hardware version: rho4

Software release: ROPS4

Trademarks

All trademarks of software installed on Rexroth products upon delivery are the property of the respective manufacturer.

Upon delivery, all installed software is copyright-protected. The software may only be reproduced with the approval of Rexroth or in accordance with the license agreement of the respective manufacturer.

MS-DOS® and Windows™ are registered trademarks of Microsoft Corporation.

PROFIBUS® is a registered trademark of the PROFIBUS Nutzerorganisation e.V. (user organization).

MOBY® is a registered trademark of Siemens AG.

AS-I® is a registered trademark of AS-International Association.

SERCOS interface™ is a registered trademark of Interessengemeinschaft SERCOS interface e.V. (Joint VDW/ZVEI Working Committee).

INTERBUS-S® is a registered trade mark of Phoenix Contact.

DeviceNet® is a registered trade mark (TM) of ODVA (Open DeviceNet Vendor Association, Inc.).

Program structure

2 Program structure

2.1 Program composition

Syntax	Description
;;CONTROL = RHO4	Declaration of control type
;;KINEMATICS: (1=ma_1, 2=ma_2)	Number of kinematics with kinematic names and numbers. statements as in the machine parameter converter.
;;ma_1.JC_NAMES= A_1,A_2,A_3,A_4	Number of axes and names in JC for kin 1
;;ma_1.WC_NAMES= X_K,Y_K,Z_K,A_K	Number of axes and names in WC for kin 1
;;ma_2.JC_NAMES= B_1,B_2,B_3	Number of axes and names in JC for kin 2
;;ma_2.WC_NAMES= X_B,Y_B,Z_B	Number of axes and names in WC for kin 2
PROGRAM example	Program name max. 8 characters
;;KINEMATICS = ma_1	All movement instructions and point statements without kinematic statement refer to kinematic ma_1, kinematic preselection.
EXTERNAL: maon, maoff	Declarations for external subroutines or processes.
INPUT: 1=partthere, 2=push_forward	Declarations for inputs/outputs
OUTPUT: 1=belt_on	
SPC_FCT: 1=path_io (VALUE INTEGER: . . .)	Declaration for special functions The order or other variable declarations can be at will, but must be placed before BEGIN.
ma_1.POINT: pal_corner_1, pal_corner_2 ma_2.JC_POINT: @stroke_center	Declare points for different kinematics.
BEGIN .	Beginning of main program
MOVE ma_2 TO @stroke_center	
subexample .	Call of subroutine
PROGRAM_END	End of main program
SUBROUTINE subexample	Subroutine identification, max. 12 characters
INPUT: 4= li_bar REAL: counter	Declaration part for the subroutine
BEGIN .	Beginning of subroutine
belt_on = 1	
SUB_END	End of the subroutine



Program structure

2.2 Compiler statements

With exception of the statements KINEMATIC, INCLUDE, WARNING and INT, all compiler statements must be placed in front of the program names. The compiler statements must not be followed by further characters in the same line (comments represent an exception).


Syntax	Description
<code>::CONTROL = rho4</code>	Declaration of control types at present available: rho4.
<code>::DEBUGINFO+</code>	With the compiler statement DEBUGINFO+, programs can be tested via the test system.
<code>::DEBUGINFO-</code>	With DEBUGINFO-, no testing is possible, the IRD file is, however, smaller and is processed faster. The default is DEBUGINFO+.
<code>::WARNING+</code> <code>::WARNING-</code>	With the compiler statement WARNING-, the output of warnings into the error file can be suppressed. The default is WARNING+.
<code>::ERROR = 10</code>	Sequence errors often occur during the compilation so that the error file gets very big. By the definition of the max. number of errors, 1 to 100, the compilation procedure is aborted when having reached the set value. Without this statement, max. 100 errors will be put out.
<code>::KINEMATICS: (1=ma_1, 2=ma_2)</code>	Definition of the number of kinematics with the corresponding kinematic numbers and kinematic names.
<code>::ma_1.JC_NAMES= A_1,A_2</code>	Definition of the number of axes and names in machine coordinates (JC) for kin1. ☞ This statement must be in one line
<code>::ma_1.WC_NAMES= XC,YC</code>	Definition of the number of axes and names in world coordinates (WC) for kin1. ☞ This statement must be in one line
<code>::PROCESS_KIND= PERMANENT</code>	Definition that this program can run as permanent process. A permanent process runs in the operation modes auto and manual. A permanent process is not aborted by a reset. ☞ This statement must be in one line
<code>::KINEMATICS = ma_2</code>	Definition of a global kinematic. If not otherwise defined by this declaration, the preset kinematic is always No. 1. ☞ This statement must be in one line

Program structure

Syntax	Description
;;INCLUDE exdat	The compiler inserts here the file with the file extension '.qll' (exdat.qll) and generates the IRD code for it.  No further INCLUDE statements must be in exdat.qll.
;;INCLUDE exdat.inc	The file 'exdat.inc' is inserted.
;;INCLUDE exdat.	The file 'exdat.' is inserted without file extension. The file extension are the three letters following the dot in a file designation.
;;INT = PTP ;;INT = LINEAR ;;INT = CIRCULAR	Default of the type of interpolation. The type of interpolation can be determined in a program by the compiler statement, the default is PTP.
;;ma_2.INT = PTP ;;ma_2.INT = LINEAR ;;ma_1.INT = CIRCULAR	Default of the type of interpolation with kinematic assignment. The type of interpolation can be determined in a program by the compiler statement, the default is PTP.
;;SER_IO_STOP-	With the compiler statement, the abort of a user program can be avoided if an interface error occurs.
;;SER_IO_STOP+	This compiler statement indicates that a user program is to be interrupted if an interface error has occurred.
;;FILE_ERROR-	No program abort in case of an ASSIGN statement to an opened file.  Errors occurred are stored in the meantime internally. They can be inquired with the standard function CONDITION.
;;FILE_ERROR+	Program abort in case of an ASSIGN statement to an opened file, default.




Program structure

2.3 Program declarations

Syntax	Description
PROGRAM fina . . .	Main program declaration, the length of the program name must be max. 8 characters.  The first character must be a letter. As special character only the underline “_” is admitted.
BEGIN . PROGRAM_END	Main program begin/end. After the declaration part, the main program begins with the keyword BEGIN. It ends with the keyword PROGRAM_END.
BEGIN . HALT PROGRAM_END	Main program end with HALT. This HALT is to be used optionally and can also be at several locations in the main program if the program is to be ended at several locations (e. g. in case of IF...THEN... inquiries).
EXTERNAL: name_sub	External subroutine definition. The names for the external subroutines must be declared before the keyword BEGIN.
name_sub	External call of subroutine
program name_sub (VALUE INTEGER: co,li VALUE REAL: de_n)	Program declaration with parameter transfer. Can be called as external subroutine.
EXTERNAL:name_sub (VALUE INTEGER: co, li, VALUE REAL: de_n)	Definition of external subroutines with parameter transfer.
name_sub (3,2,-50)	Call of subroutine with parameter transfer.

Program structure

2.4 Subroutine declarations

Syntax	Description
SUB subname BEGIN . SUB_END	Subroutine declaration The length of the subroutine name must not exceed 12 characters.  The first character must be a letter. As special characteri only the underline ”_” is admitted.
SUBROUTINE sub_name(VALUE REAL:aw) BEGIN . SUB_END	Subroutine declaration with parameter list. If in the subroutine 'sub_name', the variable aw is changed, the variable oh will be unchanged after the return, see below.  With the term VALUE, only the value of the variable is transferred. The value of the variable in the called program remains unchanged.
SUBROUTINE sub_name(REAL:aw) BEGIN . SUB_END	Subroutine declaration with parameter list. If in the subroutine 'sub_name', the variable aw is changed, the variable oh will have the changed value after the return, see below.  Without the keyword VALUE, the address of the variable will be transferred.
. sub_name . .	Call of subroutine A subroutine call can be made from the main program. From the subroutine, also another subroutine can be called.
. sub_name (oh) . .	Call of subroutine with parameter transfer
RETURN	End subroutine Can be used as an option to be able to quit the subroutine at different locations.

Program structure

Notes:

Constants and variables

3 Constants and variables

3.1 Standard constants

Syntax	Description
WRITE PHG, CLS	Clear Screen (CLS) By output of CLS the PHG display will be cleared.
IF VERSION < 2.44 THEN WRITE PHG,'old compiler version'	VERSION In the BAPS program, the version number of the compiler can be inquired. The constant is of the real type (REAL).

Constants and variables

3.2 File type

Syntax	Description
POINT: p_name	Points in world coordinate presentation
JC_POINT : @p_name	Points in machine coordinate presentation
ma_1.POINT: kp_name	Points in world coordinate presentation with kinematic assignment.
ma_2.JC_POINT: @kp_name	Points in machine coordinate presentation with kinematic assignment.
INTEGER: i_var	Integral values (integer)
REAL: d_var	Decimal (real) floating point
BINARY: b_var	Binary variable resp. I/O signals. There are two conditions, logic 0 or logic 1.
CHAR: char	ASCII characters according to DIN 66003
TEXT: message	String, e. g. 'school'
FILE: finadat	Here, the names of the files with the extension .dat are defined. From the .dat files, values can be read or written, see section 10.2, file I/O.
SEMAPHORE: sema_var	Variables of the type SEMAPHORE are required for the EXCLUSIVE statement.

Constants and variables

3.3 User-defined types

By the expansion of the declaration part of a BAPS program for the type definition part, the user can define own types in BAPS. It is thus possible to assign new names to any type of files. This applies to both the standard types, e. g. BINARY, INTEGER, and the structured types formed therefrom, such as arrays. This leads to clearly structured BAPS programs.

Syntax	Description
PROGRAM types TYPE: TMatrix = ARRAY [0..7] ARRAY [0..7] INTEGER MyInteger = INTEGER	 Type definition part Two-dimensional array wit T-Matrix New name for INTEGER
ARRAY[0..7]ARRAY[0..7]INTEGER: MatrixOne TMatrix: MatrixTwo MyInteger:INTEGERVar	Variable definition part
BEGIN InitOld (MatrixOne) InitNew (MatrixTwo) PROGRAM_END	
UP InitOld (ARRAY[0..7] ARRAY[0..7] INTEGER:Matrix) BEGIN . SUB_END	Long way of writing Variable matrix initialization
UP InitNew (TMatrix:Matrix) BEGIN . SUB_END	Short, clearly arranged way of writing Variable matrix initialization

Constants and variables

3.4 Record types

Using record types results in an improvement with regard to the clear arrangement and structuring of programs.

Syntax	Description
PROGRAM records TYPE: Tname = RECORD TEXT: first name, last name, Title RECORD_END Taddress = RECORD TEXT: street INTEGER: ZIP_code TEXT: town RECORD_END Tstaffmbr = RECORD Tname: name Taddress: address TEXT: SV_number BINARY: salary RECORD_END	Type definition part
Tstaffmbr: worker	Variable definition part Declaration of a staff member
BEGIN . Worker.SV_number = 0815 Worker.lastname.firstname = 'Kalli' Worker.name.title = 'Dr.' PROGRAM_END	Access to the components Access to the components of a component

Constants and variables

3.5 Point declarations

Syntax	Description
DEF POINT: d_pt	Point definition in world coordinates, which is stored in a point file. It can be influenced in the modes Define, Teach In and by the BAPS program.
DEF JC_POINT: @d_mcp	Point definition in machine coordinates, which is stored in a point file. It can be determined in the modes Define, Teach In and by the BAPS program.
DEF ma_1.POINT: d_pt DEF ma_1.JC_POINT: @d_pt	As POINT, but additionally with kinematic assignment. As JC_POINT, but additionally with kinematic assignment.
DEF ARRAY [1..6] POINT: ap	Point definition in world coordinates via an array with min. and max. limit.
DEF ARRAY [1..6] JC_POINT: @ap	Point definition in machine coordinates via an array with min. and max. limit.
DEF ARRAY [1..6] ma_1.POINT: ap DEF ARRAY [1..6] ma_2.JC_POINT: @ap	As POINT, but additionally with kinematic assignment. As JC_POINT, but additionally with kinematic assignment.

