Rexroth
Bosch Group

Rexroth Rho 4
Control functions

1070072179
Edition 07

Software manual

Title Rexroth Rho 4
Control functions

Type of Documentation Software manual

Document Typecode DOK-RHO*4*-STEUERFUNK*-PR07-EN-P

Purpose of Documentation The present manual informs about:

- The available functions of the rho4

Record of Revisions

Description	Release Date	Notes
DOK-RHO*4*-STEUERFUNK*-PR07-EN-P	10.2003	Valid from VO07

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)

Contents

Contents

Contents

Contents

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 technicans. 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 provides information about the available functions of the rho4.

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
Application IndraControl VEH 30	1070 170 330	1070 170 331
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:.**

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

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

Safety Instructions

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.).

Survey of special functions

2 Survey of special functions

With the 'special functions', special functions available in control rho4 for which no BAPS3 language elements are reserved, are rendered accessible for the BAPS3 programmer.

Special functions are an expansion of the scope of the BAPS3 language. They can be called in a program if they are activated as option in the software of your control and have been declared before calling, similar to a variable.

☞ **Not all here 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

Survey of special functions

In the rho4, the following special functions are available at the moment:

Fct. No.	Brief function description
1	Accurate position switching of digital outputs on the path
2	Accurate position switching of decimal outputs on the path
3	Setting the machine position
4	Calling operating system functions
15	Parameterization of the belt characteristic
16	Selection of point file
17	mirror
21	Belt kind
23	System date and time
24	System counter
27	WC main range
28	Setting the belt counter
29	Switching on recording of reference path
30	Switching off recording of reference path
31	Reading of reference path values
43	Flying measuring ON, only available for rho4.1
44	Flying measuring OFF, only available for rho4.1
45	MOVE_FILE, travel movement of curve course from .bnr file
46	Setting the block preparation
47	Define exception
48	Detect exception
51	Setting of the belt counter reset value (from version VO05 no more available)
52	Velocity adjustment for PTP movements
53	Belt synchronous working area belt kind 4
54	Current belt speed
55	Change the belt simulation speed
56	Accurate beltsynchronous position switching of digital outputs on the path
57	Accurate beltsynchronous position switching of decimal outputs on the path

Survey of special functions

Declaration of the special functions

The declaration of a special function contains identification number and name of the function as well as names and data types of the transfer parameters. With the transfer parameters, it is possible to define when, where and how the function should be active. The declaration must be done in the declaration part of the program.

The name of the special function and the names of the transfer parameters can there be freely selected.

☞ **The data types are definitely determined by the specification of the respective special function.**

Example
SPC_FCT: x = example (VALUE REAL: wap)

SPC_FCT: x	Function number
example	Special function name
VALUE REAL:	Designation of data type of transfer parameter, e.g. decimal value
wap	Names of the transfer parameter (place holder)

Call of the special functions

The call in the declaration part of the program is carried out by indication of the special function name and definition of the declared transfer parameters.

In the call, the name of the special function as well as the types of the transfer parameters must be maintained in the same way in which they have been determined in the declaration of the program.

General example
example (15.5)

Meaning

example	Special function name
(15.5)	Definition of the transfer parameter

Survey of special functions

Notes:

3 Accurate position switching

General

The rho4 offers the possibility to switch on a path. So it is possible to trigger external peripheral devices during a travel movement from a BAPS program with accurate positioning and with a delay time.

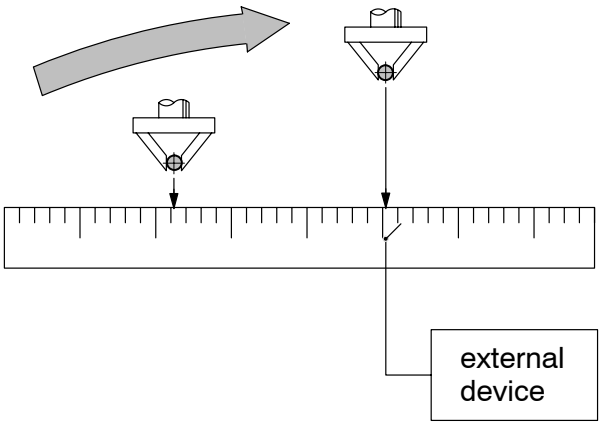
For that purpose, two special functions are available. With special function 1, digital outputs can be switched at the interface. Special function 1 is called IO logic (input, output logic) in the following.

With special function 2, integer values of 0 to 255 can be put out. Special function 2 is called PPO logic (process parameter output logic) in the following.

With these values you are able to control your technology, e.g. dosing the colour quantityfor lacquering applications or switching on and off of pistol.

Accurate position output

Accurate position output means that in the program, a definite position of the tool center point (gripper position) is determined at which the external device is approached per interface.



Programming

The special functions must be declared, similar to variables, before being called, in the declaration part of the BAPS program. It must be ensured in any case that the parameter list is in exact conformance with the following declaration.

Accurate position switching

3.1 Accurate position switching of digital outputs on the path

The declaration of the special function contains the identification number 1 and the special function designation, as well as the names and the type designation of the function values by which is determined – in the call of the special function – at which position of the corresponding axis which control value is to be put out.

```
SPC_FCT:1=IOL(VALUE INTEGER: IONo
              VALUE INTEGER: KinNo
              VALUE INTEGER: CoordNo
              VALUE REAL:    SwitchPos
              VALUE REAL:    ParamValue
              VALUE INTEGER: DelayTime)
```

Meaning

IONo = 1 to 16	Number of the IO-logic. 16 RC outputs are available in the interface, which can be switched with special function 1. These are the PLC addresses I36.0 to I37.7, cur. no. 288 to 303 in the signal descriptions manual.
KinNo	Number of kinematic to which the switch position is referring.
CoordNo	Number of coordinate within the kinematic to which the switch position is referring.
SwitchPos	Position at which it is switched i.e. the parameter value is put out.
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).
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.

Accurate position switching

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

3.2 Accurate beltsynchronous position switching of digital outputs on the path

This special function offers the possibility to switch on a beltsynchronous linear or circular path.The declaration of the special function contains the identification number 56 and the special function designation, as well as the names and the type designation of the function values by which is determined – in the call of the special function – at which position of the corresponding axis which control value is to be put out.

```
SPC_FCT:56=IOLx(VALUE INTEGER: IONo
                VALUE INTEGER: KinNo
                VALUE INTEGER: CoordNo
                VALUE REAL:    SwitchPos
                VALUE REAL:    ParamValue
                VALUE INTEGER: DelayTime)
```

Accurate position switching

Meaning

IONo = 1 to 16	Number of the IO-logic. 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.
KinNo	Number of kinematic to which the switch position is referring.
CoordNo	Number of coordinate within the kinematic to which the switch position is referring.
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.
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).
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.

Note:

Because of input parameter *SwitchPos* is a world coordinate point, special function 56 must be defined seperately for each existing kinematic.

Example of declaration:

In a control there exist the two kinematics *Scara* and *Portal*. The special function 56 can be defined for each kinematics as follows:

```
SPC_FCT:56=IOLx1(VALUE INTEGER: IONo
                 VALUE INTEGER: KinNo
                 VALUE INTEGER: CoordNo
                 VALUE REAL:    SwitchPos
                 VALUE REAL:    ParamValue
                 VALUE INTEGER: DelayTime)

SPC_FCT:56=IOLx2(VALUE INTEGER: IONo
                 VALUE INTEGER: KinNo
                 VALUE INTEGER: CoordNo
                 VALUE REAL:    SwitchPos
                 VALUE REAL:    ParamValue
                 VALUE INTEGER: DelayTime)
```

Accurate position switching

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

3.3 Accurate position switching of decimal outputs on the path

```
SPC_FCT:2=PPO(VALUE INTEGER: PPONo
              VALUE INTEGER: KinNo
              VALUE INTEGER: CoordNo
              VALUE REAL:    SwitchPos
              VALUE REAL:    ParamValue
              VALUE INTEGER: DelayTime)
```

Meaning

PPONo = 1 to 16	Number of the PPO logic. No. of the byte output which is switched. In the interface, 16 byte outputs (each 8 bits wide) are available which can be switched with special function 2. These are the PLC addresses I118.0 to I133.7, cur. no. 944 to 1071 in the signal descriptions manual.
KinNo	Number of kinematic to which the switch position is referring.
CoordNo	Number of coordinate within the kinematic to which the switch position is referring.
SwitchPos	Position at which it is switched i.e. the parameter value is put out.
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.
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 output is already switched before reaching the desired position. With that, reaction times of the periphery can be compensated.

Accurate position switching

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Accurate position switching

3.4 Accurate beltsynchronous position switching of decimal outputs on the path

```
SPC_FCT:57=PPOx(VALUE INTEGER: PPONo
                VALUE INTEGER: KinNo
                VALUE INTEGER: CoordNo
                VALUE REAL:    SwitchPos
                VALUE REAL:    ParamValue
                VALUE INTEGER: DelayTime)
```

Meaning

PPONo = 1 to 16	Number of the PPO logic. No. of the byte output which is switched. 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.
KinNo	Number of kinematic to which the switch position is referring.
CoordNo	Number of coordinate within the kinematic to which the switch position is referring.
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.
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.
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 output is already switched before reaching the desired position. With that, reaction times of the periphery can be compensated.

Note:

Because of input parameter *SwitchPos* is a world coordinate point, special function 57 must be defined seperately for each existing kinematic.

Example of declaration:

Accurate position switching

In a control there exist the two kinematics *Scara* and *Portal*. The special function 57 can be defined for each kinematics as follows:

```
SPC_FCT:57=PPOx1(VALUE INTEGER: PPONo
                 VALUE INTEGER: KinNo
                 VALUE INTEGER: CoordNo
                 VALUE REAL:    SwitchPos
                 VALUE REAL:    ParamValue
                 VALUE INTEGER: DelayTime)

SPC_FCT:57=PPOx2(VALUE INTEGER: PPONo
                 VALUE INTEGER: KinNo
                 VALUE INTEGER: CoordNo
                 VALUE REAL:    SwitchPos
                 VALUE REAL:    ParamValue
                 VALUE INTEGER: DelayTime)
```

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Accurate position switching

3.5 Example

We look at a control with 2 kinematics, SCARA is the name of the second kinematic. With the IO No = 13 a pistol is switched and with the PPO No = 5 a colour volume. With the BAPS syntax

```
(1) MOVE SCARA LINEAR TO (200,200,100,0)
(2) IOL (13, 2, 3, 250,1, 125)
(3) IOL (13, 2, 3, 562, 0, 0)
(4) PPO ( 5, 2, 3, 250, 135, 0)
(5) MOVE SCARA LINEAR TO (300,300,700,0)
```

the following behaviour is achieved:

1 The travel block gets the SCARA into a defined output position.

2 With the block, the control is told that the pistol (= IO No 13) is switched on, 125 ms before the Z axis (third coordinate) of the SCARA (second kinematic) reaches the position 250 mm, i.e. the digital output I37.4 is set to ’1’.

3 The pistol is switched off again when the Z axis reaches the position 562 mm, i.e. the output I37.4 is set to ’0’.

4 As colour volume (= PPO-no. 5) the value ’135’ is output, when the Z axis has reached the 250 mm position. This means that the value ’10000111’ = ’135’ is digitally issued at the byte outputs I122.0 to 122.7.

5 During the travel movement of this block, all three switching procedures are carried out at the progammed positions. The PPO and IO functions are afterwards freely available again.

Accurate position switching

3.6 Messages/warnings

During runtime IONo/PPONo, KinNo, CoordNo and ParamValue are checked for validity in the block preparation. For these parameters only, the control is able to detect a faulty input. Switch position and time offset cannot be checked. In case of a wrong input, one of the following messages is displayed correspondingly:

PPO/IOL: inadm.Fct.No:	IONo	≠ 1 to 16, PPONo ≠ 1 to 16
PPO/IOL: inadm.Kin.No:	KinNo	≠ 1 to number of kinematics
PPO/IOL: inadm.Coord.:	CoordNo	≠ 1 to number of axes
PPO/IOL: inadm.Fct (P2)	Function number	≠ 1 or 2
PPO: inadm. Param.Value:	ParamValue	≠ 0 to 255

With the time offset, reaction times of the process periphery can be compensated. The switching condition is fulfilled by the time offset before the programmed switching position. If the programmed time offset is too long so that the correct interpolation direction, i.e. with travel movement in the direction of the programmed switching position, is too late, it is switched directly and one of the following warnings is displayed:

IOL On:	rate–t too l.
IOL Off:	rate–t too l.
PPO:	rate–t too l.

You can call the special functions 1 and 2 as often as you want in the BAPS program. If a specific PPO or IO function number is programmed again and the last programmed switching condition has not yet been met, i.e. the function number is still occupied, the corresponding PPO and IO function is overwritten with the new switching condition. In this case, one of the following warnings is displayed:

IOL On:	Fct. in use
IOL Off:	Fct. in use
PPO:	Fct. in use

Accurate position switching

3.7 Remarks

With special function 1, a switch-on position (ParamValue = 1) and a switch-off position (ParamValue = 0) can be programmed for each IONo simultaneously. If the switch-on and switch-off positions are so close together that the two conditions are fulfilled during one scanning cycle, the output is disabled.

If on one path, the same peripheric device is to be switched several times, the same hardware address is allocated to several PPO and IOL function numbers in the PLC program. If you copy e.g. IO No 1, 2 and 3, i.e. the PLC addr. I36.0, I36.1 and I36.2, all onto the same digital output, you can switch the same output with

```
IOL (1, ..., SwitchPos1, ...)
IOL (2, ..., SwitchPos2, ...)
IOL (3, ..., SwitchPos3, ...)
```

three times on one path. With the PPO functions, you can proceed in the same way for switching several times on a path.

If it is interpolated into the wrong direction, i.e. you are moving away from the programmed switching position, the corresponding PPO or IO function remains active until the condition will be fulfilled later.

IO No. 1 to 16 are not additionally declared in the BAPS program. They do not concern the normal digital BAPS user outputs, PLC addr. 134.0 to 158.6, ser. no. 1072 to 1270. Also the PPO No. 1 to 16 are not additionally declared in the BAPS program. They are positioned on the same addresses in the RC internal interface as in the BAPS INTEGER outputs (= BAPS channel No. 401 to 416). The user must ensure that there are no overlappings, i.e. the same outputs should not be simultaneously used as BAPS channel and as PPO channel.

Speed changes which occur during the time offset, cannot be taken into account. This kind of problem normally only occurs in case of long time offsets.

A specific PPO No. can naturally be switched again, concerning the hardware, if the strobe time has elapsed. With long strobe times, there can possibly occur delays.

Accurate position switching

3.8 Reset behaviour

Resetting the control

If the control is reset via mode 11.1 or via interface (PLC addr. O16.2, ser. no. 130), all PPO and IO functions are deactivated, i.e. there is no active function any longer.

Furthermore, all digital outputs are deleted again that have been set by the special function 1 (IO logic).

Resetting from BAPS

Active PPO or IO functions can be reset from BAPS. This is especially interesting if a movement program has been interrupted by a monitoring process via STOP Proc_x, while a PPO or IO function has been active. For a scheduled reset, the special functions 1 or 2 are called with the corresponding negative function numbers.

The programmed parameter value is immediately put out and the corresponding PPO and IO function number is enabled. So the interrupted program can be restarted without display of the warning '.....: Fct. in use'.

In this case CoordNo, SwitchPos and TimeOffset are ignored. There must only be any admissible values. To avoid control internal delays, the reset call should be effected with the same KinNo. with which the PPO and IO function had been activated.

Example

(1) IOL (13, 2, 1, 1, 0, 0)
(2) IOL (13, 2, 1, 1, 1, 0)
(3) PPO (5, 2, 1, 1, 135, 0)

1 The pistol (= IO No. 13) is switched off immediately, i.e. output I37.4 is set to '0'.
2 The pistol (= IO No. 13) is switched on immediately, i.e. output I37.4 is set to '1'.
3 As colour volume (= PPO No. 5) the value '135' is immediately output, i.e. the byte output I122.0 to 122.7 is set to '10000111' = '135'.

Accurate position switching

Notes:

4 Setting the machine position

☞ **This extension stage is only realized for drives with CANrho interface. For CANopen or Sercos interface, it cannot be used.**

With special function 3, the internal machine positions can be changed from a BAPS3-program. This can be especially useful e.g. for replacing the approach of the reference points.

Control internally, the current machine position of each individual axis (machine coordinate) results from the reading of the measuring system, for incremental transmitters in connection with referencing.

Declaration of the special function

```
SPC_FCT:3=mach_pos(VALUE INTEGER: kin_no
                kin_name.JC_POINT: @p_name)      ;With special function 3, the
                                                 ;internal machine position is set
                                                 ;to the values of the
                                                 ;point variable @p_name

SPC_FCT:3=mach_pos(VALUE INTEGER: kin_no
                         JC_POINT: @p_name)      ;If no kinematic is identified
                                                 ;in the declaration of the special
                                                 ;function, the point refers to the
                                                 ;preset kinematic
```

mach_pos	Freely selectable name of special function
kin_no	Number of the kinematic for which the machine positions are manipulated.
kin_name	Name of the kinematic, resp. preset kinematic, for which the machine position is manipulated.
@p_name	Name of the point which is transferred with the call of the special function. It contains the values (in [mm] or [degrees]) onto which the machine positions of the corresponding kinematic are set.

If in a program, the machine positions of several kinematics are to be manipulated, the special function for each kinematic must be defined individually. There must be different names.

☞ **@p_name also contains belt values that could possibly occur. But these cannot be manipulated.**

Setting the machine position

To ensure that the control takes over immediately the new values in machine as well as in world coordinates, a 'MOVE-instr.' must be programmed in the BAPS program after call of the special function which refers to the same kinematic. This can also be a dummy 'Move-Instr.' which does not cause a movement e.g. MOVE_REL KIN_X (0,...,0). If the 'Move-Instr.' is not programmed, it is possible that under POS resp. axis displays/positions (mode 7.1 at PHG2000) the old positions are displayed in world coordinates.

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

The special function can, as a rule, be called at any point of the BAPS-program. It is, however, especially recommended that the call is carried out directly after booting the control, before any movement has taken place. In this case, the following program example replaces the referencing.

```
;;CONTROL=rho4

;;KINEMATICS:(1=scara, 2=portal)

;;scara.JC_NAMES=SC1,SC2,SC3,SC4

;;scara.WC_NAMES=XSC,YSC,ZSC,OSC

;;portal.JC_NAMES=PO1,PO2,PO3

;;portal.WC_NAMES=XPO,YPO,ZPO

PROGRAM SPC3

SPC_FCT:3=sc_machpos(VALUE INTEGER: kin_no
                     JC_POINT: @machpos)

SPC_FCT:3=po_machpos(VALUE INTEGER: kin_no         ;Preset kinematic
                 portal.JC_POINT: @machpos)
```

Setting the machine position

```
DEF scara.JC_POINT:  @sc_begpos

DEF portal.JC_POINT: @po_begpos

BEGIN

   @sc_begpos=@POS

   @sc_begpos.SC4=@sc_begpos.SC4-360 ;Manipulate Scara machine position of the
                                     ;fourth axis by 360 degrees

   sc_machpos(1,@sc_begpos)

   MOVE_REL scara @(0,0,0,0)         ;Dummy-MOVE-Inst. for take-over of the
                                     ;new values

   @po_begpos=@(100,100,500)

   po_machpos(2,@po_begpos)          ;Set portal machine position

   MOVE_REL portal (0,0,0)           ;Dummy-MOVE-Inst. for take-over of the
                                     ;new values

PROGRAM_END
```

Operation mode of setting machine position

The special function 3 sets the control internal machine position onto the JC-point determined in the transfer parameter. The control does not know later where the machine has actually been before. The mode of operation quasi corresponds to a referencing with a new actual value of the reference point. Under @POS the new JC-position is displayed. With @MPOS the new machine positions can also be obtained.

Simultaneously, the current world coordinate position POS is changed. It results from the forward transformation of the new machine position.

As shown in the example above, it is possible with special function 3 to set the machine positon for all axes simultaneously and also to manipulate individual axes.

The travel range limits adjusted with the machine parameters P202, P203, P204, P205 always refer to the current machine position changed via special function 3.

The machine position set in the program remains active until the next call of the special function. It is not changed by ’basic position’.

After a new boot of the control, no manupulation is active, i.e. those values are valid that are delivered by the measuring system.

A repeated referencing resets the values manipulated with special function 3 to the machine parameter settings.

Setting the machine position

DANGER
If wrong machine positions are activated due to faulty programming, the control no longer knows where the machine actually is.In this case of error, there can be unexpected and pseudo-wrong movements, especially in connection with a coordinate transformation.

It can also happen that the set travel range limits do not react correctly or do not react at all.

DANGER
If a control is operating with special function 3 as well as with the referencing function, you have to ensure that the referencing speed (machine parameter P108) and the first reduced referencing speed (machine parameter P109) are identical. Otherwise, there is the danger that the axis approaches the reference point switch with a too high speed during repeated referencing and possibly finds a wrong marker.

5 Calling operating system functions

With special function 4, operating system functions (commands) of the rho4 can be activated from BAPS3 user processes.

☞ **Special function 4 must be declared in the declaration part of a BAPS program.**

```
SPC_FCT:4=command(VALUE INTEGER: com_ident_no
                       TEXT: source,destination
                    INTEGER: status)
```

command	Name of the special function by which it is called.
com_identifier	Indicates which operating system command is to be carried out. The meaning of the corresponding parameters is to be found in the table 'com_identifier'.
source, destination	Contain the parameters necessary for the respective function.
status	Returns the result of the function carried out, in form of a number ot the type INTEGER. The meaning of the corresponding parameters is to be found in the table 'status'.

com_identifier

1	COMPILE (.qll–file)
2	COPY (any file)
3	DELETE (any file)
4	START (user process)
5	STOP (active user process)

Calling operating system functions

status

0	No errors, command correctly carried out
–1	Incorrect command transferred to special function
–2	Error in the destination file name during copying, error in the name of the file to be deleted or to be compiled
–3	File extension for compilation is not .qll
–4	Error in source file name
–5	The file to be deleted or the destination file still open during copying
–6	Error of copying, e.g. in case of insufficient storage
–7	Compiler is already active at the moment it is called
–8	Error during compilation, .qll file not existing
> 0	Total of all compiler errors or compiler warnings or number of error which has occurred when starting the indicated process.
–1051	User process already exists
–1057	File has already been opened too often
–1073	Program can only be run as external subroutine
–1079	Subprocess tries to stop its main process
–1085	Available .pkt file is not permitted for selected process
–3873	Selected program is not available (.ird file not available)
–1112	No sequentialization of the selected .ird or the respective .pkt file. Not enough memory available.

Compilation of BAPS programs

Call of the special function 4 with com_ident_no=1 calls the BAPS3 compiler in the rho4.

The instruction line within a BAPS program looks like as follows:

command(1,dat_name,temp_name,result)

The file whose name is to be found in the variable 'dat_name', is compiled. The result i.e. 0 or a value unequal 0, is available to evalution for the BAPS program in the variable 'result'. 'dat_name' must contain the file extension .qll.

Furthermore, a corresponding message is displayed by the compiler on the PHG2000 without having to wait for the actuation of a key at the PHG2000.

If the compiler messages are not to be displayed on the PHG2000, the RC-input 227 must be set to 1, PLC addr. O28.3, 'Disable output of system messages to the PHG'.

'temp_name' is not evaluated and not influenced, i.e. 'dat_name' could also be the parameter here.

Copying files

With the com_ident_no=2 files can be copied with special function 4.

The special function checks during run time whether the file extensions of source and destination file are admissible. Error statuses are returned via 'status'.

The instruction line within a BAPS program looks like as follows:
command(2,source_name,destin_name,status)

'source_name' contains the name of the file to be copied.

'destin_name' contains the file name to which is copied. If the destination file is already available, it is deleted before copying.

A message via 'status' is only put out if the destination file is still open for writing or reading.

Admissible file extensions are:

Source	Destination
.qll	.qll, .err, .dat
.dat	.qll, .err, .dat
.pkt	.pkt
.sym	.sym
.ird	.ird

Deleting files

With the com_ident=3 files can be deleted with special function 4. The instruction line within a BAPS program looks like the following:

command (3,dat_name,temp_name,status)

'file_name' contains the name of the file to be deleted.

'temp_name' is not used and not influenced.

'status' supplies the result of the deletion process.

☞ **Files to be deleted must not have been opened by another BAPS process for reading or writing, i.e. they must not yet have been opened!**

Calling operating system functions

Starting processes

With com_ident_no=4, processes can be started with special function 4:

command (4,dat_name,temp_name,status)

’dat_name’ contains the name of the program to be started.

’temp_name’ is not used and not influenced.

’status’ supplies the result of the start process.

Stopping processes

With com_ident_no=5, processes can be stopped with special function 4.

command (5,dat_name,temp_name,status)

’dat_name’ contains the name of the program to be stopped.

’temp_name’ is not used and not influenced.

’status’ supplies the result of the stop process.

Restrictions, error treatment

In special function 4, no plausibility check of the file extensions is carried out when copying files.

Names and file extensions are not checked for validity.

The compiler is not to be called by several processes simultaneously. If this is ignored, error code –7 is returned.

6 Parameterization of the belt characteristic

In the following, those parameters are described which are necessary for adjusting the synchronization type to the specific characteristic of the conveying equipment. Parameterization is carried out via special function 15.

The threshold values for the detection of the belt conditions ’belt stopped’ and ’belt running’ and the time within which the messages are tolerated, are programmed via special function 15.

```
SPC_FCT:15=belt_param(VALUE INTEGER: belt_no
                      VALUE REAL:    v_belt_stopp
                      VALUE REAL:    v_belt_run
                      VALUE INTEGER: time_ms)  ;Function for parameterization of
                                               ;the belt characteristic. It is
                                               ;necessary for being able to adjust
                                               ;the kind of synchronization to the
                                               ;specific characteristic of the
                                               ;conveying equipment
```

belt_param	Name of the function which can be freely selected.
belt_no	Number of the belt for which the values are set (1 to 16).
v_belt_stopp	Threshold value for the belt speed in mm/s at which ’belt stopped’ is detected.
v_belt_run	Threshold value for the belt speed in mm/s at which ’belt running’ is detected.
time_ms	Time in milliseconds in which the messages are tolerated.

It is rounded to a whole interpolation cycle.

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Parameterization of the belt characteristic

Example:

```
PROGRAM SPC15

SPC_FCT:15=belt_param(VALUE INTEGER: belt_no
                      VALUE REAL:    v_belt_stopp
                      VALUE REAL:    v_belt_run
                      VALUE INTEGER: time_ms)

BEGIN

   belt_param(1,10.0,50.0,ROUND(160.00/66.6)*1000.0)

PROGRAM_END
```

The names for the special function and their parameters used in the example are only place holders as for other special functions as well. They can also be renamed without effecting their functionality. The types of the parameters must, however, be declared in this manner and in this order.

For changing the belt parameters, the kinematic of that belt is occupied briefly. If another process is just moving this kinematic belt synchronously, the belt parameters are switched over at the earliest after the end of the belt-synchronous program section.

After a new boot of the control, the following values are the default values:

belt stopped	0.005 mm/interpolation cycle
belt running	0.005 mm/interpolation cycle
time	0 ms

The belt parameters remain active for the same belt for all processes until the next call of special function 15.

If the belt drifts below the threshold value for ’belt stopped’ the monitoring functions are switched off.

The belt speed is calculated from the number of measuring system pulses per interpolation cycle. The result is that the determined belt speed is rastered.

Parameterization of the belt characteristic

Belt stop/belt start with belt synchronization kind 2

If during the application of the belt synchronization without belt-parallel axis, high belt speed changes occur in connection with belt stop/belt start, a program can be stopped in several cases with different messages.

These messages occur because of the fact that the robot's acceleration and deceleration phases are considered during a reduced travel path, either the position according to the belt value cannot be reached in time (belt synchronous – tolerance too high) or an attempt was made to preset a speed in space which exceeds a maximum axis speed.

The program stop caused by these messages can be avoided if for the belt stop/belt start a deviation of the robot is allowed for a limited time which exceeds the programmed tolerance.

To recognize a belt stop or belt start, the control needs threshold values for the belt conditions 'belt stopped' and 'belt running'.