Constants and variables

3.6 Global variables


Global variables enable the easy data exchange between several independent BAPS user programs (processes). The basic idea is to combine programs working with the same global variables in one program group. Within this group, it is possible for a program to export data while the other programs of this group are only allowed to import data. On the control, the number of program groups is not limited (only by the evaluable storage place).

Program example

Syntax	Description
PROGRAM exp_var	Exporting program
PUBLIC DEF POINT: start_pos PUBLIC INTEGER: index PUBLIC SEMAPHORE: write_prot	Declaration part in the exporting program Global data Global variables are declared by adopting the keyword GLOBAL into the type declaration of these variables. The data range for these variables is reserved in the ird file. Teachpoints, e. g. identified by the keyword DEF, are stored in the PNT file.
REAL: mvalue	Local data
BEGIN EXCLUSIVE write_prot start_pos=POS index= 5 EXCLUSIVE_END . . . PROGRAM_END	BAPS statements Additional statements

Syntax	Description
PROGRAM imp_var	Importing program
EXTERNAL exp_var: start_pos EXTERNAL exp_var: index EXTERNAL exp_var: write_prot	Declaration part in the importing program Global data The variables can be accessed from other BAPS programs if the variables are declared with EXTERNAL and the name of the exporting program.
POINT: end_pos	Local data

Constants and variables

Syntax	Description
BEGIN EXCLUSIVE write_prot REPEAT index TIMES	BAPS statements
MOVE UNTIL start_pos MOVE UNTIL end_pos REPEAT_END EXCLUSIVE_END PROGRAM_END	 Restrictions, see next page

Restrictions

When using global data, the following restrictions have to be taken into account:

- The exporting program of a program group must be compiled before the importing programs of this group. This is necessary due to the type verification by the compiler. The user has to make sure that each importing program is of a more recent date than the exporting program. If this is not the case, an error message or warning will be displayed when starting the program.
- In a program it is not possible to import data and to export data simultaneously.
- Only data from one program can be imported.
- Global data can only be exported or imported in the declaration part of the main program, not in subroutines.
- To ensure the transfer of a whole data block without interruptions, the access to the variables must be protected by using the EXCLUSIVE statement. The consistency, i. e. the validity of the data of simpler standard types, such as BINARY, INTEGER, REAL and CHAR, is given by the operating system.

Constants and variables

3.7 Array declarations

Arrays are data structures consisting of a fix number of elements of the same type. The min. and max. limits are indicated in the brackets. After the bracket the data type is defined. When declaring arrays of element types, the array limits are monitored within the following range [-8388608 to 8388607].

Syntax	Description
ARRAY [1..6] POINT: ap	Array of the type POINT in world coordinates
ARRAY [1..6] JC_POINT: @ap	Array of the type POINT in machine coordinates
ARRAY [1..6] ma_2.POINT: ap	Array of the type POINT with kinematic assignment
ARRAY [1..6] ma_1.JC_POINT: @ap	Array of the type JC_POINT with kinematic assignment
ARRAY [1..6] CHAR: ac	Array of the type CHAR
ARRAY [21..36] TEXT: at	Array of the type TEXT
ARRAY [1..6] INTEGER: ai	Array of the type INTEGER
ARRAY [21..36] REAL: ad	Array of the type REAL
ARRAY [1..6] BINARY: ab	Array of the type BINARY
ARRAY [1..4] ARRAY [1..2] INTEGER: a2diminteger	Two-dimensional array of the type INTEGER
a2value = a2diminteger [1] [2]	Access to array components

Constants and variables




3.8 I/O array declarations

Syntax	Description
ARRAY [1..2] OUTPUT BINARY: (1,32) = o_sen_F1	Array of the type OUTPUT BINARY
ARRAY [11..14] INPUT BINARY: (5,6,7,22) = i_sen_F2	Array of the type INPUT BINARY
ARRAY [1..3] OUTPUT INTEGER: (401,402,403) = oin_sen_F3	Array of the type OUTPUT INTEGER
ARRAY [11..13] INPUT INTEGER: (401,404,408) = iin_sen_F4	Array of the type INPUT INTEGER
ARRAY [1..2] OUTPUT REAL: (201,202) = od_sen_F5	Array of the type OUTPUT REAL
ARRAY [11..12] INPUT REAL: (201,203) = id_sen_F6	Array of the type INPUT REAL

 I/O arrays are only allowed to be declared one-dimensionally



Constants and variables

3.9 Channel declarations

Syntax	Description
INPUT BINARY: 2 = gr_force OUTPUT BINARY: 9 = magnet	<p>The names of the user inputs/outputs are declared. The names must not be identical with BAPS keywords. The BINARY statement can be omitted since the type BINARY is the default.</p> <p> Admissible channel numbers: 1 to 199, resp. 610 for sensor inputs.</p>
INPUT INTEGER: 401 = decade OUTPUT INTEGER: 403 = lcd_display	<p>The names of the user inputs/outputs of the type INTEGER are declared. They are put out at the interface with a width of 8 bits (significance 0 to 256).</p> <p> Admissible inputs: 401 to 408 Admissible outputs: 401 to 416</p>
BELT: 501 = belt1_ma1 ma_2.BELT: 502 = belt2_ma2	<p>The names of the belts are declared.</p> <p> Admissible channel numbers: 501 to 516</p>

Asynchronous inputs

The declaration of asynchronous inputs is made by indicating an offset of 1000 in the channel number.

Syntax	Description
INPUT BINARY: 1002 = gr_force	<p>The names of the asynchronous inputs are declared. The names must not be identical with BAPS keywords. The BINARY statement can be omitted since the type BINARY is the default.</p> <p> Admissible channel numbers: 1001 to 1199</p>
INPUT INTEGER: 1401 = decade	<p>The names of the asynchronous inputs of the type INTEGER are declared.</p> <p> Admissible inputs: 1401 to 1408 Admissible outputs: 1401 to 1416</p>


Program control

4 Program control

4.1 Program flow control

Syntax	Description
WAIT 5.5	The execution of the program is stopped for the indicated time. The constant 5.5 indicates the time in seconds.
REAL: delay_time delay_time = 5.5 WAIT delay_time	Instead of a constant it is also possible to use a variable. This must have been declared before as decimal (REAL) or integral (INTEGER) variable. <ul style="list-style-type: none"> ☞ A value must have been assigned to the variable before using it.
INPUT: 4 = part_avail WAIT UNTIL part_avail = 1	Waiting until a condition has been met. <ul style="list-style-type: none"> ☞ In the condition, only the inquiry of inputs is permitted. ☞ If the condition consists of several input channels, the WAIT UNTIL instruction must be divided into several steps. <p>Example: WAIT UNTIL signal=1 WAIT UNTIL light=1</p> <p>WAIT UNTIL signal=1 AND light=1 would be wrong The compiler would interpret this instruction as follows: WAIT UNTIL signal= ((1 AND light)=1)</p>
INPUT: 5 = sig WAIT UNTIL sig = 0 MAX_TIME = 5.6 ERROR statement1	Waiting until a condition with a time limit has been met. If the time has elapsed, the instruction is processed errorwise. If the condition is met within the indicated time frame, the error statement is ignored and the next BAPS instruction is processed.
PAUSE	The execution of the program is stopped. The continuation of the program is made with the signal program start.
REPEAT 5 TIMES . . REPEAT_END	A fix number of program cycles is executed between the two instructions.

Program control

Syntax	Description
INTEGER: number number = 5 REPEAT number TIMES REPEAT_END	Instead of a constant it is also possible to use a variable. This must have been declared before as INTEGER.  A value must have been assigned to the variable before using it.
IF condition THEN statement1 ELSE statement2	Conditional statement IF the condition is met, THEN the control executes statement 1, ELSE statement 2 is executed if the condition has not been met.
IF in1 = 1 THEN BEGIN of36 = 0 of01 = 1 END ELSE BEGIN END	BEGIN ... END record At places at which only a single statement can appear, several statements can be combined by using the record statement BEGIN ... END.
JUMP label_1 label_1:	Within main programs or subroutines it is possible to jump to labels. In this case, forward and backward jumps are possible.
INTEGER:amount,offset BEGIN amount = 0 offset = 0 READ amount CASE amount EQUAL 9,10 : offset = 0 EQUAL 11,12 : offset = 1 EQUAL 13 : offset = 2 DEFAULT offset = 5 CASE_END PROGRAM_END	By means of the branching statement CASE, a selection from several alternatives can be made. Interleaved IF – THEN inquiries can thus be avoided and the execution time of the program can be reduced.
INPUT:1 = I1 OUTPUT:1 = O1 BEGIN LOOP: IF I1 = 1 THEN P1 = 1 BREAK JUMP LOOP PROGRAM_END	The BAPS function BREAK is realized as BAPS standard subroutine without parameters. After each loop process, the example process enables the CPU prematurely. The remaining time until the next clock interrupt is made available to other processes with the same priority.

Program control

4.2 Parallel processes

External

Syntax	Description
EXTERNAL: gripper	External definition of a program in the declaration part.
START gripper	The main program and external program (gripper) are processed parallel.
START gripper PRIO=100	On a scale, a priority from 100 (highest) to 150 (lowest) can be allocated.
STOP gripper	The parallel program (gripper) is stopped. Only the main program continues running.

☞ **No synchronized end of the parallel processes.**

☞ **If a process, which is possibly still running, is to start again from the beginning, 'STOP gripper' and then 'START gripper' can be used!**

Internal

Syntax	Description
<pre> . . PARALLEL MOVE pal_pos_1 I = 5 . . </pre>	1st process,
<pre> ALSO MOVE ma_2 TO pal_corner_I </pre>	2nd process and
<pre> ALSO . </pre>	3rd process run simultaneously.
<pre> PARALLEL_END . . MOVE TO home </pre>	If all processes are finished, the program will continue after PARALLEL_END.

☞ **Synchronized end of the parallel processes.**

Program control

Syntax	Description
SEMAPHORE: sema_var	<p>If common resources, e. g. a printer, are to be accessed exclusively, this can be achieved by using the exclusive statement. The semaphore variable must have been declared before in the declaration part. It has a global effect in the operating system, i. e. an interlocking can be made by all programs.</p> <p>If another parallel process has occupied the semaphore, the second process dwells until the semaphore is enabled again.</p>
EXCLUSIVE sema_var	With the exclusive statement, one semaphore can be occupied.
EXCLUSIVE_END	The EXCLUSIVE_END instruction enables the semaphore again.

 **If the parallel process does not inquire the semaphore, the interlocking remains without effect.**

Syntax	Description
EXCLUSIVE sem_var subroutine_call EXCLUSIVE_END	A semaphore can be occupied several times by one and the same process without causing an internal blocking.
SUBROUTINE subroutine_call BEGIN	
EXCLUSIVE sem_var ;statements EXCLUSIVE_END	
SUB_END	

Program control

4.3 Software-PLC functions

The rho4 works with the integrated software-PLC PCL, which supplies an own image and an access to the inputs and outputs of the rho4. The programming of the software-PLC is made under WinSPS.