It is assumed that the belt only runs in the programmed belt direction. The threshold value for 'belt stopped' can be easily used for suppressing the message 'belt dir. change progr.' which is triggered by the hunting oscillation of the belt position when it is at standstill while a program waits for a belt to exceed the start value.

Meaning of the threshold value for 'belt stopped'

The belt condition 'belt stopped' is recognized, if the absolute value of the belt speed is less than the indicated threshold value. In this condition, the robot is braked to a speed of 0 and the monitoring devices for 'wrong belt direction' and 'belt synchronous – tolerance too high' are switched off.

Only if this threshold value is exceeded, a speed default value is calculated again for the robot and an attempt is made to have the robot and belt reach the interpolation point of the path simultaneously.

At the time when the threshold value has been exceeded for the last time, the monitoring 'belt synchronous – tolerance too large' is switched off for a programmed time.

If the belt nevertheless moves against the programmed belt direction, the program stops immediately when the threshold value is exceeded.

Parameterization of the belt characteristic

Meaning of the threshold value for ’belt running’

The belt status ’belt running’ is recognized when the absolute value of the belt speed is higher than the indicated threshold value. In this condition, the belt speed is averaged for up to 10 interpolation increments and the speed specification for the robot is calculated from the averaged speed. If the unaveraged belt speed falls below this threshold value, the message for the belt speed is switched off and the monitoring ’belt synchronous – tolerance too large’ is switched off for a programmed time.

If the belt does not stop completely but continues with a speed below the threshold value for ’belt running’, the monitoring function for ’belt synchronous – tolerance too high’ is switched on again at the end of the set time interval.

Only if the belt speed falls below the threshold value for ’belt stopped’, the monitoring of ’belt synchronous – tolerance too high’ remains switched off.

Avoiding the message ’axis velocity exceeded’

A limiting function for the path speed can only very roughly limit the axis speeds which result from the transformation. A message can be avoided by the limitation of the axis speed via machine parameters. Settings for automatic mode in machine parameter P103 and for manual mode in machine parameter P114.

Set axis speeds which exceed the maximum axis speeds, are divided into two or more interpolation intervals (50 to 100 % of the maximum value) and the interpolator is stopped for this period.

The limitation of the axis speed leads to an extension of the travel time and thus to a greater deviation of the robot position from the path.

Parameterization of the belt characteristic

General remarks about synchronization type 2

The coupling factors P503 are not taken into account for this kind of belt synchronization.

The speed change must be so small that the robot can follow it.

This kind of belt synchronization is activated with special function 21 and can be changed in a user-specific way, see chapter 9.

The speed estimated in the first interpolation cycle is approached step by step to the ideal value in each cycle. So there are even with longer blocks no accuracy problems.

Example program

```
;;KINEMATICS:1=rob1                              ;Compiler instructions

;;rob1.WC_NAMES=k1,blt

;;rob1.JC_NAMES=k1,blt

PROGRAM belt_kind2                              ;Program name

SPC_FCT:21=belt_kind(VALUE INTEGER: belt_no     ;Declaration of special function
                     VALUE INTEGER: kind_belt)  ;21 for switch-over of the belt
                                                ;synchronization types

rob1.BELT:501=conveyor

INPUT:1=conveyor_syn

BEGIN

   belt_kind(1,2)                               ;Call of special function 21

   A=100,V=500

   MOVE LINEAR VIA (0,50)

   WAIT UNTIL conveyor_syn=1

   WAIT 0.05                                    ;Possibly necess. for
                                                ;synchronization of
                                                ;PLC scanning time and control
                                                ;cycle

   SYNCHRON rob1 conveyor

      MOVE LINEAR(150.200)

      MOVE LINEAR(150,240)                      ;Corresp. to wait until
                                                ;conveyor >= 240
      MOVE LINEAR(0,360)

   SYNCHRON_END

PROGRAM_END
```

Parameterization of the belt characteristic

Resetting the belt counter is carried out via a RC input, ser. no. 472 to 487. This input must be linked to a user input in the PLC program (here 'conveyor_syn').

With MOVE LINEAR VIA (0, 50), rob1 is run to position 0. The belt value 50 has no significance yet for this movement, but it is important for the first belt synchronous travel block MOVE LINEAR (150, 200). There rob1 is waiting until the belt value has reached the value 50.

If the belt value 50 has been exceeded, it is checked in each scanning step via the averaged belt speed whether the kinematic rob1 reaches the value 150 at the same time as the belt reaches the value 200. If it is necessary, a new speed limit is calculated.

7 Selecting a point-file

With the help of special function 16, it is possible to select a specific point file .pkt file for the operating modes Define, Teach and Print .pkt file which is offered directly without pressing a key additionally, when the corresponding operating mode is selected. For the operating modes Define and Teach, it is also possible to select a point name within the selected point file.

Declaration of special function 16 in the BAPS program:

```
SPC_FCT:16=pt_select(TEXT:    pt_fina
                     TEXT:    pt_ptna
                     INTEGER: ret_code)
```

pt_select	Freely selectable name of special function
pt_fina	Name of point file which is offered in case of Define/ Teach. Enter file name with extension .pkt.
pt_ptna	Name of point which is offered in the file.
ret_code	0 everything OK
	1 faulty file name
	2 wrong file extension, enter extension .pkt
	3 file does not exist
	4 file cannot be opened
	5 file is empty
	11 faulty point name
	12 point in file not found
	13 index in point field not found

As point name, an ’empty name’, i.e. ptna=’ ’, can be transferred. In this case, the first point of the point file is offered for Define/Teach.

For the operating mode Print .pkt file, the point name is not important, the complete file is always printed.

The selected names are reset when selecting the corresponding operating mode.

Selecting a point-file

Example

```
PROGRAM SPCF16

SPC_FCT:16=pt_select(TEXT:    pt_fina
                     TEXT:    pt_ptna
                     INTEGER: ret_code)

TEXT: ptfina,ptptna

INTEGER: ret

BEGIN

   ptfina='test.pkt'

   ptptna='begpos'

   ret=0

   pt_select(ptfina,ptptna,ret)

PROGRAM_END
```

Mirror

8 Mirror

Special function 17 makes it possible to reduce the programming time for symmetric processes that have to be carried out. With points in world or machine coordinates, the indicated coordinates are inverted for all interpolation types. External programs and subroutines are also carried out mirrored. The referencing process is not changed. The special function relates to the just active kinematics (default kinematics).

Declaration

```
SPC_FCT:17=mirror(VALUE BINARY: X_INV,
                                Y_INV,
                                Z_INV,
                                ORI_INV)
```

Call

With the call of the special function, a variable or a constant is transferred for each world coordinate which determines whether the respective world coordinate is to be moved inverted or not inverted.
At the declaration and the call of the special function, according to the axis number of the kinematics (machine parameter P302), a value for each axis must be specified (position–, orientation– and additional axis).

Example of a 4–axis kinematics:

```
mirror(1,1,0,0)              ;1=mirror coordinate, 0=move coordinate normally

MOVE (50,120,-250,10)
```

In this example, the coordinates 1 and 2 are mirrored, coordinates 3 and 4 are moved normally. The robot then moves to the target point (–50, –120, –250, 10).

Switch off

The mirroring of the individual coordinates is active as long as the special function with the value 0 is called for the respective coordinate.

Example for switching off:

```
mirror(0, 0, 0, 0)
```

Mirror

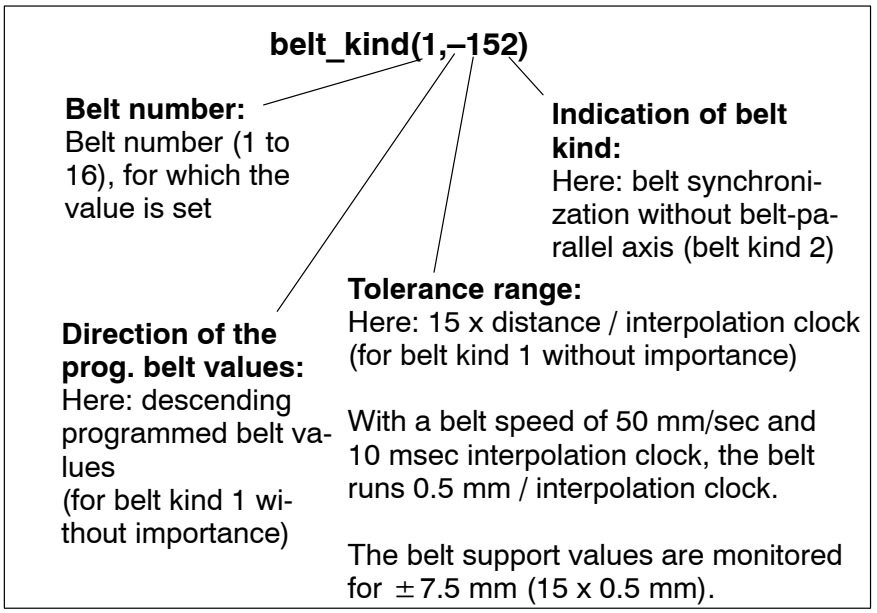
Notes:

Belt kind

9 Belt kind

With special function 21 three kinds of the belt synchronization can be selected. The control synchronization type 1 is active after rebooting . If the synchronization type is changed, this setting is valid for all processes, until the setting is changed again or the control is rebooted.

```
SPC_FCT:21=belt_kind(VALUE INTEGER: belt_no   ;Declaration of the special function
                     VALUE INTEGER: kind_belt)
```



The following synchronizations are partly realized:

belt_no:	Belt number for which the value is set (1 to 16)
kind_belt = 1	Belt synchronization with belt parallel travel of the kinematic
kind_belt = 2	Belt synchronization without belt parallel travel of the kinematic
kind_belt = 3	Cam plate interpolation
kind_belt = 4	as belt kind 1, however with belt-synchronous PTP moving possibility

For the belt synchronization types 2 and 3, the belt synchronous tolerance can additionally be adjusted in the ten’s place.

Standard setting:		5 x distance / interpolation clock
Example:	72 =	7 x distance / interpolation clock 2 = belt synchronization without belt parallel axis

Belt kind

Input range

ten's place	1 to ($2^{31} - 1$) / 10
kind_belt 1	1
kind_belt 2	2, 12, 22, 32, . . ., 122, . . .
kind_belt 3	3, 13, 23, 33, . . ., 123, . . .
kind_belt 4	4

Summary of coding of belt kind

Unit's place of belt

1 = Belt synchronization with belt-parallel axis

2 = Belt synchronization without belt parallel axis

3 = Cam plate interpolation

4 = as belt kind 1, however with belt synchronous PTP moving possibility

Preceding signs

For belt kind 1 without importance

For belt kind 2 and 3:

\+ = ascending programmed belt values

– = descending programmed belt values

Ten's place and higher

For belt kind 1 and 4 without importance

For belt kind 2 and 3:

Tolerance range of the belt values in interpolation clocks. The travel distance resulting from this depends on the belt speed.

Belt kind

Application of special function 21

In the following, a BAPS3 example program is provided by which the application of the special function is explained.

Example

```
PROGRAM SPC21                                    ;Program name

SPC_FCT:21=belt_kind(VALUE INTEGER: belt_no   ;Declaration of the special function
                     VALUE INTEGER: kind_belt)

BEGIN

   belt_kind(1,2)                                ;Selection of the belt kind by
                                                 ;using the special function

PROGRAM_END
```

Remarks and error messages

If a value is programmed for the belt_no in machine parameter P501 which is out of the range from 1 to 16, the runtime message 'Inadmiss. belt No', code 147712 will appear.

If a value is programmed for kind_belt, which exceeds the permitted range (see preceding page), the runtime message 'Wrong belt-kind/-tol code 147200' will appear.

The belt kind remains active until the next call of special function 21 for the same belt.

After a reboot up of the control, the value 1 is entered.

☞ **When a belt is only operated in belt synchronization type 1, it is not necessary to call this special function.**

When a belt is operated in one belt synchronization type only, it is recommended to call this special function in the program INIT.

When a belt is operated in different synchronization types, it is recommended to include the call of this special function before each SYNCHRON instruction.

Belt kind

Notes:

System date and time

10 System date and time

With special function 23 it is possible to access the system clock of the rho4 from a BAPS program. Day, month, year, hours and minutes are determined.

```
SPC_FCT:23=clock_date(INTEGER: hours,minutes,day,month,year)    ;Declaration of the
                                                                ;special function
```

System date and time

Notes:

System counter

11 System counter

The variable transferred with the call of special function 24 is written with the value of the system counter.

The value of the variable is in milliseconds [ms]. The counting raster corresponds to the clock time. With each run-up of the control, the real time counter is reset to zero.

```
SPC_FCT:24=sys_count(INTEGER: c_count)        ;Declaration of the special function
```

System counter

Notes:

12 WC main range

Due to the coordinate transformation, overlappings of the space orientation angles in WC can occur. With special function 27, these overlappings can be eliminated.

```
SPC_FCT:27=wc_mainrange(VALUE INTEGER: kin_no)    ;Declaration of special function
```

☞ **Before the call of special function 27, the kinematic must be run into the main range with PTP-interpolation, e.g. MOVE PTP TO @(0,0,...)!**

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Example

```
;;INCLUDE define                                  ;define contains compiler instr.

PROGRAM SPC27                                     ;Program name

SPC_FCT:27=wc_mainrange(VALUE INTEGER: kin_no)    ;Declaration of special function
                                                  ;kin_no: number of kinematic for
                                                  ;which overlap. is to be eliminted

BEGIN

   MOVE robi_2 PTP@(0,0,0,0,0,0)                  ;Moves the 6-axes-kinematic with
                                                  ;name robi_2 in PTP-interpolation
                                                  ;into main range @(0,0,0,0,0,0)

   wc_mainrange(2)                                ;Call of special function 27 for
                                                  ;the second kinematic

PROGRAM_END
```

WC main range

Notes:

13 Setting the belt counter

With special function 28, a reset value is allocated to the belts. The internal belt counters are set to the corresponding reset value via the signals 'set belt counter', RC-inputs O59.0 to O60.7.

```
SPC_FCT:28=belt_set(VALUE INTEGER: belt_no        ;Declaration of special function
                    VALUE REAL: reset_value)
```

belt_no	Number of the belt for which the value is set
reset_value	Value in [mm] to which the internal belt counter is set via the corresponding input signal. Bit No. (beginning with 0) and belt No. (beginning with 1) must fit.
belt_set	Freely selectable name of the special function.