Activation of programs

Syntax	Description
START PLC_PROCESS (PB_xxx)	Starts process on the PLC, BAPS program is continued. ☞ The argument (PB_xxx) can be both a constant and a variable of the type INTEGER.
STOP PLC_PROCESS (PB_xxx)	The process will no longer be executed in the next PLC cycle.
PLC_PROCESS (PB_xxx)	Unique call of a PLC program sequence. The processing of a BAPS program is only continued when the PLC program sequence has ended.
cond = CONDITION (PLC_PROCESS (PB_xxx))	Inquiry of the process status 0 = not active 1 = active

Activation of programs

Syntax	Description
START PLC_TIME(TASK_NO) ALL T = n	Starts PLC time task with time value default. ☞ n = time value in sec.
STOP PLC_TIME (TASK_NO)	Ends PLC time task with the next call.
cond = CONDITION (PLC_TIME (TASK_NO))	Inquiry of a task status 0 = not active 1 = active

Program control

Notes:

Value assignments and links

5 Value assignments and links

5.1 Mathematical functions

Syntax	Description
+ - * / MOD	Arithmetic operations Addition Subtraction Multiplication Division The modulo function determines the integral rest by the division of two values.
SIN COS ATAN SQRT	Standard functions Sinus Cosinus Arcus tangent Root function
AND OR NOT	Logical operations
< , > <= , >= <> =	Comparative operations less, greater, less-equal, greater-equal, unequal, equal

Value assignments and links

5.2 Value assignments

Syntax	Description
CONST: yellow = 0, white = 1, blue = 4, red = 3 INTEGER: colour	Constants Constants are defined before the variables of a program. The type of the constants results from the assigned value.
variable = expression	By means of an assignment, a new value is assigned to a variable. This value must be compatible in type with the variable type.
p1 = (0,0,-50.123,0) @p1 = @(0,0,-19.4,0)	Position assignment Complete value assignment of a world or machine coordinate point. The number of the components equals the number of axes of the kinematic.
REAL: a_value a_value = -50.0 p1 = (0,0,a_value,0)	Value assignment of a point via variables.
p2.Z_C = 31.2 p1 = (0,0,p2.Z_C,0)	Axis value assignment The axis designation is used for the value assignment.
INTEGER: i . . i = 0 i = i+1	The value assignment can be used for all data types, such as REAL or INTEGER. Also for all standard variables, such as V_PTP, A, VFACTOR, AFACTOR etc.
INTEGER: int_var, int_var1, int_var2 REAL: real_var, real_var_r int_var = int_var1/int_var2 real_var = 1.0 * int_var1/int_var2 real_var_r = int_var1/int_var2 real_var = int_var1 real_var = real_var/int_var2	Value assignments with equal, resp. different variable types Result is integral part Corresponds to TRUNC (real_var_r) Integral part without position after decimal point To get the position after decimal point with a division of a variable of type INTEGER by a variable of type INTEGER, the dividend must assigned before to a variable of type REAL

Value assignments and links

5.3 ACT/measuring position

Syntax	Description
<pre> posa = POS IF posa.X_K > 300 THEN posa.X_K = 300 MOVE TO posa </pre>	<p>Actual position</p> <p>By means of the actual position (POS), the current position of the robot can be read and is thus available to the program for further evaluation (acts on the pre-set kinematic).</p>
<pre> posa = ma_2.POS IF posa.X_K > 300 THEN posa.X_K = 300 MOVE TO posa </pre>	<p>Actual position</p> <p>As before, however, with indication of kinematic.</p>
<pre> @actual = @POS </pre>	<p>The actual position can also be read in machine coordinates. Acts on the preset kinematic.</p>
<pre> @actual = ma_1.@POS </pre>	<p>The actual position can also be read in machine coordinates.</p>
<pre> @p1 = @MPOS @p1 = ma_1.@MPOS p1 = WC(ma_1.@MPOS) </pre>	<p>The component @MPOS contains the measuring position which can be determined with high accuracy by means of the probe inputs*. @MPOS is only effective in the machine coordinate system, but can be converted with the standard function JC.</p> <p>* only available with rho4.1 and incremental interface</p>

Value assignments and links

Notes:

Standard functions


6 Standard functions

Standard functions TRUNC, ABS, ROUND, ORD, CHR

Syntax	Description
INTEGER: i1 REAL: r1 r1 = TRUNC (12.58) i1 = TRUNC (1.32)	Standard function TRUNC Truncation of the places after the decimal point with assignment of an integral value or variable resp. assignment of a decimal value or variable.
INTEGER: i1, gt REAL: i1, dt dt = ABS (d1) dt = ABS (i1) gt = ABS (i1)	Standard function ABS Forming of absolute value (amount) of INTEGER resp. REAL variables or numbers.
REAL: d1 INTEGER: i1 d1 = ROUND (10.51) i1 = ROUND (1.32)	Standard function ROUND Assignment of a decimal number or variable into an INTEGER or REAL variable with rounding. > 0.5 is rounded up, < = 0.5 is rounded down.
INTEGER: varo1 varo1 = ORD ('a')	The standard function ORD converts an ASCII character in an integral value. In this example varo1 = 97, since ASCII value of a = 97. From compiler version 2.40, expressions of the types INTEGER and BINARY can also be used.
INTEGER: varo2, varo3 CHAR: ch_var ARRAY[1..8] CHAR: txt txt = 'pRima____' varo2 = ORD (txt [2]) ch_var = 'a' varo3 = ORD (ch_var)	ORD with array variable In this example txt[2] = R i. e. varo2 = 82, since ASCII value of R = 82 ORD with variable in this example varo3 = 97, since ASCII value of a = 97
CHAR: varc1 varc1 = CHR (97) varc1 = CHR (80 + 17)	The standard function CHR converts an integral value into an ASCII character. The character to be presented depends on the used character set. In this example varc1 = a
CHAR: ch_varc2 ARRAY[1..3] INTEGER: ainteger ainteger [2] = 97 ch_varc2 = CHR (ainteger [2])	The value range von CHR (X) should be between 0 and 255. No range check will be made. For values outside this range, the function argument is converted internally as follows: $X = (X \text{ MOD } 256)$ CHR with variable resp. array variable In this example varc2 = a

Standard functions

Standard function INT_ASC


Syntax	Description
INTEGER: int_number INTEGER: index_asc INTEGER: length_char INTEGER: error_mess ARRAY[1..10] CHAR: ascii_array	Default int_number = number to be converted. Only admissible of the type INTEGER, e. g. integer_number < = 2147483647 Result message ascii_array = result array of the type ARRAY [1..10] CHAR
int_number = -1234 index_asc = 1 length_char = 5	Default index_asc = start index in the character array Default length_char = maximum number of characters reserved for the number to be converted
INT_ASC (int_number, ascii_array, index_asc, length_char, error_mess)	 The destination range is initialized before the conversion with blanks.
IF error_mess <> 0 THEN WRITE 'Error at INTEGER-ASCII conversion' ELSE WRITE ascii_array	Result message error_mess = error number output

Meaning of error number

- 0 no errors
- 1 start index is outside the array limits
- 2 end index, consisting of start index and length, is outside the array limits
- 3 reserved length is too short
- 4 range exceeding (value too high)
- 5 array length < 0
- 6 array length = 0

The standard procedure INT_ASC converts an integral value into an array of characters.

Standard functions

-  **The format is right-aligned, possibly with leading empty characters.**
If an error is detected during the conversion, the character array remains unchanged and the corresponding error code is returned.

Standard function ASC_INT

Syntax	Description
INTEGER: int_number INTEGER: index_asc INTEGER: length_char INTEGER: error_mess ARRAY [1..10] CHAR: ascii_array ascii = 'AaBc' index_asc = 1 length_char = 5 ASC_INT (int_array, int_number, index_asc, length_char, error_mess)	Result message int_number = converted number (only admissible of the type INTEGER) Default index_asc = position in the array from which on the number is to be read length_char = max. number of characters to be read ascii_array = array to be converted of the type ARRAY [1..10] CHAR
IF error_mess <> 0 THEN WRITE 'Error at ASCII-INTEGER conversion' ELSE WRITE int_number	Result message error_mess = error number output

Standard functions

Meaning of error numbers

- 0 no errors
- 1 start index is outside the array limits
- 2 end index, consisting of start index and length, is outside the array limits
- 3 -----
- 4 range exceeding (value too high)
- 5 array length < 0
- 6 array length = 0
- 7 Character string does not start with a number or a sign

The standard procedure ASC_INT converts an array of characters into an integral value.

The procedure reads characters, beginning with the start position, until a character is recognized which is not a number or the maximum number of characters has been read, resp. the end of the character array has been reached.

Standard functions

Standard functions JC, WC, R, R_PTP

Syntax	Description
@pa = JC (ph) @p1 = JC ((180, 90)) + @p3	Conversion of world coordinates into joint coordinates
pos_a = WC (@pal) pos_1 = p3 + WC (@p4)	Conversion of joint coordinates into world coordinates
R = 50 MOVE LINEAR VIA p1	Passing radius, block-exceeding Radius R is effective with linear interpolation (kinematic-specific)
MOVE LINEAR WITH R = 20 VIA p1	Passing radius, block-specific Radius R is effective with linear interpolation (kinematic-specific)
R_PTP = 1.4 ;(or R_PTP=140%) MOVE PTP VIA p2	Passing distance factor, block-exceeding R_PTP is effective with PTP interpolation (kinematic-specific) ☞ R_PTP is multiplied by the value indicated in machine parameter P213.
R_PTP = 1.4 ;(or R_PTP = 140%) MOVE PTP with R_PTP = 0.5 VIA p2	Passing distance factor, block-specific R_PTP is effective with PTP interpolation (kinematic-specific)

☞ If R = 0 or R_PTP = 0 is programmed, the spatial passing is disabled.

Standard functions

Standard function CONDITION

Application possibilities of the CONDITION function

- File/interface
- Process
- I/O

Syntax	Description
<pre> PROGRAM io_test EXTERNAL: procname INTEGER: index,number,status FILE: fina CONSTANT: txtconst = 'E85.7' txtbyconst= 'E13' BEGIN number = 0 READ V24_1, index IF CONDITION(V24_1) < 0 THEN WRITE 'read error' </pre>	<p>The standard function CONDITION permits</p> <p>to determine the status of an interface.</p> <p>Supported are V24_1 to V24_4, PHG, TTY, Win_1 to Win_4, PLC</p> <p>Function values</p> <ul style="list-style-type: none"> 0 no error -1 interface not found -2 interface disabled -3 timeout -4 parity error -5 overrun error -6 framing error -7 interface error -8 wrong string length -9 protocol error -10 inadmissible real-expression -11 point not defined -30 BUEP data error PCL -31 data module to PCL not ok -32 data module length too large -33 data module length not engaged -34 data module number not engaged
<pre> status = CONDITION(fina) </pre>	<p>to determine the status of a file.</p> <p>Function values</p> <ul style="list-style-type: none"> 0 no error, file available -1 file not available

Standard functions

Syntax	Description
<p>status = CONDITION(procname)</p>	<p>the access to process statuses.</p> <p>The process name must be declared by the statement EXTERNAL (procname).</p> <p>Function values</p> <ul style="list-style-type: none"> -1 process not found 1 process is running 2 process idle 3 process ready 4 process has reached breakpoint (BAPS plus) 5 process stopped (e.g. at Emergency-Stop) 11 1 sub-process active 12 2 sub-processes active 13 3 sub-processes active 14 4 sub-processes active 15 5 sub-processes active 10+n n sub-processes active (n = 1 until 90) >100 process error (runtime error) <p>error code see manual "Status messages and warnings"</p>
<p>status = CONDITION('I0.0') status = CONDITION('I0')</p> <p>status = CONDITION(txtconst)</p> <p>status = CONDITION(txtbyconst)</p> <p>PROGRAM_END</p>	<p>the access to PLC signals</p> <p>There are two programming possibilities:</p> <p>CONDITION('I0.0') The corresponding address is directly inserted as parameter between inverted commas.</p> <p>CONSTANT: txtconst = 'I85.7' CONDITION(txtconst) The desired address is assigned indirectly by means of the text constant txtconst.</p> <p>The same applies to an input byte.</p> <p>Function values</p> <ul style="list-style-type: none"> 0/1 signal condition 0 resp. 1 0..255 signal condition 0 to 255 when reading a byte -1 invalid signal group -2 invalid bit address -3 invalid character -4 invalid signal address

Standard functions

Extension of the standard function Condition

Both standard sub-programs PLC_Process and PLC_time are allowed as arguments in the extension.

The return values of the standard function Condition with the use of both standard sub-programs are defined as follows:

- 0 Program module not active
- 1 Program module active

Syntax of the implied declaration

```
Condition=INTEGER: CONDITION(argument) ;Standard function
```

argument = file variable or interface or name or text expression

name = external_hp_name or PLC_process (parameter) or
PLC_time (parameter)



parameter = constant_expression

In the syntax above:

- name:
External main program name, PLC_process or PLC_time. PLC_process and PLC_time require the entry of the desired PLC program component as parameter.
- parameter:
Constant expression, number of the desired PLC program module.