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Example

```
;;INCLUDE head                                  ;Head contains compiler instruct.

PROGRAM SPC28                                   ;Program name

SPC_FCT:28=belt_set(VALUE INTEGER: belt_no      ;Declaration of special function
                    VALUE REAL: reset_value)

BEGIN

   belt_set(2,-2000)

PROGRAM_END
```

If after termination of the program SPC28 the signal 'set belt counter' is given for the second belt (RC-input 59.1=1), the internal belt counter is set to the value –2000.

Setting the belt counter

The reset value remains active until the next call of special function 28 for the same belt.

After a new run-up of the control, the reset value 0 is entered.

At the belt synchronization in a BAPS program, the internal belt counters are equally set to the reset values active at the moment of the call, for instance

SYNC belt1, I1=1
SYNC belt2 ≤ 100
SYNC belt3 ≥ 200

14 Recording of reference path

With the special functions described in the following, the user is able to trace back the movement sequence of his machine after a stop condition has occurred.

The application of these functions is necessary e.g. when the kinematic is moved synchronously with a belt and the belt cannot be moved properly via the robot control.

When the belt is stopped during synchronization, the belt stops within a specific time. The movement of the machine or of the belt can be traced back when using the functions described in the following.

The functions also permit the determination of measuring values synchronous to the programmed movement sequence via belt channels which are allocated to the moved kinematic. With that, e.g. the dynamic recording of contour deviations is possible (dynamic measuring).

Recording of reference path

14.1 Switching on the reference path recording

With special function 29, the storing of set point values is switched on. The data are stored in a ring memory. The ring memory has a scope of 50 entries. The machine as well as the world coordinates are stored. The storing times result from the travel change of a freely selectable measuring system input. The travel change, the measuring system input as well as the allocated kinematic are parameters of the special function.

```
SPC_FCT:29=pt_mem_on(VALUE INTEGER: kin_no   ;Declaration of special function
                     VALUE INTEGER: ms_no
                     VALUE REAL:    distance)
```

kin_no — The kinematic number indicates the kinematic whose coordinate values are stored. The coordinate values of the belts of this kinematic are also stored.

ms_no — The measuring system number is the number of the measuring system number of the belt, of the axis or of the analog input, to which the parameter 'distance' refers. In the measuring system numeration, first the axes of the 1st kinematic, then those of the 2nd etc. are counted successively. The measuring systems of the belts follow after the last axis measuring system of the last kinematic, starting with the belts of the 1st kinematic.

Finally the measuring system number can refer to an analog input. In this case, the number must be between (number of axes + number of belts + 1) and (number of axes + number of belts + number of analog inputs). The number of axes is the number of all axes in all applied kinematics. Number of belts is the number of all applied belts in all kinematics. If an ms_no is programmed that is too high or negative, the process is stopped with the runtime message 'Inadmis. meassystem', error code: 147584.

distance — If the coordinate value of the belt or of the axis indicated under 'ms_no' changes by more than the value of 'distance' (in [mm] or [degrees]), a storing process is carried out. The storing is performed in the interpolation clock.

☞ **With this special function it is also possible to store points in the time raster of the transformation clock. If you want to use this kind of function, option byte 36 must be set to 1. Storing according to tolerance default is preset, i.e. option byte 36 = 0.**

☞ **Activate special function 29 only once per kinematic, as otherwise values which have already been stored are overwritten.**

Recording of reference path

Release signals

☞ **For an accurate execution of the special function the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Example

```
;;CONTROL=RHO4

;;KINEMATICS:(1=ROB_1,2=ROB_2)

;;ROB_1.JC_NAMES=A11,A12,A13,BN1              ;A11 is the 1. measuring system
                                              ;A12 is the 2. measuring system

;;ROB_1.WC_NAMES=K11,K12,K13,BK1              ;A13 is the 3. measuring system
                                              ;A21 is the 4. measuring system

;;ROB_2.JC_NAMES=A21,A22,A23,BN2              ;A22 is the 5. measuring system
                                              ;A23 is the 6. measuring system

;;ROB_2.WC_NAMES=K21,K22,K23,BK2              ;BN1 is the 7. measuring system
                                              ;BN2 is the 8. measuring system

SPC_FCT:29=pt_mem_on(VALUE INTEGER: kin_no    ;Declaration of special function
                     VALUE INTEGER: ms_no
                     VALUE REAL:    distance)

BEGIN

   pt_mem_on(2,8,13)                          ;The values of the 2nd kinematic A21,
                                              ;A22, A23, BN2 and K21, K22, K23, BK2
                                              ;are always stored when the value
                                              ;of the 8th measuring systems BN2
                                              ;changes by more than 13 mm.

PROGRAM_END
```

Storing of the position values in the transformation clock

With special function 29 it is also possible to store points in the time raster of the transformation clock (also rough interpolation clock or clock raster).

The switch-over is carried out optionally via machine parameters.

Recording of reference path

OPTION BYTE = 0 :	Recording of the reference path if the tolerance indicated in 'distance' (in [mm] resp. [degrees]) is exceeded.
OPTION BYTE = 1 :	Storing of the points in the transformation time raster. 'distance' = number of transformation clocks In the parameter 'distance' of special function 29, the number oft transformation clocks is transferred as decimal number (clocks P5) in the raster of which the point values are to be stored.

Example

```
SPC_FCT:29=pt_mem_on(VALUE INTEGER: kin_no    ;Declaration of special function
                     VALUE INTEGER: ms_no
                     VALUE REAL:    distance)

BEGIN

   pt_mem_on(1,4,10)                          ;Call

PROGRAM_END
```

In the example, the point values of the first kinematic are stored in each 10th transformation clock (rough interpolation clock).

The measuring system number (4) is redundant in this case. A valid value must, however, be transferred with the special function call to avoid that an error message is put out. If an invalid 'ms_no' is programmed, the process is stopped with the runtime message 'Inadmis. meas.sys.', error code: 147584.

14.2 Switching off the reference path recording

With special function 30, the storing of set point values is switched off. Stored data are retained until special function 29 is called again with the same kinematic number.

The special function supplies the number of points that have been stored until the switch-off process in the parameter 'num_points'.

```
SPC_FCT:30=pt_mem_off(VALUE INTEGER: kin_no     ;Declaration of special function
                      INTEGER: num_points)
```

Recording of reference path

kin_no	Number of kinematic values of which are no longer to be stored.
num_points	Return parameter of special function. Indicates the number of valid points in the ring memory that have been stored since the call of special function 29.

Example

```
;;CONTROL=rho4

;;KINEMATICS:(1=ROB_1,2=ROB_2)

;;ROB_1.JC_NAMES=A11,A12,A13,BN1              ;A11 is the 1. measuring system
                                              ;A12 is the 2. measuring system

;;ROB_1.WC_NAMES=K11,K12,K13,BK1              ;A13 is the 3. measuring system
                                              ;A21 is the 4. measuring system

;;ROB_2.JC_NAMES=A21,A22,A23,BN2              ;A22 is the 5. measuring system
                                              ;A23 is the 6. measuring system

;;ROB_2.WC_NAMES=K21,K22,K23,BK2              ;BN1 is the 7. measuring system
                                              ;BN2 is the 8. measuring system

SPC_FCT:30=pt_mem_off(VALUE INTEGER: kin_no   ;Declaration of special function
                            INTEGER: num_points)

BEGIN

   pt_mem_off(2,num_points)                   ;The values of the second kinematic
                                              ;A21, A22, A23, BN2 and K21, K22,
                                              ;K23, BK2 are no longer stored
                                              ;when the processing of special
                                              ;function (pt_mem_off) is started.
                                              ;The BAPS variable num_points
                                              ;is occupied by the special
                                              ;function with the number of
                                              ;the stored points and can be used
                                              ;for reading out the points

PROGRAM_END
```

Recording of reference path

14.3 Reading of reference path values

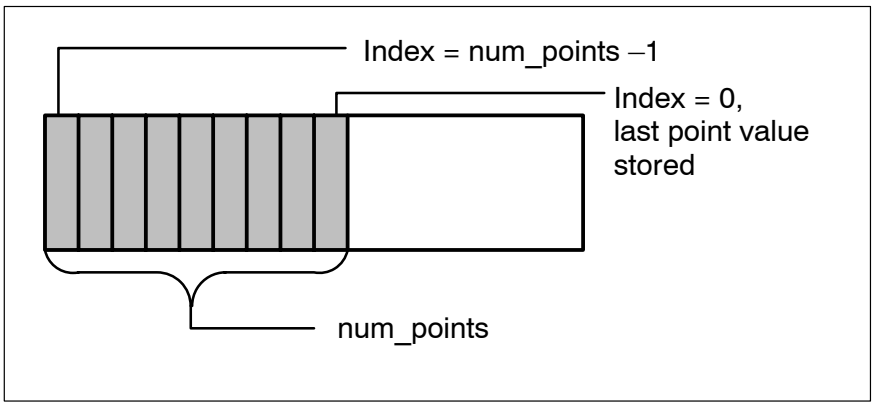
With special function 31, the stored set point values can be read in machine coordinates and in world coordinates. A maximum of 50 values can be read from a ring memory.

For getting valid data, it is necessary to call special functions 29 and 30 before calling special function 31. If special function 31 is to be used for several kinematics, a special function name, as e.g. 'pt_mem_read', 'pt1_mem_read' or 'pt2_mem_read', must be declared for each kinematic.

For reading valid points from the ring memory in any case, the reset parameter 'num_points' of special function 30 should be taken into account. If one tries nevertheless to access an element of the ring buffer that has not yet been stored, the process is stopped with the runtime message 'point is invalid', code: 147256.

```
SPC_FCT:31=pt_mem_read(VALUE INTEGER: kin_no    ;Declaration of special function
                       VALUE INTEGER: index
                       POINT:         @position
                       JC_POINT:      @position)
```

kin_no	Indicates the kinematic whose coordinate values have been returned. The coordinate values of the belts which belong to this kinematic are also available.
index	Indicates which ring memory entry is read.
@position	Here, the machine coordinate values of the kinematic indicated under 'kin_no' are entered after the call.
position	Here, the world coordinate values of the kinematic indicated under 'kin_no' are entered after the call.



Recording of reference path

Example

```
;;CONTROL=rho4

;;KINEMATICS:(1=rob_1,2=rob_2)

;;rob_1.JC_NAMES=A11,A12,A13,BN1                  ;A11 is the 1st measuring system
                                                  ;A12 is the 2nd measuring system

;;rob_1.WC_NAMES=K11,K12,K13,BK1                  ;A13 is the 3rd measuring system
                                                  ;A21 is the 4th measuring system

;;rob_2.JC_NAMES=A21,A22,A23,BN2                  ;A22 is the 5th measuring system
                                                  ;A23 is the 6th measuring system

;;rob_2.WC_NAMES=K21,K22,K23,BK2                  ;BN1 is the 7th measuring system
                                                  ;BN2 is the 8th measuring system

SPC_FCT:31=pt_mem_read(VALUE INTEGER:    kin_no   ;Declaration for kinematic 1
                       VALUE INTEGER:    index
                      rob_1.JC_POINT:    position_1
                      rob_1.JC_POINT:    @position_1)

SPC_FCT:31=pt2_mem_read(VALUE INTEGER:   kin_no   ;Declaration for kinematic 2
                        VALUE INTEGER:   index
                       rob_2.JC_POINT:   position_2
                       rob_2.JC_POINT:   @position_2)

;;KINEMATICS=rob_2

POINT:WC_position

JC_POINT:@JC_position

BEGIN

   pt2_mem_read(2,1,WC_position,@JC_position)       ;The values of the second
                                                    ;kinematic A21, A22, A23, BN2
                                                    ;and K21, K22, K23, BK2 are
                                                    ;returned: in WC_position the
                                                    ;world coordinates and in
                                                    ;@JC_position the
                                                    ;machine coordinates.
                                                    ;The values are those of
                                                    ;the storing before
                                                    ;the last storing operation

PROGRAM_END
```

Recording of reference path

Notes:

15 Flying measurement (rho4.1 only)

For measuring with a delay time as short as possible in dependence of an external signal, the special functions 43 and 44 'Flying measurement via probe input' are used.

Two special functions are defined with the help of which the flying measurement is activated and deactivated. The measured value is written into the standard variable '@MPOS' and can be read within a user program.

Syntax of the special function 43

```
SPC_FCT:43=probe_on(VALUE INTEGER: kin_no
                    VALUE INTEGER: edge
                    INPUT:         channel_no)
```

Syntax of the special function 44

```
SPC_FCT:44=probe_off(VALUE INTEGER: kin_no
                           INPUT: channel_no)
```

KIN_NO	Number of kinematic for which the function is to be activated or deactivated.
EDGE	Defines the edge of the probe signal at which the measurement is to be started. (EDGE = 0: positive edge, EDGE = 1: negative edge)
CHANNEL_NO	Channel No of the probe input (at present only 610 possible)

Release signals

☞ **For an accurate execution of these special functions the following interface signals must be set:**

- **Automatic**
- **no Emergency stop**
- **Drive on all axes, resp. the single signals of the corresponding axes**
- **Feed allow all kinematics, resp. the single signals of the corresponding kinematics**
- **Travel enable all kinematics, resp. the single signals of the corresponding kinematics**
- **no emergency operation**

Flying measurement (rho4.1 only)

Process of a measurement

★ Activate measurement via function Probe_On.

★ Program movement if required (movement to the measuring point).

★ Inquiry if probe has triggered (if probe = 1...).

If the probe has triggered, the position value at the time of the activation of the probe is saved in @MPOS. @MPOS has a BAPS standard variable. The measuring position is only available in machine coordinates. It can however be converted into space coordinates via the BAPS command WC. When the probe has triggered, the measuring function is automatically deactivated.