Standard functions

Standard function ASSIGN

Syntax	Description
PROGRAM new_name FILE: fivar TEXT: finame, oneline, channel_var BEGIN	<p>The ASSIGN statement permits to change the names of files or the assignment of the standard channels (V24_1 to V24_4, SER_1 to SER_4, WIN_1 to WIN_4, PHG) at program runtime by means of the user program.</p> <p> The ASSIGN statement can only be used for closed files and standard channels.</p>
WRITE PHG, CLS WRITE PHG, 'file name:' READ PHG, finame	Clear PHG display.
ASSIGN fivar,finame READ_BEGIN datvar READ fivar, oneline WRITE PHG, oneline CLOSE datvar	<p>Assignment of the content of a text variable finame to the file variable fivar.</p> <p>The following read access to the file is made with the name read in the variable finame.</p>
channel_var = 'V24_2.' ASSIGN PHG, channel_var WRITE PHG, oneline ASSIGN PHG, 'PHG.' PROGRAM_END	<p>Assignment of the output channel PHG to the serial interface V24_2 by means of the text variable channel_var.</p> <p>Output of a line to channel V24_2.</p> <p>Reassign the channel PHG to the PHG.</p> <p> What is important, is the character '.' as end identifier for the channel name. If no point follows as last character, the assignment will be made to a file name (extension .dat), e. g. PHG.dat.</p>

Standard functions


Standard function SIZEOF

Syntax	Description
PROGRAM size POINT: EndPos	The function returns the storage requirement of the indicated type or the variable in bytes. The standard function Size_of can be used within the constant definition part.
BEGIN	
WRITE 'REAL-size:', Size_of(REAL) WRITE 'POINT-size:', Size_of(EndPos)	Output: 4 Output: 16 (4 axes * size of REAL)
PROGRAM_END	

Standard function READ PLC, WRITE PLC

Apart from the rho4 interface, the rho4 has an own data channel which is able to transfer larger data amounts from or to the PLC or software PLC. This data channel is addressed via the standard channel PLC.

The implicit declaration of the PLC channel of the type BNR_FILE enables to transfer variables and expressions of any type. The data are sent as binaries, i.e. unformatted. There is no ASCII conversion as it is usual for DAT files or channels. For the realization of the data buffer to the PLC, it is recommended to represent it through a record type variable.

-  **Since the name PLC is an implicitly declared BAPS standard variable, there is no need to define it by the user. It can simply be used, such as e. g. V24_1.**
As it is the case for all BAPS standard variables, the user can create a variable with the name PLC. It covers the standard variable PLC. It can no longer be addressed.

For the communication with the PLC, the following must be observed:

- only one channel is supported for the communication (standard channel PLC).
- The number of the data module must be contained in the data buffer to the PLC.
- A function module that transfers the data in the corresponding module (depending on the number of the module in the data buffer) is made available. If this function module is to be used, the length of the buffer must be entered in the first component of the data buffer and the number of the desired data component in the second component (both entries as INTEGER).

Standard functions

- The communication buffer consists of the control informations Length and PLC_Dm_No, as well as PLC data (Write/Read to/from PLC)
Only 512 Byte (128*4) maximal can be exchanged between control and PLC by one Write/Read instruction. The data module registered in PLC_Dm_No must exist in the PLC project.

Example: communication with the PLC

Syntax	Description
PROGRAM coupIPLC	Program name
TYPE:	Type arrangement of the data buffer to the PLC
tPLCdata=RECORD	
INTEGER: length	Length of the data buffer
INTEGER: PLC_dmno	Number of the data module to the PLC
INTEGER: PLCcode	Function code from the PLC
REAL: beltpos	Current belt position
POINT: actpos	Current position of the robot in WC
RECORD_END	
tPLCdata: PLCData	Declaration of the data buffer
BELT: 501: belt_1	Declaration of the belt
BEGIN	

Standard functions

Syntax	Description
PLCdata.length= SIZEOF (PLCdata)	Determine extent of the data buffer by means of the standard function Sizeof
PLCdata.PLC_dmno = 1	Number of data module
PLCdata.PLCcode = 0	Initialize function code
PLCdata.beltpos=belt_1	Current belt position
PLCdata.actpos= POS	Current actual position
WRITE PLC, PLCdata	Transfer to the PLC
READ PLC, PLCdata	Get modified data from the PLC
IF PLCdata.PLCcode=0 THEN WRITE 'Everything OK!' ELSE BEGIN WRITE 'Error!' HALT END	Check function code
PROGRAM_END	


Movements and speeds

7 Movements and speeds

7.1 Movement instructions

Syntax	Description
REF_PNT (3) REF_PNT (1,2,3,...)	Referencing The axes which are to approach the reference point simultaneously are indicated in brackets. This instruction should be used as first instruction in the INIT program. The preset kinematic is moved.
REF_PNT ma_1 (3) REF_PNT ma_2 (1,2,3,...)	Referencing with indication of kinematic.
MOVE pickpos MOVE @pickpos	Absolute travel By means of this move instruction, the robot is travelled in world or machine coordinates. The preset kinematic is moved.
MOVE ma_2 pickpos MOVE ma_1 @pickpos	Absolute travel with indication of kinematic.
MOVE pick + up MOVE @pick + @up	As absolute travel the position is, however, computed first then the machine travels.
MOVE ma_2 pick + up MOVE ma_1 @pick + @up	As absolute travel with indication of kinematic the position is, however, computed first, then the machine travels.
MOVE_REL down MOVE_REL @down	Incremental travel The robot travels a distance (in WC) from the current position. The robot travels an angular path (in JC) from the current position. The preset kinematic is moved.
MOVE_REL ma_2 down MOVE_REL ma_2 @down	As before, however, with indication of kinematic.
MOVE_REL down + distance_z MOVE_REL @down + @distance_z	As incremental travel the position is, however, computed first, then the machine travels. The preset kinematic is moved.
MOVE_REL ma_2 down + distance_z MOVE_REL ma_2 @down + @distance_z	As before, however, with indication of kinematic.
MOVE TO pos1 MOVE_REL EXACT path	Exact approach of points (with short axis stop).
MOVE p1, p2, p3	Exact approach of points (with short axis stop).

Movements and speeds

Syntax	Description
MOVE VIA @palpos MOVE_REL APPROX home	Approach of points without exact stop.
MOVE UNTIL force = 1 TO hole_5	<p>Movement break Travel in direction of hole_5 until hole_5 is reached or the condition force = 1 has been met.</p> <p> Only one interrogation is possible with a MOVE UNTIL instruction.</p> <p>Example for a wrong instruction:</p> <p>MOVE UNTIL force=1 AND part_here=1 TO hole_5</p> <p>MOVE UNTIL force=1 AND part_here=1 would be wrong The compiler would interpret this instruction as follows: MOVE UNTIL force= ((1 AND part_here)=1)</p>
MOVE UNTIL force = 1 ERROR pause TO hole_5	<p>Movement break Travel in direction of hole_5 until the condition force = 1 has been met. When the destination position has been reached and the condition has not been met, the ERROR instruction is executed.</p>
MOVE PTP @pos_door MOVE_REL PTP palpos	Point-to-point travel in machine or world coordinates. The control travels synchronously PTP, i. e. all axes start together and arrive at the same time at the programmed point.
MOVE LINEAR TO a_z MOVE_REL LINEAR APPROX up	<p>Linear interpolation The points are, as far as permitted by the workspace, connected by an exact straight line (path).</p>
MOVE CIRCULAR (hi_pt,end_pt) MOVE_REL CIRCULAR (@pos1,@pos2)	<p>Circular interpolation With the circular interpolation, circular arcs can be moved. Always one point pair is required to enable the control to compute the arc of the circle.</p>

Movements and speeds

7.2 Speeds / accelerations

Syntax	Description
V_PTP = 80% V_PTP = 0.8	Point-to-point speed The global PTP speed is assigned in a block. It is active until it is overwritten by another one. The preset kinematic is moved.
ma_1.V_PTP = 80% ma_2.V_PTP = 0.8	The global PTP speed is written in a block with indication of the kinematic. It is active until it is overwritten by another one. It only acts on the indicated kinematic.
V = 800	Path speed The global path speed has an influence on the linear speed and circular movements of the preset kinematic.
ma_1.V = 800	The global path speed with indication of kinematic has an influence on the linear speed and on the circular movements of the indicated kinematic.
A = 765	Path acceleration The path acceleration has only an influence on linear speed and circular movements, and not on PTP speed movements.
ma_1.A = 765	The path acceleration has only an influence on linear speed and circular movements of the indicated kinematic, and not on PTP speed movements.
MOVE PTP WITH V_PTP = 30% TO home	Local point-to-point speed The local PTP speed only is only effective in a MOVE resp. MOVE_REL block. After that, only the global speed acts.
MOVE ma_2 PTP WITH V_PTP = 30% TO home	The local point-to-point speed only is only effective in a MOVE, resp. MOVE_REL block of the indicated kinematic. After that, only the global speed acts.
MOVE LINEAR WITH V = 2000 TO edge	Local path speed The local path speed only is only effective in this block. The indication is in mm/s.
MOVE ma_2 LINEAR WITH V = 2000 TO edge	The local path speed for the indicated kinematic only is only effective in this block. The indication is in mm/s.
MOVE PTP WITH VFIX_PTP = 0.8 TO @pos_top	Local point-to-point speed Acts blockwise, without changing the speed factor, i. e. VFACTOR = 100 %.
MOVE ma_1 PTP WITH VFIX_PTP = 0.8 TO @pos_top	Acts blockwise on the indicated kinematic without changing the speed factor, i. e. VFACTOR = 100 %.

Movements and speeds

Syntax	Description
MOVE LINEAR WITH VFIX = 300 VIA pal_pos_bottom	Local path speed Acts blockwise without changing the speed factor, i. e. VFACTOR = 100 %.
MOVE ma_2 LINEAR WITH VFIX = 300 VIA pal_pos_bottom	Acts blockwise on the indicated kinematic without changing the speed factor, i. e. VFACTOR = 100 %.
AFIX = 300	Path acceleration without effect on the acceleration factor (AFACTOR = 100 %). This statement, too, is only possible locally in the MOVE instruction for LINEAR and CIRCULAR, analog to VFIX.
MOVE WITH T = 2.35 TO destpos	Time default in seconds The next position is to be approached within a specific time. ☞ Only local programming is possible.
MOVE ma_2 WITH T = 2.35 TO destpos	The next position of a kinematic is to be approached within a specific time. ☞ Only local programming is possible.


Movements and speeds

7.3 V/A/D-factors

Syntax	Description
VFACTOR = 100% VFACTOR = 1	Speed factor With the VFACTOR, the point-to-point speed and the path speed can be influenced. The VFACTOR can also be changed via the PHG (acts on the preset kinematic).
ma_1.VFACTOR = 100% ma_2.VFACTOR = 1	With the VFACTOR the point-to-point speed and the path speed of the indicated kinematic can be influenced. The VFACTOR can also be changed via the PHG.
AFACTOR = 100% AFACTOR = 1 DFACTOR = 100% DFACTOR = 1	Acceleration and deceleration factors These factors influence the path accelerations and the stored point-to-point speed axis accelerations.
ma_1.AFACTOR = 100% ma_2.DFACTOR = 1	Influence of the path accelerations and the stored point-to-point speed axis accelerations of the indicated kinematic.

Movements and speeds

7.4 Slope speed types


Syntax	Description
PROGR_SLOPE	Global switch-over to the program slope, i. e. several coherent movements are executed without changing the speed or acceleration between the individual movements. Only after a MOVE TO statement, delays resp. accelerations are permitted again within a movement sequence. The MOVE TO statement only acts on the preset kinematic.
;;KINEMATICS = ma_2 PROGR_SLOPE	If another kinematic than the preset kinematic is to be switched it has to be set at first as default.
BLOCK_SLOPE	Global switch-over to the block slope. Also between MOVE VIA blocks a deceleration is carried out before a programmed point.
;;KINEMATICS = ma_2 BLOCK_SLOPE	Switch-over to the block slope of the indicated kinematic.  See manual machine parameters.

Belt-synchronous

8 Belt-synchronous

The synchronization statements ensure that the controlled machine takes the correct position relative to the belt, as far as position and orientation are concerned. In doing so, the belt can move forward and backward respectively change its speed or stop.

The belt must be in a straight line, it can be located anywhere in the tool.

Syntax	Description
ma_1.BELT: 501 = belt_kin1	Declaration of the belt variable Several belts can be declared for one kinematic. The belt names must be different. The same belt can be used for several kinematics.  The number of belts is stored in machine parameter 501.
SYNC belt_kin1 >= 10.0 SYNC belt_kin1, light = 1	The belt counter is reset. The reset can depend on the current value of the belt variable itself or from a condition (see special function 28).
SYNCHRON ma_1 belt_kin1	With the SYNCHRON instruction, the belt synchronization is switched on. From now on, the programmed movements are synchronized with the belt, as far as position and orientation with the belt are concerned. The indication of the kinematic is optional.
SYNCHRON_END ma_1 belt_kin1	With the SYNCHRON_END instruction, the belt synchronization is switched off. The indication of the kinematic is optional. If no kinematic is indicated the pre-set kinematic will be used.

 **See also special function 28, belt_set!**

Belt-synchronous

Notes:

Workspace limitation

9 Workspace limitation

Syntax	Description
LIMIT_MIN = (x_value,...,u_value) LIMIT_MAX = (x_value,...,u_value) LIMIT_OFF	Workspace limitation with the minimum and maximum values. These values are always world coordinate points. Disable of workspace limitation.
ma_1.LIMIT_MIN = (x_value,...,u_value) ma_1.LIMIT_MAX = (x_value,...,u_value) LIMIT_OFF ma_1	Workspace limitation with indication of the kinematic.

 **Only the position of the end point is monitored!**

Workspace limitation


Notes:

Write/read functions

10 Write/read functions

10.1 Interfaces

The WRITE/READ operations are made in BAPS3 on so-called logical devices which are assigned specific transfer protocols via default and machine parameters. The assignment to a physical interface is also made via machine parameters. It is thus possible to use the interfaces available on the control in a flexible way.







Syntax	Description
PHG	Serial interface Firmly assigned to the PHG.
V24_1 V24_2 V24_3 V24_4	Serial interfaces Logic and physical assignment, see default via PHG resp. machine parameters.
WIN_1 WIN_2 WIN_3 WIN_4 PLC	Standard channels for the communication between rho4 and Windows programs. The communication is made via TCP/IP protocol, the establishment of the connection is made automatically.  The data for WIN_x and PLC are transferred binary. The reading timeout for WIN_x and PLC channels is infinite.

For the communication, various communication protocols are available, which can be selected via machine parameter default values or via mode 9.1 on the PHG.



P.-No.	Protocol establishment	Reading echo
1 a 1 b	<DATA> followed by <CR><LF> <DATA> followed by <CR> or <LF>	yes
2	<DATA>	yes
3	<SOH><STX><DATA><ETX> followed by <SOH><STX><CR><LF><ETX>	no
4	<SOH><STX><DATA><ETX>	no
5	<DATA>	no
6	PHG protocol	yes
7	rho1/rho2 compatible after P.No. 3	no
8	Data back-up shift according to Siemens protocol 3964/R	no

1a = data input, 1b = data output

Write/read functions

Syntax	Description
WRITE V24_1,POS WRITE PHG, 'number', i	In the BAPS program it is possible to write texts and variables onto the interfaces V24_1 to V24_4, SER_1 to SER_4, WIN_1 to WIN_4 and the PHG.
ARRAY[1..70] CHAR:F1_CHAR WRITE F1_CHAR ARRAY[1..4] ARRAY[1..50] CHAR:F2_CHAR WRITE F2_CHAR [1]	<p>In the BAPS program it is possible to write texts, variables and 1-dimensional arrays onto the interfaces V24_1 to V24_4, SER_1 to SER_4, WIN_1 to WIN_4 and the PHG.</p> <p> If a communication has to take place from a BAPS program via the interface through which the ROPS4 communicates with the rho4, the coupling has to be deactivated first.</p> <p>Writes the first line (50 characters) onto the PHG.</p>
WRITE PHG, '<-[J', ... resp. WRITE PHG, CLS, ...	Delete PHG display  '<-' corresponds to control character of ESC
WRITE PHG, '<-[K', ...	Delete until end of line on the PHG display.  '<-' corresponds to control character of ESC
WRITE PHG, '<-[line;columnH', ...	Position cursor on line and column of the PHG display. The H is required as final character of the column.  '<-' corresponds to control character of ESC
READ M READ V24_2, M	In the BAPS program, texts and variables can be read from the interfaces V24_1 to V24_4 and from the PHG.
READ WIN_3, vision_data	Read data via TCP/IP on WIN channel 3.  The data are transferred binary.
WRITE WIN_3, meas_cycle	Put out message via TCP/IP on WIN channel 3.  The data are transferred binary.


Write/read functions

Syntax	Description
READ_BEGIN PHG	<p>When calling this instruction, the CONDITION of the indicated device is set to zero. Possible file variables are PHG (default) or the interfaces V24_1 to V24_4, SER_1 to SER_4, WIN_1 to WIN_4.</p> <p> The output of a line number possible for files is not evaluated in this case.</p>
WRITE_BEGIN PHG	<p>If an output is prepared on the PHG in a preceding statement, it will be deleted by this instruction with the abort of the process. Possible file variables are PHG (default) or the interfaces V24_1 to V24_4, SER_1 to SER_4, WIN_1 to WIN_4.</p> <p> The output of a line number possible for files is not evaluated in this case.</p>

Write/read functions




10.2 I/O files

Read

Syntax	Description
FILE:finadat	Type declaration for ASCII files, which are to be accessed by reading and/or writing within a BAPS program, e. g. finadat.dat.
BNR_FILE:file1	Type declaration of a BINARY file.
FILE:finadat READ_BEGIN finadat, 5	Positioning of the READ marker to the beginning of a defined line of the default file. In this example this would be the fifth line in the file finadat.dat. If the line indication is deleted, the positioning will be made to line 1.  Before the first READ instruction it is in any case required to program a READ_BEGIN. READ_BEGIN opens the file.
FILE:finadat INTEGER:otto READ_BEGIN finadat READ finadat, otto	The first value is read from the file (finadat.dat), starting with the positioned line and stored in the indicated variable (otto). The invisible READ marker is advanced to the next value.
FILE:finadat IF FILE_END (finadat) THEN JUMP finished finished: ...	The end of a file (finadat.dat) can be inquired.

Write/read functions

Write

Syntax	Description
FILE: finadat INTEGER: number number = 1 WRITE_BEGIN finadat, 1 WRITE_BEGIN finadat, number	Causes the jump of the WRITE marker to the beginning of a specific line in the indicated file, finadat.dat.  The data from this position up to the end of the file is deleted.  Before the first WRITE or WRITE_END instruction it is in any case required to program a WRITE_BEGIN. Both WRITE instructions open an existing file or create a new one if it does not exist.
FILE: finadat REAL: erg TEXT: display display = 'the result:' WRITE_BEGIN danadat, 5 WRITE finadat, display, res	The contents of the variables are written into the indicated file. The writing place is the position on which the WRITE marker is placed.  The remaining data from this line will be deleted.
FILE: finadat WRITE_END finadat	The WRITE marker is placed to the end of the file. With the next WRITE instruction, the file is extended at its end.
FILE: finadat CLOSE finadat	A .dat file must always be closed after the writing so that it can be read. At the end of the program, the .dat file is closed automatically.

Write/read functions

Notes:

Special functions

11 Special functions

✉ **Not all described special functions are suggestive resp. practicable for each rho4 controllable kinematics. If you have a question, please apply to the technical support:**

mailto: Mounting-Handling-RC.brc@boschrexroth.de
Phone: +49 (0) 60 62 / 78-0

Syntax	Description
SPC_FCT:	Functions to which no BAPS key words are assigned, are declared as special functions. In the following, all special functions are listed consecutively according to their number.
SPC_FCT:1 = ioI(VALUE INTEGER: ioI_no VALUE INTEGER: kin_no VALUE INTEGER: coord_no VALUE REAL: outp_pos VALUE REAL: para_outp VALUE INTEGER: time_offset)	Accurate position switching of digital signals on the path with time offset. With this function, up to 16 channels can be used, see manual rho4 'Control functions'.
SPC_FCT:2 = ppo(VALUE INTEGER: ppo_no VALUE INTEGER: kin_no VALUE INTEGER: coord_no VALUE REAL: outp_pos VALUE REAL: param VALUE INTEGER: offset)	Accurate position output of process parameters which are put out with time offset. Special function 2 contains 16 functions for the output of 8-bit wide outputs (value range 0 to 255) or of any decimal value at the interface of the control, see manual rho4 'Control functions'
SPC_FCT:3 = mach_pos (VALUE INTEGER: kin_no kin_name.JC_POINT: @p_name) SPC_FCT:3 = mach_pos (VALUE INTEGER: kin_no JC_POINT: @p_name)	This special function can only be used with drives with CANrho interface. With special function 3, the internal machine position is set to the values indicated in the point variable @p_name. The special function must be declared separately for each kinematic to which the machine positions are to be assigned. If no kinematic is indicated when declaring the special function, the point refers to the preset kinematic, see manual rho4 'Control functions'.

Syntax	Description
SPC_FCT:4 = instruction (VALUE INTEGER: op_ident TEXT: source,targe tINTEGER: status)	With special function 4, operating system functions (instructions) of the rho4 can be activated out of BAPS3 user processes, see manual rho4 'Control functions'

Special functions

instruction =	Name of the SPC_FCT, through which it is called
op_ident =	Indicates which operating system instruction is to be executed: 1 = Compile (QLL-file) 2 = COPY (any file) 3 = DELETE (any file) 4 = START (user process) 5 = STOP (active user process)
source, target =	contain the parameters required for the corresponding positions
status =	provides the result of the executed function in form of an INTEGER number: 0 = no error, instruction correctly processed -1 = forbidden instruction transmitted to SPC_FCT -2 = Error in target file name during copying process or error in the name of the file to be deleted or error in the name of the file to be compiled -3 = File identification during compilation is not QLL -4 = Error in the QLL file name -5 = File to be deleted or target file during copying still open -6 = Error during copying, e.g. not enough memory -7 = Compiler already active when being called -8 = Error during compilation, .qll file not existing >0 = Sum of all compilation errors and compilation warnings or numbers of the error that has occurred at the start of the indicated process: -1052 User process already existing -1092 Selected program does not exist

Special functions


- 1073 Program can only run as external subroutine
- 1057 File already too often open
- 1085 present point file is not allowed for the selected process
- 1112 selected IRD file or the corresponding PNT file cannot be sequentialized because there is not enough memory
- 1079 Sub-process tries to stop its own main process

Syntax	Description
SPC_FCT:15 = belt_param (VALUE INTEGER: belt_no VALUE REAL: v_blt_stopped VALUE REAL: v_blt_runs VALUE INTEGER: time_ms)	Function for the parameterization of the belt characteristics. It is required to be able to adapt the synchronization type of the specific characteristic to the conveying equipment.


Syntax	Description
SPC_FCT:16 = pnt_select (text: pnt_fina text: pnt_ptna INTEGER: ret_code)	By means of this special function, a specific .pkt file can be selected for the operating modes Define, Teach In and Print, which is offered directly, i. e. without pressing any key, when selecting the corresponding operating mode. For the operating modes Define and Teach In, a point name can additionally be preselected within the selected point file.

Syntax	Description
SPC_FCT:17 = mirror (VALUE BINARY: X_INV, Y_INV, Z_INV) mirror (1,1,0) MOVE @(50,120,-25.21)	The axes 1 and 2 are mirrored in this example.
SPC_FCT:21 = belt_kind (VALUE INTEGER: belt_no VALUE INTEGER: kind_belt)	With this special function, the desired type of belt synchronization can be selected. The following synchronizations are at present realized: kind_belt = 1: belt synchronization with belt-parallel travelling of the kinematic kind_belt = 2: belt synchronization without belt-parallel travelling of the kinematic kind_belt = 3: cam disk interpolation.