If the probe has not triggered, the measurement must be deactivated via the function Probe_off. The value of the last successful measurement remains in this case saved in @MPOS. In the first measurement, @MPOS remains in the active program unoccupied. This means that in case of a read access to @MPOS, the program will be interrupted with the runtime message 'Point not def.', code:

☞ **The probe input only works if the incremental encoder input (x41) of the rho4.1 is assigned (parameter 401).**

Example

```
;;KINEMATICS:(1=robi1)
;;robi1.JC_NAMES=A1,A2,A3
;;robi1.WC_NAMES=K1,K2,K3

PROGRAM measure

CONST:spc_no1=43,
      spc_no2=44,
      pos_edge=0,
      neg_edge=1

robi1.JC_POINT:@P1,@Meas_Pos

INPUT:610=probe1

SPC_FCT:spc_no1=probe_on (VALUE INTEGER: kin_no
                          VALUE INTEGER: edge
                                  INPUT: channel_no)

SPC_FCT:spc_no2=probe_off(VALUE INTEGER: kin_no
                                  INPUT: channel_no)

BEGIN

   probe_on(1,pos_edge,probe)
```

Flying measurement (rho4.1 only)

```
   Move to @P1

   IF probe1=1
     THEN
      BEGIN
      @Meas_Pos=@MPOS
      Write (@Meas_Pos)
      End
     ELSE
      BEGIN
      probe_off(1,probe)
      write 'Probe has not triggered'
      END

PROGRAM_END
```

Flying measurement (rho4.1 only)

Notes:

MOVE_FILE

16 MOVE_FILE

With special function 'MOVE_FILE', position profiles can be exactly copied in the rho4. For each position controller clock, a freely definable position setpoint can be preset.

Transfer parameter indicates the name of the file which is to be executed and an identifier which codes whether its supporting points are stored in the interpolation or position controller raster. This file is a binary file and contains, in binary and unformatted form, the setpoints ot the individual axes of a kinematic for each clock. It does not contain any separators. The axis setpoints are always position setpoints.

16.1 Structure of the binary file

Binary files for special function 45 contain the file extension .bnr. Binary files consist of a head block and the actual data division.

16.1.1 File head

The file head contains different file parameters, which are used for addressing the data division start and for comparison to the control internal machine parameters resp. the special function parameters.

The file head has the following structure:

File parameters	Length	Type
Head length	4 bytes	Integer
Interpolation / position controller base	4 bytes	Integer
Time base	4 bytes	Integer
Number of axes	4 bytes	Integer
Version identification	4 bytes	Integer
Disable base check	4 bytes	Integer
Reserve	14 * 4 bytes = 56 bytes	Integer

Head length

The head length of a .bnr file is 80 bytes. But also other bnr files with other head lengths can be used in conjunction with 'MOVE_FILE'. But it is important that the correct head length is entered in the file head to be able to address the start of the data division.

MOVE_FILE

Interpolation / position controller base (IP/PC base)

The parameter indicates whether the individual point values of the file are available in the interpolation raster or in the further subdivided position controller raster.

IP/PC base = 1, the points are available in the interpolation raster

IP/PC base = 2, the points are available in the position controller raster

Time base

This parameter contains the time base in micro-seconds in which the file has been created. Dependent on parameter 2, this is the value of machine parameter P5 (Clock time) * 1000, if necessary divided by the servo board divisor.

Number of axes

This parameter contains the number of axes of the point values stored in the file.

Version identifier

This parameter contains an integer number, corresponding to the associated control software version.

Disable base check

By a defined setting of this parameters to value 1, the check of the interpolation / position controller base as well as of the time base is switched off. With values unequal 1, these file parameters are compared with the special function parameters and the control internal machine parameters.

Reserve

The rest of the file head is designed as a reserve buffer of 56 bytes (14 * 4 bytes) length and available for user specific file identifiers and extensions. This results in a head length of 80 bytes. The reserve buffer is preset with 0.

Data division

The data division follows immediately after the head block and contains point components in the 32 bit IEEE floating point format without separators. The length of a point component is 4 bytes. The storing requirement for a point is calculated as number of axes * 4 bytes. The address of the first point of the file results from the start address of the file + head length.

MOVE_FILE

Sequence within the data division:

Interpolation point 1	1st clock	Axis 1
	1st clock	Axis 2
	1st clock	Axis 3
Interpolation point 2	2nd clock	Axis 1
	2nd clock	Axis 2
	2nd clock	Axis 3

Example

Structure of a binary file

Example of a binary file

Address relative to the file start (HEX):	Structure of a binary file	Example of a binary file
00	Head length	80
04	IP/PC base	2
08	Time base	400
0C	Number of axes	1
10	Version identification	500
14	Base check OFF	0
18	Reserve	Reserve
50 54	Data division	0.0288 0.0864 0.1728 0.2880 0.4320 0.6048 . . . . 359.712 359.827 359.914 359.971 360.000

All address information is represented in hexadecimal form. The data division consists of 32 bit IEEE floating point numbers.

The example shows a binary file for a single axis kinematic. The data division contains position setpoints which refer to a position controller clock of 400 micro seconds.

Base check OFF = 0 means that the parameter 'IP/PC base' and 'time base' are checked.

16.2 MOVE_FILE

With the call of special function 45, the course of a curve is followed which is stored in a binary file. From the file, the values are read and put out unchanged in the interpolation raster or in the position controller raster of the servo board clock as position setpoints. The call of special function 45 is called in the following MOVE_FILE block.

```
SPC_FCT:45=move_file(VALUE INTEGER: kin_no              ;Declaration of the
                     BNR_FILE:      curve_x             ;special function
                     VALUE INTEGER: base
                     JC_POINT:      @mod_flag
                     VALUE ARRAY[1..6] INTEGER: reserve)
```

kin_no	Indicates the kinematic to be moved
curve_x	Name of the binary file for the movement
base	Indicates whether the position setpoints of the file are issued in the interpolation raster or in the position controller raster. 1 = interpolation raster 2 = position controller raster
mod_flag	The modulo flag is required for endless axes as identifier. For other axis types, the value must be preset with 0.0. The following decimal values are permissible as identifiers: 0.0 = No modulo computation 1.0 = Modulo computation at the end of the block. Modulo value is the last axis value in the binary file. 2.0 = Modulo computation at the end of the block. Modulo value is the set machine parameter value P311.
reserve	This parameter is determined for possible expansions of later operating system versions.

MOVE_FILE

If at the end of a MOVE_FILE block a modulor computation is to be executed for an endless axis, there are two possible modulations. Modulation is possible either with the file end value of the corresponding axis or with the modulo value set in machine parameter P311.

The modulo flag must always be preset, as otherwise the process is stopped with the runtime error message 'MF: MOD-Flag n. init.', code 145792.

Opening the bnr file

Before the first call of the MOVE_FILE, the associated binary file must first be opened for reading by the BAPS command 'READ_BEGIN'. When opening the file, the read indicator within the control system must be positioned at the start of the data division of the .bnr file. This is obtained by including a byte offset in the READ_BEGIN command. The offset is in this case equal to the file head length (80 bytes).

The command for opening the file is

READ_BEGIN curve_x, 80

or:

CONST: head_length = 80

READ_BEGIN curve_x, head_length

After the file has been opened, it can be used as often as desired in MOVE_FILE calls. The file must be closed again with the BAPS command CLOSE after the last MOVE_FILE call. If no READ_BEGIN has been programmed before the first MOVE_FILE block, the runtime error message 'READ_BEGIN expected', code 136192 is issued.

The following example shows the declaration and the call of the special function. Two curve courses are travelled alternatively which are stored in two .bnr files 'JC_0_90', 'JC_90_0'.

MOVE_FILE

Example

```
;;CONTROL=rho4

;;KINEMATICS:1=robi1

;;robi1.JC_NAMES=A1,A2

;;robi1.WC_NAMES=K1,K2

PROGRAM movedat

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)

BNR_FILE: jc_0_90,jc_90_0

JC_POINT: @mod_flag

ARRAY[1..6] INTEGER: reserve

CONST: ip_raster=1,
       pc_raster=2,
       head_length=80

BEGIN

   READ_BEGIN jc_0_90, head_length

   READ_BEGIN jc_90_0, head_length

   @mod_flag=@(0,0)                                    ;No endless axes

   REPEAT 10 TIMES

     move_file(1,jc_0_90,pc_raster,@mod_flag,reserve)

     move_file(1,jc_90_0,pc_raster,@mod_flag,reserve)

   REPEAT_END

   CLOSE jc_0_90                                       ;Close .bnr files

   CLOSE jc_90_0

PROGRAM_END
```

MOVE_FILE

16.3 Safety checks

The entries in the file head are compared with the machine parameters and with the programmed special function parameters in the MOVE_FILE block.

The following runtime errors may arise during this check:

Runtime messages	Status code	Description
Inad. real expression Decimal number	142208	The first and the last points of the .bnr file are checked at the start of the MOVE_FILE block to ascertain whether their components are resonable floating point numbers in IEEE format. For runtime reasons, it is, however, not possible to check the complete .bnr file. To exclude system errors (FPU traps) completely, the user can use a BAPS program to check the .bnr file for unpermitted floating point numbers (type REAL). This check only has to be carried out once per .bnr file and can take place at a temporally uncritcal point in the BAPS program or once before the start of the MOVE_FILE process. The BAPS program could resemble the 'l_curve' example program shown below. The program 'l_curve' reads the file 'curve.bnr' and additionally creates from this a file 'd_curve.dat'.
MF: Version identification	145024	The version identification of the file is not compatible with the installed rho4 operating system version.
MF: Number of axes	145152	The number of axes of the kinematic is unequal to the number of axes entered in the file head.
MF: Read marker	145280	The control internal read marker has not been positioned within the data division of the .bnr file in the READ_BEGIN command. It points to the file head.
MF: Time base	145408	The file parameter does not fit the corresponding control internal clock time (interpolation or position controller clock). This monitoring can be switched off by the the file parameter base_check_off.
MF: IP/PC base	145536	The file parameter 'ip/pc base' does not fit the special function parameter 'base' (interpolation or position controller base). This monitoring can be switched off by the the file parameter base_check_off.

MOVE_FILE

Runtime messages	**Status code**	**Description**
MF: Format error	145664	Either the data division of the .bnr file has an incorrect length or the control internal read marker has not been positioned correctly, in the READ_BEGIN command, inside the data division of the .bnr file. The 2nd case arises when the read marker is not positioned at the start of the data division, which is theoretically permitted. In this case, it should absolutely be ensured that the number of bytes from the read marker to the file end is divisible by (4*number of axes). The following permissible values then result for the offset in the READ_BEGIN command: Offset = Head length + N*(4*number of axes) N: integer >= 0 Head length: 80
MF: MOD-Flag not init.	145792	The special function parameter '@modulo_flag' has not been preset. For all axis types it must be preset with at least 0, as the data type of the flag itself is a '@JC_POINT' type.
MF: No endless axis	145920	In the special function call of the MOVE_FILE, the value 1.0 has been transferred for the axis component of the modulo flag, although the corresponding axis in the machine parameter program is not declared as an endless axis.

MOVE_FILE

Example

```
PROGRAM l_curve

CONST:                  ax=2            ;Number of axes
BNR_FILE:               curve           ;Source, mixed .bnr-file (INTEGER + DEC)
FILE:                   d_curve         ;Destination, .dat file
ARRAY[1..20] INTEGER:   head            ;Head length = 80 bytes = 20 INTEGER values
ARRAY[1..ax] REAL:      data            ;Number of array elements = number;of axes

BEGIN

   READ_BEGIN curve                     ;Read file head from .bnr file curve.bnr and

   WRITE_BEGIN d_curve                  ;write to .dat file d_curve.dat

   READ curve,head                      ;Read data from .bnr file curve.bnr and

   WRITE d_curve,head                   ;write to dat file d_curve.dat

   mark_1:                              ;read loop

      READ curve,data

      WRITE d_curve,data

      IF NOT END_OF_FILE(curve)

      THEN JUMP mark_1

   CLOSE curve

   CLOSE d_curve

PROGRAM_END
```

MOVE_FILE

Notes:

17 Setting the block preparation

With the help of this function it is possible to reduce the internal block preparation, which the block preparation constructs relative to the currently active MOVE block. The application is recommended for use in connection with the special functions'move_file' and 'asynchronous inputs'. Especially when using asynchronous inputs, it may be necessary to reduce the block preparation in order have a quicker effect of the input change on the prepared blocks.

Processes which move one or more kinematics are separated in the rho4 into a block preparation section and a kinematic-related section which controls the axis movement.

The term block preparation refers to the number of blocks prepared for a kinematic. The preset block preparation is 11 blocks. It can, however, be set variably for program runtime with the help of special function 46. It is also permitted to change the block preparation several times within a program.

```
SPC_FCT:46=block_prepar(VALUE INTEGER: kin_no     ;Declaration of special function
                        VALUE INTEGER: number)
```

kin_no	Indicates the kinematic to be moved
number	In this parameter an integer number in the value range of 1 to 11 can be transferred to the special function. This number represents the upper limit for the number of kinematic blocks to be prepared. With values <1 the value 1 is internally set, with values >11 the value is internally set to 11.

Call in BAPS

```
block_prepar(1.5) ;5 blocks of block preparation for kinematic 1

block_prepar(3,8) ;8 blocks of block preparation for kinematic 3
```

Setting the block preparation

Notes:

18 Exception handling

The term “exception handling” means the programmable reaction to state messages in user processes of the rho4. Without manual intervention of the user, some exceptions can be recognized and eliminated through specific starting of exception processes. Programmable exceptions are state messages of the belt kind 4 and messages that are recognized in the block preparation.

18.1 Defining exceptions

Up to 32 exceptions are programmable. They are defined with the help of the special function 47 EXC_DEFINE and saved in a table. Each entry gets a number between 1 and 32 (EXC_NR) assigned by the user. 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.

18.1.1 Declaration of special function EXC_DEFINE

The special function 47 is declared as follows:

```
SPC_FCT : 47 = EXC_DEFINE     ( VALUE INTEGER : EXC_NR
                                VALUE INTEGER : ERR_CODE_1
                                VALUE INTEGER : ERR_CODE_2
                                         TEXT : EXC_PR_NAME
                                      INTEGER : STATE       )
```

The meanings of the function parameters are described in table 1.
Input parameters are parameters of the function that must be set by the user. Output parameters are parameters that are returned by the function.

Exception handling

Input parameter	
EXC_NR	Exception number, permissible numbers are 1..32
ERR_CODE_1	Error code of the exception, resp. error code range lowest limit “0” means: exception get deleted
ERR_CODE_2	Error code range highest limit “0” means: only ERR_CODE1 is effective
EXC_PR_NAME	Program name of exception process (max. 8 character), which is started automatically when a process state message by ERR_CODE1 resp. ERR_CODE1..ERR_CODE2 occur
Output parameter	
STATE	Return code of function (0: no Error, <0: Error, >0: warning) “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

Table 1: Parameter of special function EXC_DEFINE

18.1.2 Permissible state messages and codes

Table 2 describes the state messages and codes that can be programmed as an event to start an exception process. If the programmed codes are outside the limits given in table 2, the function EXC_DEFINE returns the STATUS = –2 (inadmissible ERR_CODE).