Special functions

Syntax	Description
SPC_FCT:23 = clock_date (INTEGER: hours, minutes, day, month, year)	Access to the clock time and the date of the rho4 system clock.
SPC_FCT:24 = sys_time (INTEGER: start_time)	Access to an internal system counter (real time counter).
SPC_FCT:27 = wc_mainr (VALUE INTEGER: kin_no)	Due to the coordinate transformation, overlappings of the world coordinate angles in WC can occur. The special function 27 serves the elimination of these overlappings.
SPC_FCT:28 = set_belt (VALUE INTEGER: belt_no VALUE REAL: reset_value)	When calling the special function 28, the value of the variable reset_value is assigned to the internal belt reset value of the corresponding belt. The internal belt counter is reset to the reset value by means of the belt counter signal (SYNC).
SPC_FCT:29 = pnt_store_on (VALUE INTEGER: kin_no VALUE INTEGER: ms_no VALUE REAL: distance)	With special function 29, reading of referencing path, the storage of reference points is switched on. Both, machine and world coordinates, are stored. kin_no = kinematic whose coordinate value is to be stored ms_no = measuring system number of axis, belt or analog signal to which the parameter distance refers distance = distance after which, if exceeded, the corresponding value is stored  With this special function, it is additionally possible to store points in the time raster of the transformation clock. If this function mode is desired, the option byte 36 has to be set to 1. The default is storing after tolerance default, i. e. option byte 36 = 0.
SPC_FCT:30 = pnt_stor_off (VALUE INTEGER: kin_no INTEGER: no_points)	With special function 30, the storage of reference points is switched off. The stored values are maintained until the special function 29 is called again with the same kinematic number. kin_no = number of the kinematic whose values are no longer to be stored no_points = number of the values stored since the call of special function 29

Special functions

Syntax	Description
SPC_FCT:31 = pnt_sto_read (VALUE INTEGER: kin_no VALUE INTEGER: index JC_POINT: @position POINT: position)	<p>With special function 31, the stored reference point values can be read in machine coordinates and in world coordinates. The data are located in a ring memory.</p> <p>kin_no = indicates the kinematic whose coordinate values are returned. The coordinate values of the belts of this kinematic are also available.</p> <p>index = indicates which ring memory entry is read.</p> <p>@position = here, the machine coordinate values indicated under kin_no are located.</p> <p>position = here, the world coordinate values indicated under kin_no are located.</p>
SPC_FCT:43 = probe_on (VALUE INTEGER: kin_no VALUE INTEGER: edge INPUT: channel_no)	<p>With special function 43, the actual position can be stored with an achievable reaction time of 0.01 ms. The measured value is written into the standard variable @MPOS.</p> <p>The second parameter edge indicates with which edge (positive or negative) the actual position is to be stored. Via the channel number it is possible to make an inquiry, whether a measurement has been made.</p> <p>kin_no: kinematic number edge: positive edge = 0, negative edge = 1 channel_no: no measuring is made = 0, measuring is made = 1</p> <p> The probe input only works with incremental transmitters. The basis for the probe inputs is 600.</p>
SPC_FCT:44 = probe_off (VALUE INTEGER: kin_no INPUT: channel_no)	<p>With special function 44, the already activated special function 43 can be disabled again.</p> <p>kin_no: kinematic number channel_no: input number of probe</p>

Special functions

Syntax	Description
<p>SPC_FCT:45 = move_file (VALUE INTEGER: kin_no BNR_FILE: curve_x VALUE INTEGER: base JC_POINT: @mod_flag VALUE ARRAY[1..6]INTEGER:reserve)</p>	<p>When calling special function 45, move_file, the course of a curve stored in a binary file is travelled by reading the values from the file and putting them out unchanged as position values in the interpolation raster resp. in the position controller raster of the servo-board clock.</p> <p>kin_no: indicates the kinematic to be travelled curve_x: name of the binary file to be travelled basis: indicates whether the set position values of the file are to be put out in the interpolation raster or in the position controller raster. 1 = interpolation raster 2 = position controller raster</p> <p>mod_flag: the modulo flag is required as identification for endless axis. For other axis types, this value has to be preoccupied with 0.0.</p> <p>The following decimal values are admissible as identification 0.0 = no modulo computation 1.0 = modulo computation at the end of the block. The modulo value is the last axis value in the binary file. 2.0 = modulo computation at the end of the block. The modulo value is specified machine parameter value P311.</p> <p>reserve: this parameter has been provided for possible expansions in future operating system versions.</p>
<p>SPC_FCT:46 = block_advance (VALUE INTEGER: kin_no VALUE INTEGER: number)</p>	<p>With special function 46, reduce block advance in MOVE blocks, the internal block advance can be reduced.</p> <p>The use of this special function is recommended in combination with special function 45 and the component of asynchronous inputs. Especially when using asynchronous inputs it can be necessary to reduce the block advance to permit a faster effect of input changes onto the prepared blocks.</p> <p>kin_no: indicates the kinematic to be travelled</p> <p>number: in this parameter, an integral number in the value range from 1 to 11 can be transferred to the special function. This number represents an upper limit for the number of kinematic blocks to be prepared. The value is set to 1 internally for values <1 and to 11 for values >11.</p>

Special functions

Syntax	Description
<p>SPC_FCT:47 = exc_define (VALUE INTEGER: exc_no VALUE INTEGER: err_code_1 VALUE INTEGER: err_code_2 TEXT: exc_pr_name INTEGER: state)</p>	<p>With the special function 47 exc_define, it is possible to define 32 exceptions.</p> <p>Exceptions enable the programmable reaction to status messages in user processes. Without manual intervention of the user, some exceptions can be recognized and eliminated through specific starting of exception processes. Programmable exceptions are status messages of the belt type 4 and messages that are recognized in the block preparation.</p> <p>An error code (err_code_1) or alternately an error code area (err_code_1 ... err_code_2) is used as a trigger condition (event). If the condition is met, i.e. a process reaches a state with the declared error code, the exception process is automatically started within the rho4 operating system.</p> <p>exc_no: exception number (permissible numbers 1..32)</p> <p>err_code_1: error code of the exception, resp. error code range lowest limit "0" means: exception get deleted</p> <p>err_code_2: error code range highest limit "0" means: only err_code_1 is effective</p> <p>exc_pr_name: program name of exception process (max. 8 character) which is started automatically when a process state message by err_code_1 resp. err_code_1 ..err_code_2 occur</p> <p>state: Return code of function (0: no Error, <0: Error, >0: warning)</p> <p>"0" : Exception was defined "+1" : Warning: Exception IRD-program not available Exception was defined "-1" : Error in definition: inadmissible exception number Cause: wrong exc_no (permissible is 1..32) "-2" : Error in definition: inadmissible err_code</p>

Special functions

Syntax	Description
<pre> SPC_FCT:48 = exc_detect (VALUE INTEGER : err_code_1 VALUE INTEGER: err_code_2 INTEGER: num_err_proc ARRAY [1..max_err_proc] TEXT: err_pr_name ARRAY [1..max_err_proc] INTEGER: err_sub_no ARRAY [1..max_err_proc] INTEGER: pr_err_code INTEGER: state) </pre>	<p>With the special function 48 <code>exc_detect</code> it is possible to recognize processes that have reached a state defined by <code>SPC_FCT 47</code>.</p> <p>Entry parameters of the function are the error codes <code>err_code_1</code> and <code>err_code_2</code> used in <code>exc_define</code>. The function <code>exc_detect</code> returns the number and the name of the processes that have reached these error states. In the rho4 operating system, up to 16 incorrect processes will be saved. <code>exc_detect</code> should be called within an exception process. The process names <code>err_pr_name[n]</code> returned by the function can then be used to stop and start the incorrect processes.</p> <p>The constant <code>max_err_proc=16</code> is declared in the BAPS-Include-File rmain.inc.</p> <p><code>err_code_1</code>: error code of the exception, resp. error code range lowest limit</p> <p><code>err_code_2</code>: error code range highest limit. "0" means: only <code>err_code_1</code> (no limit declaration) is effective</p> <p><code>num_err_proc</code>: number of faulty processes</p> <p><code>err_pr_name</code>: program name (max. 8 character) of faulty processes with error code <code>err_code_1 .. err_code_2</code>. Up to 16 faulty processes can be stored</p> <p><code>err_sub_no</code>: if an error occur in a parallel branch of a process, the number of the parallel branch is indicated. If the error occur in the main process, <code>err_sub_no</code> is "0"</p> <p><code>pr_err_code</code>: returns the occurred process error code by limit declaration</p> <p><code>state</code>: Return code of function (0: no error, <0: error) = 0 : Exception was defined = -2 : Inadmissible <code>err_code</code></p>

Special functions

Syntax	Description
<p>SPC_FCT:51 = belt_calc (VALUE INTEGER: Belt_No VALUE REAL: Calc_Value)</p>	<p>From version VO05, the user interface of the belt synchronous section of belt type 4 is made more user friendly (see SPC_FCT: 53 BeltRange (...) in manual 'rho4 Control functions').</p> <p>From version VO05 the special function 51 is no more available.</p> <p>From version VO05 the call of special function 51 leads to the runtime message "Inadmissible SPC_FCT", Code 131840</p>
<p>SPC_FCT:52 = belt_ptp_fac VALUE INTEGER: Belt_No JC_POINT: Min_Ptp_Fac JC_POINT: @Max_Ptp_Fac)</p>	<p>The special function 52 is used for reading and writing of the axis geometry factors for the automatic velocity adjustment of PTP movements.</p> <p>In PTP movements, the velocities for the axes are deduced via factors from the machine parameters. In belt-synchronous movements, an axis can additionally be boosted or weakened through the belt movement.</p> <p>The amplitude of this effect depends on two application depending factors:</p> <ul style="list-style-type: none"> ● the actual belt velocity ● Position of the belt-synchronous working area in the working area of the robot <p>This effect does not need to be taken into account if the maximum belt velocity is small compared with the maximum robot velocity.</p> <p>Belt_No: Number of the belt for which the value is to be set (1 to 16) or read (-1 to -16)</p> <p>@Min_Ptp_Fac: Minimum axis geometry factor</p> <p>@Max_Ptp_Fac: Maximum axis geometry factor</p> <p>The axis geometry factors indicate how many degrees or mm an axis moves when the belt moves 1 mm. They can be positive or negative. They are determined automatically by the call of SPC_FCT 53 = BeltRange (...)</p> <p>The belt components must be set to value 0.0</p> <p>See manual rho4 'Control functions'.</p>

Special functions

Syntax	Description
<p>SPC_FCT:53 = BeltRange (VALUE INTEGER: BeltNo JC_POINT: @Basis REAL: FullLength REAL: BeginLength)</p>	<p>Special function 53 enables the description of the geometric position of the belt range with beltkind 4.</p> <p>BeltNo: Number of the belt, for which the value is to be set (1..16). If the belt number is negative (-1 .. -16), the value is read</p> <p>@Basis: JC_POINT of the begin of the belt-synchronous working area</p> <p>FullLength: Full length of the belt-synchronous working area in belt direction. The value must be greater than zero</p> <p>BeginLength: Latest begin length with belt-synchronous working. The value must be greater than zero and smaller or equal FullLength</p>
<p>SPC_FCT:54 = Belt_V (VALUE INTEGER : BeltNo REAL: BeltVel)</p>	<p>Special function 54 enables the questioning of the actual belt velocity.</p> <p>Note: The questioning is made at the moment of the preparation. So it is not synchronized with the working off of preceding movements.</p> <p>BeltNo: Number of the belt, for which the actual belt velocity is to be read (1..16)</p> <p>BeltVel: actual belt velocity</p>
<p>SPC_FCT:55 = Belt_V_Sim (VALUE INTEGER : BeltNo VALUE REAL: BeltVel)</p>	<p>With special function 55 it is possible to change temporary the actual belt simulation velocity (machine parameter P508). This function makes it more easier to test belt-synchronous applications with different belt velocities.</p> <p>BeltNo: Number of the belt, for which the belt simulation velocity is to be set (1..16)</p> <p>BeltVel: New belt simulation velocity in mm/sec. Value must be greater or equal zero. With value -1.0, the machine parameter value is reactivated</p>

Special functions

Syntax	Description
<p>SPC_FCT:56=IOLx(VALUE INTEGER: IONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: DelayTime)</p>	<p>This special function offers the possibility to switch on a beltsynchronous linear or circular path.</p> <p>IONo = Number of the IO-logic (1..16). 16 RC outputs are available in the interface, which can be switched with special function 56. These are the PLC addresses I36.0 to I37.7, cur. no. 288 to 303 in the signal descriptions manual.</p> <p>KinNo: Number of kinematic to which the switch position is referring.</p> <p>CoordNo: Number of coordinate within the kinematic to which the switch position is referring.</p> <p>SwitchPos: Workpiece corresponding world coordinate point (if necessary including belt coordinates), at which it is switched i.e. the parameter value is put out. For example, the switching position can be a teach point.</p> <p>ParamValue: The parameter value is preset as decimal value. A value of < 0.5 is put out as logic 0 (low) and a value of ≥ 0.5 as logic 1 (high).</p> <p>DelayTime: Time in [ms] which elapses between the actual output of the parameter value and the reaching of the switch position. This means that the digital output is already switched before having reached the desired position. With that, reaction times of the periphery can be compensated.</p> <p>Note: Because of input parameter <i>SwitchPos</i> is a world coordinate point, special function 56 must be defined seperately for each existing kinematic.</p>