State messages	**permissible codes**
Belt kind 4 – State messages	19584 – 19839
Travel limits	22144 – 22271
State messages of the block preparation	131840 – 150912

Table 2: Permissible state messages and codes

Exception handling

18.1.3 Call of special function EXC_DEFINE

The example program ExcDef shows the definition of several exceptions initializing exception processes.

```
1.  PROGRAM ExcDef
2.
3.  SPC_FCT : 47 = EXC_DEFINE (    VALUE INTEGER : EXC_NR
4.                                 VALUE INTEGER : ERR_CODE_1
5.                                 VALUE INTEGER : ERR_CODE_2
6.                                          TEXT : EXC_PR_NAME
7.                                       INTEGER : STATE )
8.  INTEGER  : RetState
9.  TEXT     : ExcPrName
10.
11. BEGIN
12.
13. ExcPrName = 'EXCPT01'
14. ;***** Division by Zero (Error code 138240) **************
15. EXC_DEFINE (  1, 138240, 0, ExcPrName, RetState )
16. IF RetState <> 0
17.   THEN BEGIN ;*** Error in Definition, check parameter ***
18.         WRITE PHG, 'Error in exception definition'
19.        END
20.
21. ExcPrName = 'EXCPT02'
22. ;***** Travel range limit (Error code=22144+AxNo-1) ******
23. ;***** Axes 1-6 range definition                    ******
24. EXC_DEFINE (  2, 22144, 22149, ExcPrName, RetState )
25. IF RetState <> 0
26.   THEN BEGIN ;*** Error in Definition, check parameter ***
27.         WRITE PHG, 'Error in exception definition'
28.        END
29.
30. ExcPrName = 'ExcptBA4'
31. ;***** G2 und G3 - Error beltkind 4  ********
32. ;***** Error codes for max. 16 belts ********
33. EXC_DEFINE ( 3, 19584, 19712+15, ExcPrName, RetState )
34. IF RetState <> 0
35.   THEN BEGIN ;*** Error in Definition, check parameter ***
36.         WRITE PHG, 'Error in exception definition'
37.        END
38.
39. PROGRAM_END
```

Exception 1 reacts to the process state message 'division by zero' (Code 138240).
see also rho manual 'State messages and Warnings'
If this message appears, the exception process 'EXCPT01' will be started.

Exception 2 reacts to the process state message 'travel range limit' (Code range entry 22144 .. 22149).
If this message appears for one of the axes 1 – 6, the exception process 'EXCPT02' will be started.

Exception handling

Exception 3 reacts to the process state message of belt kind 4 (Code range entry 19584 .. 19727).
If a belt kind 4 message appears, the exception process 'ExcptBK4' will be started.

To make things clear, figure 1 shows the workspace of a swivel arm motor taking parts from a belt (surface marked in grey).
If the parts on the belt exceed certain limits (G2 and G3), this leads to various process state messages. They will be used as an event for the exception handling.

If for instance a piece on the belt to be gripped reaches the Y coordinate G3 before the robot can grip the piece, the motion will be interrupted and the rho4 state message 'BS-Take Limit' (Code 19712) will appear.

If the piece reaches the Y coordinate G2 before the robot moves the first belt-synchronous block, the motion will not be started and the rho4 state message 'BS-Begin Limit' (Code 19584) will be displayed.

In both cases, the process goes into the state 'Error' and waits to be stopped and restarted. These actions are carried out by the exception process 'ExcptBK4'.

Remark: With state messages of belt kind 4, the ready2 contact will not be opened.

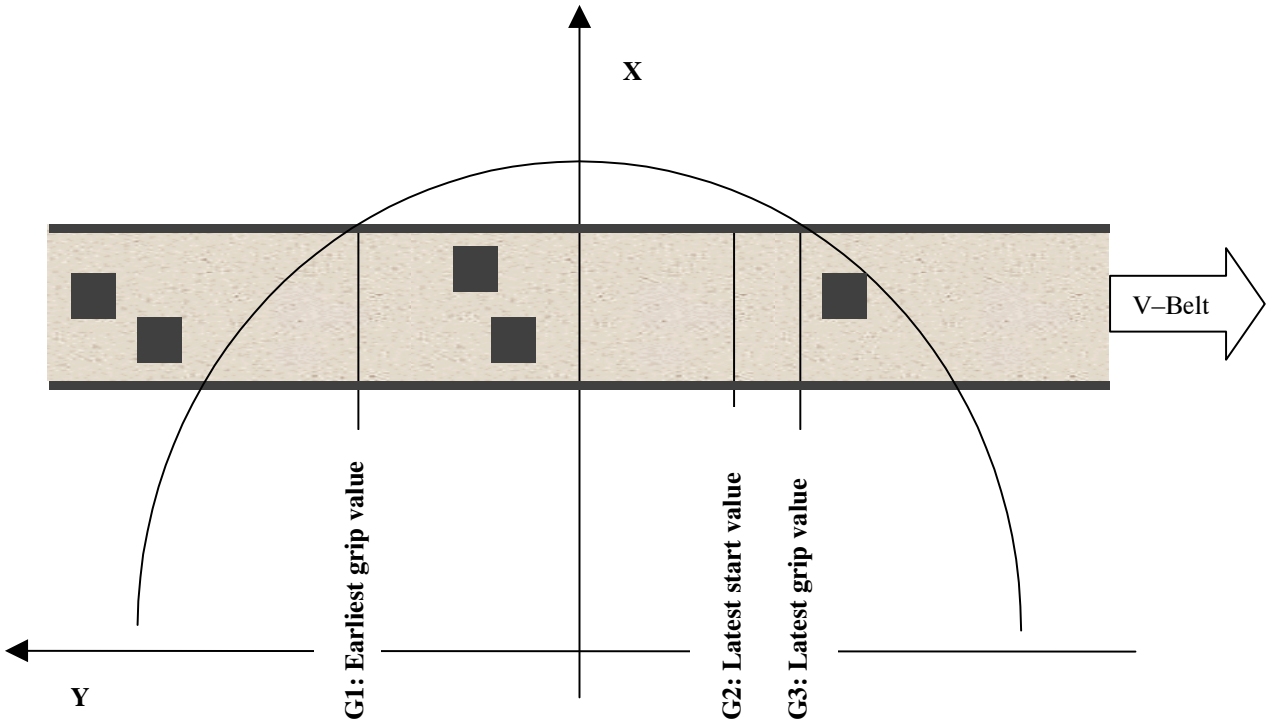


Figure 1: Belt kind 4 – Exceptions G2 and G3

Exception handling

18.2 Exception processes

When the exceptions defined through ERR_CODE1 (.. ERR_CODE2) occur, the assigned exception process (ExcPrName) will be started. It carries out the actual exception handling.

Exception processes are user processes that are freely programmable, in which user processes with exceptions are determined, stopped and restarted again. The special function 48 EXC_DETECT is used to determine the user processes with exceptions.

18.2.1 Look for happened exceptions

The special function EXC_DETECT enables to detect processes that have reached an exception state defined by EXC_DEFINE. Entry parameters of the function are the error codes ERR_CODE_1 and ERR_CODE_2 used in EXC_DEFINE. The function EXC_DETECT 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. EXC_DETECT should be called within an exception process. The process names ERR_PR_NAME[n] returned by the function can then be used to stop and start the incorrect processes.

18.2.2 Declaration of special function EXC_DETECT

The special function 48 is declared as follows:

```
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 )
```

The constant MAX_ERR_PROC=16 is declared in the BAPS-Include-File **rmain.inc**.

The meanings of the function parameters are described in table 3.
Input parameters are parameters of the function that must be set by the user. Output parameters are parameters that are returned by the function.

Exception handling

Input parameter	
ERR_CODE_1	Error code of the exception resp. lowest error code range limit for limit declaration
ERR_CODE_2	Error code range highest limit “0” means: only ERR_CODE1 (no limit declaration) is effective
Output parameter	
NUM_ERR_PROC	Number of faulty processes
ERR_PR_NAME	Program name (max. 8 character) of faulty processes with error code ERR_CODE_1 .. ERR_CODE_2 Up to 16 faulty processes can be stored
ERR_SUB_NO	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, ERR_SUB_NO = 0
PR_ERR_CODE	Returns the occured process error code by limit declaration
STATE	Return code of function (0: no error, <0: error) “0” : Function was executed correctly “–2” : Inadmissible ERR_CODE

Table 3: Parameter of special function EXC_ DETECT

18.2.3 Permissible state messages and codes

The state messages and codes described in table 2 also apply for the function EXC_DETECT. If the programmed codes are outside the limits given in table 2, the function EXC_DEFINE returns the STATUS = –2 (invalid ERR_CODE).

18.2.4 Call of special function EXC_DETECT

The call of the detect function will be explained with the help of a concrete example. For that purpose, the belt kind 4 example used for the definition of exceptions will be carried on.

Assumption: a belt synchronous move process of belt kind 4 signals the process state ’BS-Take Limit’ (Code 19712). The appearance of this message serves as trigger condition to start the exception process ’ExcptBA4’. Within this process, the function EXC_DETECT will be called to determine the number and name of the processes that have reached the belt kind 4 exceptions.

Exception handling

The parameter ErrPrName[n] is a text field, in which the special function EXC_DETECT returns the names of up to 16 processes with the appropriate error codes.
The elements of this text field can be directly used in the special function COMMAND as input parameter to eliminate the concerned processes and to restart.

☞ **If the concerned faulty process is a SUB process (parallel path), the main process name will be however returned because individual SUB processes cannot be stopped and started. In this case, the corresponding main process must be stopped and restarted. As additional information the number of the parallel path in the parameter ErrSubNo is signaled in faulty SUB processes.**

The example program 'ExcptBK4' shows the real exception handling within the exception process started automatically.

Program example : Exceptions of beltkind 4

```
40. ;;CONTROL = RHO4
41.
42. PROGRAM ExcptBK4
43.
44. ;;INCLUDE rmain.inc     ;contains the constant MAX_ERR_PROC = 16
45. CONST :   STARTING = 4,
46.           STOPPING = 5
47.
48. SPC_FCT :  4 = COMMAND ( VALUE INTEGER  : COM_IDENT_NO
49.                          TEXT     : SOURCE, DESTINATION
50.                          INTEGER  : STATE              )
51. SPC_FCT : 48 = EXC_DETECT
52.                ( VALUE INTEGER                      : ERR_CODE_1
53.                  VALUE INTEGER                      : ERR_CODE_2
54.                  INTEGER                            : NUM_ERR_PROC
55.                 ARRAY[1..MAX_ERR_PROC] TEXT         : ERR_PR_NAME
56.                 ARRAY[1..MAX_ERR_PROC] INTEGER      : ERR_SUB_NO
57.                 ARRAY[1..MAX_ERR_PROC] INTEGER      : PR_ERR_CODE
58.                  INTEGER                            : STATE )
59.
60. INPUT  : 3 = I3
61. OUTPUT : 3 = O3
62.
63. ARRAY[1..MAX_ERR_PROC] TEXT    : ErrPrName
64. ARRAY[1..MAX_ERR_PROC] INTEGER : ErrSubNo
65. ARRAY[1..MAX_ERR_PROC] INTEGER : PrErrCode
66.
67. INTEGER : NumbErrProc, PrNo, RetState
68. TEXT    : ProcName, DummyTxt
69.
70. BEGIN
71.
72. Loop:
```

Exception handling

```
O3 = 0
WAIT 0.33
O3 = 1
WRITE PHG, CLS, 'Exception process 03 started'
WAIT 0.33
IF I3=1
  THEN ;******** Check and clear up process error G2 and G3 *******
     BEGIN
     ;*** Error code 19584  G2 error belt kind 4 ***
     ;*** Error code 19712  G3 error belt kind 4 ***
     EXC_DETECT( 19584, 19712+15, NumErrProc, ErrPrName,
                          ErrSubNo, PrErrCode, RetState )

     IF RetState <> 0
       THEN WRITE PHG, '*** Error in EXC_DETECT : ', RetState, '***'

    PrNo = 0
    REPEAT NumErrProc TIMES
       PrNo = PrNo + 1
       COMMAND ( STOPPING, ErrPrName[PrNo], DummyTxt, RetState )
       IF RetState <> 0
         THEN BEGIN
             WRITE PHG, '*** Error in STOPPING ***'
              END
     REPEAT_END

     PrNo = 0
       REPEAT NumErrProc TIMES
         PrNo = PrNo + 1
         COMMAND ( STARTING, ErrPrName[PrNo], DummyTxt, RetState )
         IF RetState <> 0
           THEN BEGIN
               WRITE PHG, '*** Error in STARTING ***'
                END
           ELSE WRITE PHG, CLS, 'Exception process has cleared up'

       REPEAT_END
       END

    ELSE  ;******** Blinking of O3 and waiting for I3 **********
       BEGIN
        WAIT 0.5
        JUMP Loop
       END

PROGRAM_END
```

Exception handling

18.3 State messages of the exception handling

The display of the state messages of the exception handling are available in the PHG mode 7.2 and in ROPS4-Online in the menu State messages. Furthermore, an access to these informations is possible with the help of the rho4 library functions of the class 5000 (rhoError [rE]).

Problems occuring at the automatic start of an exception process generate the message “EXC-process start” (code 150144).

This state message says that a problem has occured at the automatic start of an exception process and the exception process programmed in EXC_DEFINE could not be started.

status messages

EXC-process start

code = 150144

The exact cause preventing the exception process from starting is displayed in a further state message with the addition of the appropriate exception number.

status messages

file not found
EXC No.: 3

code = 495744

The second state message says that the trigger condition for the third exception has occured, but the IRD file of the exception process to be started has not been found in the rho4 user memory. When referring to the example 2.3 (EXC_DEFINE), this means that the file ’ExcptBK4.IRD’ must be loaded into the rho4.