Special functions

Syntax	Description
<p>SPC_FCT:57=PPOx(VALUE INTEGER: PPONo VALUE INTEGER: KinNo VALUE INTEGER: CoordNo VALUE REAL: SwitchPos VALUE REAL: ParamValue VALUE INTEGER: DelayTime)</p>	<p>This special function offers the possibility to switch on a beltsynchronous linear or circular path.</p> <p>PPONo = Number of the PPO logic (1..16). No. of the byte output which is switched.</p> <p>In the interface, 16 byte outputs (each 8 bits wide) are available which can be switched with special function 57. These are the PLC addresses I118.0 to I133.7, cur. no. 944 to 1071 in the signal descriptions manual.</p> <p>KinNo: Number of kinematic to which the switch position is referring.</p> <p>CoordNo: Number of coordinate within the kinematic to which the switch position is referring.</p> <p>SwitchPos: Workpiece corresponding world coordinate point (if necessary including belt coordinates), at which it is switched i.e. the parameter value is put out. For example, the switching position can be a teach point.</p> <p>ParamValue: The parameter value is preset as decimal value. If the output channel is used as byte output (PPO No. = 1 to 16), a value range of 0 to 255 is available, but also the corresponding parity and strobe signals (PLC addr. I114.0 to I117.7, no. 912 to 943) must be controlled as for the BAPS INTEGER outputs.</p> <p>DelayTime: Time in [ms] which elapses between the actual output of the parameter value and the reaching of the switch position. This means that the decimal output is already switched before having reached the desired position. With that, reaction times of the periphery can be compensated.</p> <p>Note: Because of input parameter <i>SwitchPos</i> is a world coordinate point, special function 57 must be defined separately for each existing kinematic.</p>

Library functions

12 Library functions

Function declaration

The declaration of the rho4 library functions is made similar to the special functions via a function declaration part named RHO_FCT.

Syntax	Description
RHO_FCT:2031 = rKGAxPos (TKGAxPos:PKGAxPos)	RHO_FCT:<FCT number> = <FCT name> (<record type>:<record parameter>)

Function call in the BAPS program


The following line in the BAPS program carries out a rho4 library function call

Syntax	Description
Return_Code = rKGAxPos (PKGAxPos)	<Return code> = <FCT name> (<record variable>)

Each function sends back a return code containing the error messages or warnings. The return code is a variable of the type INTEGER that is declared in the user program. If processed correctly, the function sends back a 0 as return code.

0	to	-4999	Return messages of rho4 library function.
-5000	to	-5999	Error messages of the TCP/IP communication.
>10000			Error messages of the MS-C library.

The function parameters are transferred as BAPS record variables. The record variable must be declared in the user program. As type, only the record type of the function from the associated INCLUDE file (*.inc) must be used. This file also contains an exact description of the individual components of the record types.

 **The required parameters can be found in the manual rho4 DLL library.**

 **RHO-FCT can be used without assigning a return value to a variable, just as subroutines!**

Syntax	Description
rKGAxPos(PKGAxPos)	

Library functions

Notes:

Fix files

13 Fix files

13.1 EXPROG.DAT

External program selection with EXPROG.DAT. In the file EXPROG.DAT, the program numbers are assigned to the program names (ird).

Syntax	Description
00 = init 01 = pal1 . FF = initpos	

13.2 MSD.DAT

Machine status display. In the file MSD.DAT, the text numbers are assigned to the texts. These controlled texts can be displayed via the PHG in modes 7, 12, 1/2.

Syntax	Description
01 = pusher front . 99 = end msd	

13.3 TEXT.DAT

Control of texts via the interface. In the file TEXT.DAT, the text numbers are assigned to the texts.

Syntax	Description
00 = text1 . FF = text256	

Fix files

13.4 TOOLS.DAT


The TOOL instruction is used to be able to use different grippers or tools during a sequence of processes.

Until the call of TOOL, the values stored in machine parameter 309 are valid.

The individual gripper parameters are stored in the TOOLS.DAT file. For the general description, three translations (G_X, G_Y, G_Z) and three rotations (G_D1, G_D2, G_D3) are used, which refer to the flange each.

Example for the file TOOLS.DAT


GRIPPER_1	=	10	2.5	5	1	2	3	IC- grippers No. 1 and No. 2
GRIPPER_2	=	-20	0	120	5	0	6	
BUNDLE5_F_L	=	-50	-50	200	0	0	0	Bundle gripper No. 5 front left
BUNDLE5_F_R	=	-50	50	200	0	0	0	Bundle gripper No. 5 front right
BUNDLE5_B_L	=	50	-50	200	0	0	0	Bundle gripper No. 5 behind left
BUNDLE5_B_R=50	=	50	50	200	0	0	0	Bundle gripper No. 5 behind right
OFF	=	0	0	0	0	0	0	Dummy gripper for switching off

 **The order of the individual coordinates (G_X, G_Y to G_D3) has to be maintained in any case. For missing values, zeros must be inserted. The gripper name must start in the first column and its length must not have more than 12 characters.**

Syntax	Description
TOOL ma_1 gripper_1	The selection of a gripper in the BAPS program is made via the TOOL instruction. In the coordinate system of kinematic ma_1, the gripper coordinate system gripper_1 is loaded from the file.
TOOL ma_1 off	The instruction tool_end does not exist, but it can be realized via a gripper coordinate system filled up with zeros (see example above).

BAPS3 keywords

14 BAPS3 keywords

 Hereafter, all currently reserved language symbols for the BAPS3 are listed. The listed language symbols must not be used as variables, file names or sub-program names in a BAPS3 program.

German	English
@	@
ALLE	EVERY
ANFANG	BEGIN
ANSONSTEN	DEFAULT
AUSGANG	OUTPUT
BAND	BELT
BINAER	BINARY
BIS	UNTIL
CIRCA	APPROX
DANN	THEN
DATEI	FILE
DEF	DEF
DEZ	REAL
EINGANG	INPUT
ENDE	END
EXAKT	EXACT
EXKLUSIV_ENDE	EXCLUSIVE_END
EXKLUSIV	EXCLUSIVE
EXTERN	EXTERNAL
FAHRE	MOVE
FALLS	CASE
FALLS_ENDE	CASE_END
FEHLER	ERROR
FELD	ARRAY
GANZ	INTEGER
GLEICH	EQUAL
GLOBAL	PUBLIC
GRENZE_AUS	LIMIT_OFF
HALT	HALT
KONSTANTE	CONST

BAPS3 keywords

German	English
KREIS	CIRCULAR
LESE	READ
LESE_ANFANG	READ_BEGIN
LINEAR	LINEAR
MAL	TIMES
MAX_ZEIT	MAX_TIME
MIT	WITH
MK_PUNKT	JC_POINT
MOD	MOD
NACH	TO
NICHT	NOT
ODER	OR
PARALLEL	PARALLEL
PARALLEL_ENDE	PARALLEL_END
PAUSE	PAUSE
PERMANENT	PERMANENT
PRIO	PRIO
PROGR_SLOPE	PROGR_SLOPE
PROGRAMM_ENDE	PROGRAM_END
PROGRAMM	PROGRAM
PTP	PTP
PUNKT	POINT
REF_PKT	REF_PNT
RHO_FKT	RHO_FCT
RK_RAHMEN	WC_FRAME
RSPRUNG	RETURN
SATZ_SLOPE	BLOCK_SLOPE
SCHLIESSE	CLOSE
SCHREIBE	WRITE
SCHREIBE_ANF	WRITE_BEGIN
SCHREIBE_ENDE	WRITE_END
SEMAPHOR	SEMAPHORE
SONST	ELSE
SOWIE	ALSO
SPRUNG	JUMP
SPZ_FKT	SPC_FCT

BAPS3 keywords

German	English
START	START
STOP	STOP
SYNC	SYNC
SYNCHRON	SYNCHRON
SYNCHRON_ENDE	SYNCHRON_END
TEXT	TEXT
TYP	TYPE
UEBER	VIA
UND	AND
UNTERBRECHE	BREAK
UP	SUBROUTINE
UP_ENDE	SUB_END
VAR	VAR
VERBUND	RECORD
VERBUND_ENDE	RECORD_END
VERSCHIEBE	MOVE_REL
WARTE	WAIT
WDH	REPEAT
WDH_ENDE	REPEAT_END
WENN	IF
WERKZEUG	TOOL
WERT	VALUE
ZEICHEN	CHAR
ZUORDNE	ASSIGN

BAPS3 keywords

BAPS3 Translator statements

German	English
ACHSNAMEN	JC_NAMES
DATEI_FEHLER	FILE_ERROR
EINFUEGE	INCLUDE
FEHLER	ERROR
INT	INT
KINEMATIK	KINEMATICS
KOORDINATEN	WC_NAMES
PROZESS_ART	PROCESS_KIND
SER_EA_STOP	SER_IO_STOP
STEUERUNG	CONTROL
TESTINFO	DEBUGINFO
WARNUNG	WARNING
WERK_KOORD	POSE

BAPS3 keywords

BAPS3 standard variables

German	English
@IPOS	@POS
@MPOS	@MPOS
A	A
AFAKTOR	AFACTOR
AFEST	AFIX
DFAKTOR	DFACTOR
GRENZE_MAX	LIMIT_MAX
GRENZE_MIN	LIMIT_MIN
HBG	MCP
IPOS	POS
PHG	PHG
R	R
R_PTP	R_PTP
RK_SYSTEM	WC_SYSTEM
SER_1	SER_1
SER_2	SER_2
SER_3	SER_3
SER_4	SER_4
SPS	PLC
T	T
TFEST	TFIX
TTY	TTY
V	V
V_PTP	V_PTP
V24_1	V24_1
V24_2	V24_2
V24_3	V24_3
V24_4	V24_4
VFAKTOR	VFACTOR
VFEST	VFIX
WIN_1	WIN_1
WIN_2	WIN_2
WIN_3	WIN_3
WIN_4	WIN_4

BAPS3 keywords

BAPS3 standard functions

German	English
ABS	ABS
ATAN	ATAN
BNR_DATEI	BNR_FILE
CHR	CHR
COS	COS
DATEI_ENDE	END_OF_FILE
GANZ_ZFELD	INT_ASC
GANZTEIL	TRUNC
GROESSE_VON	SIZEOF
MK	JC
ORD	ORD
RK	WC
RK_RECHNUNG	WC_CALCULATION
RUNDUNG	ROUND
SIN	SIN
SPS_PROZESS	PLC_PROCESS
SPS_ZEIT	PLC_TIME
UNTERBRECHE	BREAK
WURZEL	SQRT
ZFELD_GANZ	ASC_INT
ZUSTAND	CONDITION

BAPS3 standard constants

German	English
CLS	CLS
RK_UR	WC_UR
VERSION	VERSION

BAPS3 keywords

General BAPS3 keyword list

German	English
@	@
@IPOS	@POS
@MPOS	@MPOS
A	A
ABS	ABS
ACHSNAMEN	JC_NAMES
AFAKTOR	AFACTOR
AFEST	AFIX
ALLE	EVERY
ANFANG	BEGIN
ANSONSTEN	DEFAULT
ATAN	ATAN
AUSGANG	OUTPUT
BAND	BELT
BINAER	BINARY
BIS	UNTIL
BNR_DATEI	BNR_FILE
CHR	CHR
CIRCA	APPROX
CLS	CLS
COS	COS
DANN	THEN
DATEI	FILE
DATEI_FEHLER	FILE_ERROR
DATEI_ENDE	END_OF_FILE
DEF	DEF
DEZ	REAL
DFAKTOR	DFACTOR
EINFUEGE	INCLUDE
EINGANG	INPUT
ENDE	END
EXAKT	EXACT
EXKLUSIV_ENDE	EXCLUSIVE_END
EXKLUSIV	EXCLUSIVE
EXTERN	EXTERNAL