In the connection with an exception process start, all state messages that can be produced at a normal start of processes can also be produced and displayed with one exception.

see also : rho4 manual **Status messages and warnings**

The exception is the state message ’Process already exists’. If an exception process to be started already exists, this does not lead in the case of the exception handling to a state message of the rho4 as it is the case for the programmed start of a process in BAPS. In this way, it is ensured that a second occurance of the trigger condition does not lead to a process error although the exception process has not completed the first exception handling yet.

In applications for which this case is realistic, an additional EXC_DETECT-call should occur after completion of all exception handlings to detect an occurance of the trigger condition during the running time of the exception process.

18.4 Deletion of exceptions

All programed exceptions are deleted automatically in the start up phase of the control.

If existing exceptions are to be deleted without the rho4 shutdown, this will be realized through new programming of the exceptions to be deleted with the transfer value NULL in parameter ERR_CODE_1.

By the interface signal 'Reset at interface' all programed exceptions are deleted.

18.4.1 Example: Deletion of exceptions

The program example 'ExcDel' deletes the previously programed exceptions number 1, 2 and 3.

```
119.   ;;CONTROL = RHO4
120.
121.   PROGRAM ExcDel
122.
123.   SPC_FCT : 47 = EXC_DEFINE ( VALUE INTEGER : EXC_NR
124.                               VALUE INTEGER : ERR_CODE_1
125.                               VALUE INTEGER : ERR_CODE_2
126.                                        TEXT : EXC_PR_NAME
127.                                     INTEGER : STATE )
128.
129.   INTEGER : RetState
130.   TEXT    : ExcPrName
131.
132.   BEGIN
133.   ExcPrName = '         '
134.   EXC_DEFINE (  1, 0, 0, ExcPrName, RetState )
135.   EXC_DEFINE (  2, 0, 0, ExcPrName, RetState )
136.   EXC_DEFINE (  3, 0, 0, ExcPrName, RetState )
137.
138.   PROGRAM_END
```

19 Belt counter current value
(from version VO05 no more available)

The function of the belt counter current value is to convert 'points with belt value' into 'points without belt value' in belt-synchronous sections of the belt kind 4.
The user interface for the belt synchronization type 4 has a more user-friendly design from version VO05 on (see SPC_FCT: 53 = band area (...)).
Programming the belt counter current value is no longer necessary. Calling SPC_FCT : 51 = Belt_Calc leads to the runtime error message 'Inadmissible SPC_FCT', code 131840 from version VO05 on.

CAUTION
Executing programs with the version 5 programming interface on versions VO03x or VO04x can result in the first movement being executed at the wrong time in the SYNCHRON part (normally too early) or in the wrong point being approached in the last belt-synchronous movement.

Belt counter current value

Notes:

Automatic velocity adjustment

20 Automatic velocity adjustment for PTP movements

For PTP movements, the given velocities for the axes are deduced from the machine parameters via factors. For belt synchronous movements, the effect of an axis can be additionally strengthened or weakened through the belt movement.
The extent of this effect depends on 2 factors:

- The min. or. max. belt speed
- The position of the belt synchronous working area in the working area of the robot.

By calling SPC_FCT : 53 = belt_area (...), the axis geometry factors are **automatically** determined.
These can be read or overwritten with the SPC_FCT : 52 = Belt_Ptp_Fac.

Declaration

```
SPC_FCT:52=Belt_Ptp_Fac (VALUE INTEGER : Belt_No
                         JC_POINT : @Min_Ptp_Fac
                         JC_POINT : @Min_Ptp_Fac
```

Belt_No	Number of the belt for which the values are to be set (1..16) or read (–1..–16)
@Min_Ptp_Fac	Minimum axis geometry factor
@Max_Ptp_Fac	Maximum axis geometry factor

The axis geometry factors indicate by how many degrees or mm an axis moves when the belt is moved by 1 mm. They can be positive or negative.
The belt components must be set to the value 0.0.

Several kinematics

Variables of type JC_POINT are kinematic related. If this special function should be used for several kinematics, it must then be declared under several names e.g. BeltPtpFac01, but the same number (52 = ...) for each kinematic with kinematic related type JC_POINT (e.g. Robi_1.JC_POINT).

Example

```
SPC_FCT:52=BeltPtpFac01  (VALUE INTEGER:       Belt_No
                          Robi_1.JC_POINT: @MinPtpFac01
                          Robi_1.JC_POINT: @MaxPtpFac01)

SPC_FCT:52=BeltPtpFac02  (VALUE INTEGER:       Belt_No
                          Robi_2.JC_POINT: @MinPtpFac02
                          Robi_2.JC_POINT: @MaxPtpFac02)
```

Automatic velocity adjustment

Several belts within a kinematic

If a kinematic has several belts, the special function can be used with the same declaration for each belt of the kinematic.

Effects of the axis geometry factors

By calling SPC_FCT : 53 = belt_area (...), the axis geometry factors are redetermined.

The general request is that the axes that are moved through the belt, must be able to have a (clearly) higher velocity than the

max. belt speed * max. geometry factor

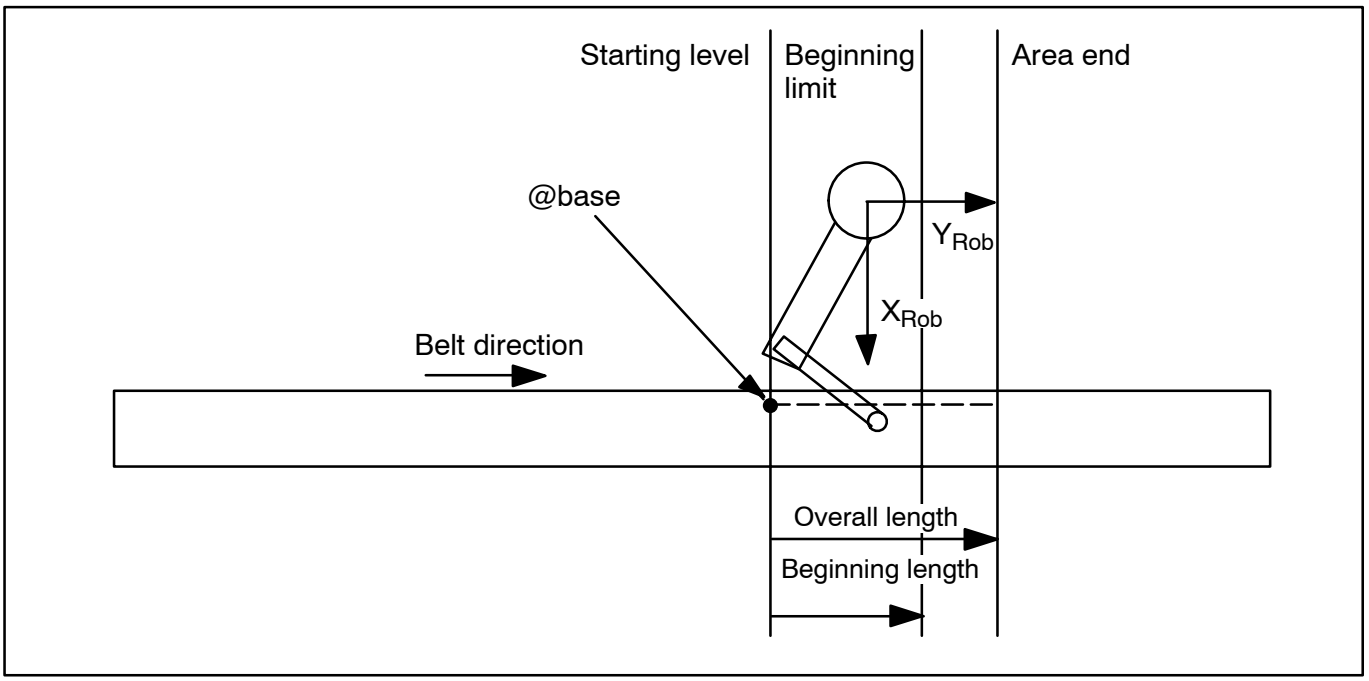
If an axis moves in direction of the belt, it can run with a higher velocity than indicated in the machine parameters. The given velocity can be increased by the

min. belt speed * min. geometry factor

If an axis moves behind the belt, it may only be given a smaller velocity than indicated in the machine parameters, so that the resulting velocity should not exceed the maximum velocity of the axis. The velocity must be reduced by the

max. belt speed * max. geometry factor

21 Belt-synchronous working area belt kind 4



Before using belt kind 4, the control has to know the belt-synchronous working area.

The **belt direction** is set with the P503 belt coupling factor machine parameter. In this image the belt is turned +90 degrees from the robot coordinate system, thus coord1=0, coord2=1, coord3=0.

The **starting level** is run via the machine coordinate point **@base** and is vertical to the belt direction. The **beginning limit** and the **area end** are parallel to the starting level.

A test is run during the first movement in the belt-synchronous section of belt kind 4 to see whether the final point of the movement is located between the starting level and the beginning limit.
If it has not yet reached the starting level, the execution of the movement will be delayed until the final point has passed the starting level.
If the final point of the first belt-synchronous movement has already exceeded the beginning limit the state message “Bs-begin limit, belt : b” code 19584 to 19599 is executed. This state message does not open the control READY contact. Thus a restart with SPC_FCT : 47 = exec_define (...) can take place.

Belt-synchronous working area belt kind 4

The Tool Center Point (TCP) is monitored at the area end when executing the belt-synchronous section of belt kind 4. If it is determined that the TCP will exceed or has already exceeded the area end, then the state message “BS–Take limit, belt: b” code 19712 to 19727 is executed. This state message does not open the control READY contact. Thus a restart with the SPC_FCT : 47 = exec_define (...) can take place.

The area end is not monitored during the last belt-synchronous movement (synchronisation and moving to a fixed position).

Declaration

```
SPC_FCT:53 =Belt_Area (VALUE INTEGER: BeltNo
                       JC_PUNKT:      @base
                       REAL:          FullLength
                       REAL:          BeginLength)
```

BeltNo	Number of the belt for which the values are to be set (1..16) or read (–1..–16)
@base	JC_POINT where the starting level goes through, or should go through
FullLength	Overall length of the belt-synchronous working area in the belt direction. The value must be greater than zero.
BeginLength	Latest start with belt-synchronous work. The value must be greater than zero and less than or equal to the overall length.

After the control has been booted, the set values for the belt area are: @base for all components 0.0, FullLength = 0.0, BeginLength = 0.0. Belt kind 4 cannot be used with these values. State message for SYNCHRON instructions “BS working area, belt : b” (code 131968 to 131983). They must be set at least one time with the SPC_FCT : 53 = belt_area (...) to useful values. This applies globally for all processes, until the next modification.

If a kinematic has several belts, there is then an individual belt area for each belt.

The belt area values that are currently valid can be read by calling up the SPC_FCT : 53 = belt_area (...) with negative belt numbers. The values will then be written in the transfer variables.

Belt-synchronous working area belt kind 4

The overall length must be greater than zero for modifications. Otherwise the state message “BS working area, belt: b” (code 131968 to 131983) will appear during the belt area instructions.
The beginning length must be greater than zero and less than or equal to the overall length. Otherwise the state message “BS working area, belt: b” (code 131968 to 131983) will appear during the belt area instructions.

All components for the machine coordination point @base must be defined. Otherwise the state message “point not defined”, code 141184, will appear.

If a process is taking place in the belt-synchronous section of belt kind 4 for the kinematics, the band area may not be changed, otherwise the state message “BS working area, belt : b” (code 131968 to 131983) will appear during the belt area instructions.

The machine coordination point @base has to be within the working area. Otherwise the state message “transf.err IRD–prog”, code 139520, will appear during the belt area instructions.

Five equidistant machine coordination points are internally set in the belt direction to the area end starting at the machine coordination point @base. These five machine coordination points also have to be within the working area for the kinematics. Otherwise the state message “transf.err IRD–prog”, code 139520, will appear during the belt area instructions.
From this total of six machine coordination points the geometric dependency of belt motions and axis motions is determined. The determined values can be read and changed via the SPC_FCT : 52 Belt_Ptp_Fac (...).

Several kinematics

Variables of type JC_POINT are kinematic related. If this special function should be used for several kinematics, it must then be declared under several names e.g. BeltPtpFac01, but the same number (53 = ...) for each kinematic with kinematic related type JC_POINT (e.g. Robi_1.JC_POINT).

Example

```
SPC_FCT:53=BeltArea01(VALUE INTEGER:           BeltNo
                            Robi_1.JC_POINT:   @base
                            REAL:              FullLength
                            REAL:              BeginLength)

SPC_FCT:53=BeltArea02(VALUE INTEGER:           BeltNo
                            Robi_2.JC_POINT:   @base
                            REAL:              FullLength
                            REAL:              BeginLength)
```

Belt-synchronous working area belt kind 4

Several belts within a kinematic

If a kinematic has several belts, the special function can be used with the same declaration for each belt of the kinematic.

22 Current belt speed

The current belt speed can be determined with this special function.

Declaration

```
SPC_FCT:54=Belt_V (VALUE INTEGER: BeltNo
                   REAL:   BeltSpeed
```

BeltNo	Number of the belt for which the belt speed should be read (1 to 16).
BeltSpeed	Current belt speed in mm/sec.

If the entered belt number is smaller than 1 and larger than the total number of the belts for all kinematics, then the state message ’Inadmiss. Belt–No’, code 147712, will appear.

In contrast to a belt position inquiry, the belt speed inquiry is directly executed with this special function, and is not synchronized with the handling of previous movements.

The belt speed is determined from the difference between two belt positions in an interpolation cycle. The value is thus rastered. This effect is increased with low belt speed and/or a low interpolation cycle.
A sliding mean value determination of the belt speed for 10 cycles can be activated with the option byte for the belt input. You receive the average belt speed when this special function is called up.

Current belt speed

Notes:

23 Changing the belt simulation speed

Belt movement can be simulated to test belt-synchronous applications. This is done with the RC input signal BELT_SIM_RCI (O32.7). All belt counters change their values then with the set belt simulation speed.

The belt simulation speed is set with the machine parameter P508. Every belt has ist own value. After the control has been booted, the machine parameter values are set.

The values for the belt simulation speed are temporarily changed with this special function. Thus, applications with varying belt speeds can be tested.

Declaration

```
SPC_FCT:55=Belt_V (VALUE INTEGER: BeltNo
                   VALUE REAL:    BeltSpeed
```

BeltNo	Number of the belt for which the belt speed should be changed (1 to 16).
BeltSpeed	New belt speed in mm/sec.