BAPS3 keywords

German	English
FAHRE	MOVE
FALLS	CASE
FALLS_ENDE	CASE_END
FEHLER	ERROR
FELD	ARRAY
GANZ	INTEGER
GANZ_ZFELD	INT_ASC
GANZTEIL	TRUNC
GLEICH	EQUAL
GLOBAL	PUBLIC
GRENZE_AUS	LIMIT_OFF
GRENZE_MAX	LIMIT_MAX
GRENZE_MIN	LIMIT_MIN
GROESSE_VON	SIZEOF
HALT	HALT
HBG	MCP
INT	INT
IPOS	POS
KINEMATIK	KINEMATICS
KONSTANTE	CONST
KOORDINATEN	WC_NAMES
KREIS	CIRCULAR
LESE	READ
LESE_ANFANG	READ_BEGIN
LINEAR	LINEAR
MAL	TIMES
MAX_ZEIT	MAX_TIME
MIT	WITH
MK	JC
MK_PUNKT	JC_POINT
MOD	MOD
NACH	TO
NICHT	NOT
ODER	OR
ORD	ORD
PARALLEL	PARALLEL

BAPS3 keywords

German	English
PARALLEL_ENDE	PARALLEL_END
PAUSE	PAUSE
PERMANENT	PERMANENT
PHG	PHG
PRIO	PRIO
PROGR_SLOPE	PROGR_SLOPE
PROGRAMM_ENDE	PROGRAM_END
PROGRAMM	PROGRAM
PROZESS_ART	PROCESS_KIND
PTP	PTP
PUNKT	POINT
R	R
R_PTP	R_PTP
REF_PKT	REF_PNT
RHO_FKT	RHO_FCT
RK	WC
RK_RAHMEN	WC_FRAME
RK_RECHNUNG	WC_CALCULATION
RK_SYSTEM	WC_SYSTEM
RK_UR	WC_UR
RSPRUNG	RETURN
RUNDUNG	ROUND
SATZ_SLOPE	BLOCK_SLOPE
SCHLIESSE	CLOSE
SCHREIBE	WRITE
SCHREIBE_ANF	WRITE_BEGIN
SCHREIBE_ENDE	WRITE_END
SEMAPHOR	SEMAPHORE
SER_1	SER_1
SER_2	SER_2
SER_3	SER_3
SER_4	SER_4
SER_EA_STOP	SER_IO_STOP
SIN	SIN
SONST	ELSE
SOWIE	ALSO

BAPS3 keywords

German	English
SPRUNG	JUMP
SPS	PLC
SPS_PROZESS	PLC_PROCESS
SPS_ZEIT	PLC_TIME
SPZ_FKT	SPC_FCT
START	START
STOP	STOP
STEUERUNG	CONTROL
SYNC	SYNC
SYNCHRON	SYNCHRON
SYNCHRON_ENDE	SYNCHRON_END
T	T
TESTINFO	DEBUGINFO
TEXT	TEXT
TFEST	TFIX
TTY	TTY
TYP	TYPE
UEBER	VIA
UND	AND
UNTERBRECHE	BREAK
UP	SUBROUTINE
UP_ENDE	SUB_END
V	V
V_PTP	V_PTP
V24_1	V24_1
V24_2	V24_2
V24_3	V24_3
V24_4	V24_4
VAR	VAR
VERBUND	RECORD
VERBUND_ENDE	RECORD_END
VERSCHIEBE	MOVE_REL
VERSION	VERSION
VFAKTOR	VFACTOR
VFEST	VFIX
WARNUNG	WARNING

BAPS3 keywords

German	English
WARTE	WAIT
WDH	REPEAT
WDH_ENDE	REPEAT_END
WENN	IF
WERK_KOORD	POSE
WERKZEUG	TOOL
WERT	VALUE
WIN_1	WIN_1
WIN_2	WIN_2
WIN_3	WIN_3
WIN_4	WIN_4
WURZEL	SQRT
ZEICHEN	CHAR
ZFELD_GANZ	ASC_INT
ZUORDNE	ASSIGN
ZUSTAND	CONDITION

BAPS3 keywords

Notes:

Appendix

A Appendix

A.1 Abbreviations

Abbreviation	Meaning
BAPS3	Programming language; Bewegungs- und Ablaufprogrammiersprache, Version 3; programming language
C:	Hard disk drive
CAN	Controler Area Network
DAC	Digital-analog converter
EEPROM	Electronically erasable programmable read-only memory
EGB	Elektrostatic sensitive components
ESD	Electrostatic discharge
LF	Line feed
MPP	Machine parameter program
MSD	Machine state display
PCL	Memory-programmable control
PE	Protective earth
PHG	Hand-held programming unit
POS	Actual position
PTP	Point to point
RC	Robot control
ROD	Incremental encoder
RPM	Rounds per minute
ROPS4	Robot programming system for rho4
TCP	Tool center point
WC	World coordinates

Appendix

A.2 Index**Symbols**

@, 5-2, 7-1
@MPOS, 5-3
@POS, 5-3
+, 5-1, 7-1
-, 5-1
*, 5-1
/, 5-1
=, 5-1
<, 5-1
<=, 5-1
<>, 5-1
>, 5-1
>=, 5-1

A

A, 7-3
ABS, 6-1
AFACTOR, 7-5
AFIX, 7-4
ALSO, 4-3
AND, 5-1
APPROX, 7-2
ARRAY, 3-8, 3-9, 10-2
ARRAY BINARY, 3-8
ARRAY CHAR, 3-8
ARRAY INPUT, 3-9
ARRAY INPUT INTEGER, 3-9
ARRAY INPUT REAL, 3-9
ARRAY INTEGER, 3-8
ARRAY JC_POINT, 3-8
ARRAY OUTPUT, 3-9
ARRAY OUTPUT INTEGER, 3-9
ARRAY OUTPUT REAL, 3-9
ARRAY POINT, 3-8
ARRAY REAL, 3-8
ARRAY TEXT, 3-8
ARRAY, two-dimensional, 3-8
ASC_INT, 6-3
ASSIGN, 6-9
asynchronous inputs, 3-10
ATAN, 5-1
axis value assignment, 5-2

B

BEGIN, 2-4, 2-5, 4-2
BELT, 3-10, 8-1
BINARY, 3-2
binary file, 10-4
BLOCK_SLOPE, 7-6

C

CASE, 4-2
CASE_END, 4-2
channel numbers, 3-10
CHAR, 3-2, 10-2
CHR, 6-1
CIRCULAR, 7-2
CLOSE, 10-5
CLS, 3-1
CONDITION, 6-6
CONSTANT, 5-2
CONTROL, 2-1, 2-2
COS, 5-1

D

DEF, 3-5
DEF ARRAY JC_POINT, 3-5
DEF ARRAY POINT, 3-5
DEF JC_POINT, 3-5
DEF POINT, 3-5
DEFAULT, 4-2
DFACTOR, 7-5
Documentation, 1-7

E

ELSE, 4-2
EMC Directive, 1-1
EMERGENCY-STOP devices, 1-5
END, 4-2
EQUAL, 4-2
ERROR, 2-2, 4-1, 7-2
ESD
 Electrostatic discharge, 1-6
 grounding, 1-6
 workplace, 1-6
ESD-sensitive components, 1-6
EXACT, 7-1
EXCLUSIVE, 3-6, 4-4
EXKLUSIVE_END, 3-6
EXPROG.DAT, 13-1
EXTERNAL, 2-4, 3-6

F

FILE, 3-2, 10-4
FILE_END, 10-4
Floppy disk drive, 1-7

G

GLOBAL, 3-6
Grounding bracelet, 1-6

Appendix

H

HALT, 2-4
Hard disk drive, 1-7

I

IF, 4-2
IF THEN ELSE, 4-2
INCLUDE, 2-3
INPUT BINARY, 3-10
INPUT INTEGER, 3-10
INT, 2-3
INT_ASC, 6-2
INTEGER, 3-2, 5-2
IOLx, 11-11

J

JC, 6-5
JC_NAMES, 2-2
JC_POINT, 3-2
JUMP, 4-2

K

KINEMATIC, 2-2

L

LIMIT_DIS, 9-1
LIMIT_MAX, 9-1
LIMIT_MIN, 9-1
LIMIT_OFF, 9-1
LINEAR, 7-2
Low-Voltage Directive, 1-1

M

MAX_TIME, 4-1
MOD, 5-1
Modules sensitive to electrostatic discharge. See
 ESD-sensitive components
MOVE, 7-1
MOVE_REL, 7-1

N

NOT, 5-1

O

OR, 5-1
ORD, 6-1

OUTPUT BINARY, 3-10
OUTPUT INTEGER, 3-10

P

PARALLEL, 4-3
PARALLEL_END, 4-3
PAUSE, 4-1
PERMANENT, 2-2
PHG, 10-2
POINT, 3-2
point declarations, 3-5
POS, 5-3, 10-2
PPOx, 11-12
PRIO, 4-3
PROCESS_KIND, 2-2
PROGR_SLOPE, 7-6
PROGRAM, 2-4
PROGRAM_END, 2-4
PTP, 7-2

Q

Qualified personnel, 1-2

R

R, 6-5
R_PTP, 6-5
READ, 10-2
READ_BEGIN, 10-4
REAL, 3-2, 5-2
RECORD, 3-4
REF_PNT, 7-1
Release, 1-8
REPEAT, 4-1
REPEAT_END, 4-1
RETURN, 2-5
ROUND, 6-1

S

Safety instructions, 1-4
Safety markings, 1-3
SEMAPHORE, 3-2, 3-6
SER_IO_STOP, 2-3
serial interfaces, 10-1
SIN, 5-1
Spare parts, 1-6

Appendix

SPC_FCT, 11-1

- 1 = accurate position switching, 11-1
- 15 = belt_param, 11-3
- 16 = pnt_select, 11-3
- 17 = mirror, 11-3
- 2 = accurate position parameter output, 11-1
- 21 = belt kind selection, 11-3
- 23 = system clock time and date, 11-4
- 24 = system clock time, 11-4
- 27 = WC_MAINR, 11-4
- 28 = belt counter setting, 11-4
- 29 = reference path reading "ON", 11-4
- 3 = machine parameter setting, 11-1
- 30 = reference path reading "OFF", 11-4
- 31 = reference point value reading, 11-5
- 4 = instruction, 11-1
- 43 = quick measuring ON, 11-5
- 44 = quick measuring OFF, 11-5
- 45 = MOVE_FILE, 11-6
- 46 = reduced block advance, 11-6
- 47 = Exception-Handling exc_define, 11-7
- 48 = Exception-Handling exc_detect, 11-8
- 51 = Belt_calc, 11-9
- 52 = Automatic velocity adjustment for PTP movements, 11-9
- 53 = Beltkind 4 BeltRange, 11-10
- 54 = Beltkind 4 Belt_V, 11-10
- 55 = Beltkind 4 Belt_V_Sim, 11-10

SQRT, 5-1

Standard operation, 1-1

START, 4-3

STOP, 4-3

SUB_END, 2-5

SUBROUTINE, 2-5

SYNC, 8-1

SYNCHRON, 8-1

SYNCHRON_END, 8-1

T

T, 7-4

Test activities, 1-5

TEXT, 3-2

THEN, 4-2

TIMES, 4-1

TO, 7-1

TOOL, 13-2

Trademarks, 1-8

TRUNC, 6-1

type definition part, 3-4

types, 3-3

U

UNTIL, 7-2

V

V, 7-3

V_PTP, 7-3

V24_1, 10-2

VALUE, 2-4

VERSION, 3-1

VFACTOR, 7-5

VFIX, 7-4

VFIX_PTP, 7-3

VIA, 7-2

W

WAIT, 4-1

WAIT UNTIL, 4-1

WAIT UNTIL MAX_TIME, 4-1

WARNING, 2-2

WC, 6-5

WC_NAMES, 2-2

WITH, 7-3

WRITE, 10-2

WRITE_BEGIN, 10-5

WRITE_END, 10-5

Bosch Rexroth AG
Electric Drives and Controls
P.O. Box 13 57
97803 Lohr, Germany
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr, Germany
Phone +49 (0)93 52-40-50 60
Fax +49 (0)93 52-40-49 41
service.svc@boschrexroth.de
www.boschrexroth.com