If the entered belt number is smaller than 1 and larger than the total number of the belts for all kinematics, then the state message ”Inadmiss. Belt–No”, code 147712, will appear.

The second parameter (BeltSpeed) must be –1.0 or greater than or equal to 0.0. Otherwise the state message “velocity not all.”, code 134784, will appear.
The machine parameters can be reactivated with the value –1.0.

Changing the belt simulation speed

Notes:

24 General functions

24.1 Kinematic-specific automatic/manual operation

The rho4 offers the possibility to move several kinematics in the automatic operation and simultaneously several kinematics in the manual operation.

There is no strict separation into automatic or manual operation.

When using this special function, the user has to take safety precautions to make sure that movements in the automatic operation will not endanger any operator working at the same time in the manual operation with another kinematic.

24.1.1 Signals and their meanings

RC inputs

Ct. No.	PLC symbol name	rho4 interface address	Signal description
245	KIN_A_MN_RCI	O30.5	Automatic/manual per kinematic
152	K01_A_MN_RCI	O19.0	Auto / manual, not kin. 1
156	K02_A_MN_RCI	O19.4	Auto / manual, not kin. 2
160	K03_A_MN_RCI	O20.0	Auto / manual, not kin. 3
164	K04_A_MN_RCI	O20.4	Auto / manual, not kin. 4
168	K05_A_MN_RCI	O21.0	Auto / manual, not kin. 5
172	K06_A_MN_RCI	O21.4	Auto / manual, not kin. 6
176	K07_A_MN_RCI	O22.0	Auto / manual, not kin. 7
180	K08_A_MN_RCI	O22.4	Auto / manual, not kin. 8
184	K09_A_MN_RCI	O23.0	Auto / manual, not kin. 9
188	K10_A_MN_RCI	O23.4	Auto / manual, not kin. 10
192	K11_A_MN_RCI	O24.0	Auto / manual, not kin. 11
196	K12_A_MN_RCI	O24.4	Auto / manual, not kin. 12
200	K13_A_MN_RCI	O25.0	Auto / manual, not kin. 13
104	K14_A_MN_RCI	O25.4	Auto / manual, not kin. 14
208	K15_A_MN_RCI	O26.0	Auto / manual, not kin. 15
212	K16_A_MN_RCI	O26.4	Auto / manual, not kin. 16

General functions

KIN_A_MN_RCI

If this signal is set, automatic/manual operation per kinematic is active and the kinematic-specific signals are determined to distinguish between automatic and manual operation.

K01_A_MN_RCI to K16_A_MN_RCI.

The signal AUTO_MN_RCI has no significance in this case.

If this signal is not set, the signal AUTO_HQ_RCI applies globally to all kinematics as before, the automatic/manual operation per kinematic is inactive. The signals K01_A_MN_RCI to K16_A_MN_RCI have no significance in this case.

Kxx_A_MN_RCI

xx=01 to 16 designates the kinematic (e. g. xx=03 means kinematic 3). This signal has the same significance as the global signal AUTO_MN_RCI referred to kinematic xx. These signals switch the kinematics xx each into the automatic or manual operation.

If KIN_A_MN_RCI is set, some of the signals obtain a somewhat different meaning. If the signal is not set, the meaning remains unchanged.

POWERRED_RCO, PLC addr I16.4 (power reduction)

This signal is only set to 0 when all kinematics are in the automatic operation and if there is no 'emerg. op. without RC'.

Otherwise the signal is on 1.

PHG_ACTV_RCO, PLC addr I16.7 (PHG operation and manual is active)

This signal is set to 1, if at least one kinematic is in manual operation.

Otherwise the same behaviour applies as before.

AUTO_MN_RCO, PLC addr I18.3 (AUTOMATIC/MANUAL, not)

This signal is set to 1, if all kinematics are in automatic operation.

Otherwise the signal is on 0.

General functions

24.1.2 Automatic mode

In principle, for the kinematic-specific automatic operation the same applies as for the global automatic operation. The reference points of all kinematics must have been approached before programs with travelling movements can be started.It is not necessary to switch the PHG inactive via mode 8 before switching-over from the kinematic-specific manual operation to the kinematic-specific automatic operation.

User programs can be started independently from the signal status of the individual kinematic-specific Kxx_A_MN_RCI. If a block is to be prepared which addresses a kinematic that is not in automatic operation, the runtime message 'inadmis.in manual op' will be displayed.

If during the processing of a kinematic-specific block the corresponding signal Kxx_A_MN_RCI is set from 1 to 0 (switching-over of kinematic xx from automatic to manual operation) the runtime message 'AUTO/MANUAL switch' will be displayed. After such a case, no program with travelling movements that uses the switched kinematic can be processed before having disabled or reset the corresponding program.

24.1.3 Referencing and setting

As far as referencing is concerned, only those axes can be started for referencing, the kinematic of which is in manual operation.

24.1.4 Teach In

Only those points can be taught in, the associated kinematic of which is in manual operation. Otherwise the same applies as for the global automatic/manual operation.

24.1.5 Testing

If during the test kinematic-specific blocks have to be executed (e. g. MOVE ...), it is tested whether or not the corresponding kinematic is in manual operation. If this is not the case, a message will be displayed. Otherwise the same applies as for the global automatic/manual operation.

General functions

24.1.6 Change to automatic/manual operation per kinematic

When changing from the global operation to an operation per kinematic the following signal sequences have to be followed in any case:

- The signals Kxx_A_MN_RCI have to be brought into the same status as the signal AUTO_MN_RCI 'automatic/manual, not'
- Then change the signal KIN_A_MN_RCI

Notes

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

To ensure a largely trouble-free operation, the selection and disabling of programs and the switching-over between the automatic and manual operation modes should be made via the PLC.

In the PLC, the corresponding programs should be disabled before switching from the automatic to the manual operation. Before selecting a program, it should furthermore be verified whether the corresponding kinematic is in automatic operation. When making use of this special function, it is reasonable to address in one program only one kinematic. This ensures a separation of the kinematics in the BAPS program.

24.1.7 Transformations with Master/Slave axes

In some coordinate transformations, there are master/slave axes.

If digital drives are used on this mechanics, e.g. Servodyn D, the following signal process is absolutely to be observed in the PLC program when the drives are released:

- If Drive On is set (DRIVE_AA_RCI = O16.7)
- or DRIVE_1_RCI = O40.0 bis DRIVE_24_RCI = O42.7

a pulse of the 'Emergency service without RC' must then follow (EMERGOP_RCI = O16.1).

The EMERGOP_RCI must be set on 0 at another place in the PLC program.

In the standard program this is done through:

```
A-EMERGOP_DI

=-EMERGOP_RCI
```

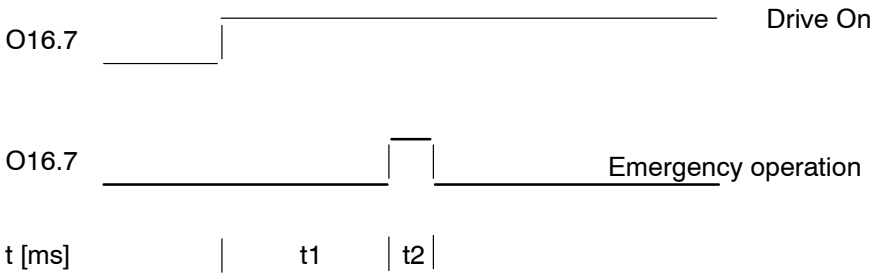
General functions

If this is not the case, EMERGOP_RCI remains because the following sequence is a self lock:

```
A-EMERGOP_RCI

O-SW_ON_TIME

=-EMERGOP_RCI
```

For the case that EMERGOP_RCI is not used:

```
A-SW_ON_TIME

=-EMERGOP_RCI
```



If this signal process is not kept, this can have the consequence that a permanent after running arises during the 'Drive On' release in a slave axis.

The time t1 must be greater than the sum from machine parameter P720 (= S–0–0505 release delay KSB) and P729 (= S–0–0206 Waiting time Drive On), however at least 2 clocks (P5).

The time t2 must be at least 2 clocks (P5).

Example sequence

```
DEF T100,-SW_ON_DELAY

DEF T101,-SW_ON_TIME

                                        ;-----------------------------------

A-DRIVE_K1_DI                           ;_DI for Drive On of all axes
                                        ;(e.g. 4) of the kinematic

=-DRIVE_1_RCI

=-BRAKE_1_RCI

=-DRIVE_2_RCI

=-BRAKE_2_RCI

=-DRIVE_3_RCI

=-BRAKE_3_RCI
```

General functions

```
=-DRIVE_4_RCI

=-BRAKE_4_RCI

                                          ;---------- Switch delay ----

L W T#500ms,A                             ;t1=500 [ms]

A   -DRIVE_K1_DI

SR  A,-SW_ON_DELAY

                                          ;---------- pulse ------------------

L W T#20ms,A                              ;t2=20 [ms]

A   -SW_ON_DELAY

SP  A,-SW_ON_TIME
```

24.1.8 Diagnosis at the PHG

The main menu of the PHG display indicates the message ’Automatic’ only when all kinematics are in automatic operation. If not all kinematics are in automatic operation, the message ’manual’ will appear.

Under mode 7.3.3 (diagnosis/system condit./actual op. mode), the current operation mode of each kinematic can be displayed.

General functions

24.2 Referencing with Servodyn GC

When using the rho4 in combination with the Bosch Servodyn GC and the digital drive interface CAN, the referencing modes described hereafter can be set.

24.2.1 Adjustment of the referencing mode

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

Example

Axis 1 with Servodyn GC drive

```
no. of servoboard:          1
no. of plug:               X51        ;CAN measuring system
modul-nb.:                  1
input on module:            1
kind of drive:              0         ;Servodyn GC
ref.-mode:                  0         ;referencing normal
                            1         ;referencing correctly oriented
                            2         ;referencing without approaching the
                                      ;zero crossing
                            3         ;referencing correctly oriented, without
                                      ;approaching the zero crossing
puls/rot.:                  16384
ms-valuation(nom. value):   16384     ;degrees resp. mm/motor revolution * cps
set value no.:              1
```

The number of subparameters are different at Servodyn GC and Servodyn D. The input of the referencing mode is equal for both kind of drives.

24.2.2 Normal

Ref. mode: 0

Function

After starting referencing the axis moves in direction of the reference point until the reference point switch is recognized. After having recognized the switch, the axis continues moving into the same direction to the next zero crossing of the transmitter. When reaching this position, the actual reference point value P207 is used as actual axis position. If a reference point offset P208 has been entered it is moved subsequently to end the referencing cycle.

General functions

Adjustment of the reference point switch

The reference point switch has to be adjusted during the normal referencing in such a way that it is away from the zero crossing, as far as possible.

Procedure

- move axis exactly onto the edge of the reference point cam
- run up rho4
- read axis position at PHG
- set home position offset
  - Servodyn GC:
    set home position offset at the drive booster
    Home position offset =
    (180 degrees + (CDR * VZ(ms.fact) * PHG value * GF)) MOD 360
  - Servodyn D:
    set home position offset via P401, subparameter home position offset
    Home position offset =
    180 degrees / GF + (PHG value MOD 360 / GF)

GF	Gear factor
VZ (ms.fact)	Sign of the measuring system valuation (P401)
CDR	CAN direction of rotation (MOOG parameter OD) +1 if positive –1 if negative

24.2.3 Correctly oriented

Ref mode: 1

This referencing mode has to be used for rotational axes which can make several revolutions and for which no measuring gear is used between rotational axis and reference point cam. An overtwisting of the axes and thus a damage of existing supply lines can thus be avoided.

Function

After starting referencing the axis moves in direction of the reference point until the reference point switch is recognized. After having recognized the switch, the zero position of the axis is computed by means of the read position value and of the gear factor. This position is approached in the next step. When having reached the zero position, the actual reference point value P207 is used as actual axis position.

If a reference point offset P208 has been entered it is moved subsequently to end the referencing cycle.

General functions

With a correctly oriented referencing, the revolution of the axis is deviated from the absolute position of the resolver at the reference point. When doing so, one resolver revolution is divided into n segments, whereby n = (1/value after decimal point of the gear factor).

Example

In case of an axis with a transmission ratio of 4.1:1, n results in 10. I. e. one resolver revolution is divided into ten 36 degree segments. The rho4.1 internal position value for one segment is between –18 degrees and +18 degrees. The zero point is thus in the middle of each segment. A revolution of 36 degrees at the resolver leads, in consideration of the transmission ratio of 4.1:1 to an axis movement of only 8.78 degrees. If the edge of the reference point cam is next to a segment transition, different reference point positions can be calculated since it is not ensured that the curve is always recognized in the same segment. For this reason, a particularly careful adjustment of the resolver zero point is required.

Adjustment of the reference point switch

For the correctly oriented referencing, the reference point switch must be adjusted in such a way that it is as close as possible to the zero crossing.

Procedure

- turn axis by hand approximately into the center position (the cables must be relaxed), exactly onto the edge of the reference point cam
- run zp rho4.1
- read axis position at PHG
- set home position offset
  - Servodyn GC:
    set home position offset at the drive booster
    Home position offset =
    (360 degrees + (CDR * VZ(ms.fact) * PHG value * GF)) MOD 360
  - Servodyn D:
    set home position offset via P401, subparameter home position offset
    Home position offset =
    360 degrees / GF + (PHG value MOD 360 / GF)

GF	Gear factor
VZ (ms.fact)	Sign of the measuring system valuation (P401)
CDR	CAN direction of rotation (MOOG parameter OD) +1 if positive –1 if negative

By means of this adjustment it can be achieved that the absolute zero point of the axis is close to the center position. At the same time it is ensured that the edge of the reference point cam is not close to a segment transition of the resolver.

24.2.4 Without approaching the zero crossing

Ref mode: 2

This resolver-specific referencing mode has to be used if the travel range of the machine is not sufficient to travel to the next transmitter zero point after recognition of the reference point switch.

Function

After starting referencing the axis moves in direction of the reference point until the reference point switch is recognized. When recognizing the switch, the position of the transmitter is read and the axis is stopped with the set delay ramp. The read transmitter position is offset against the actual reference point value P207 and is used as actual axis position. If a reference point offset P208 has been entered it is moved subsequently to end the referencing cycle.
The zero point of the transmitter is not approached with this referencing mode.

The parameter P207 actual reference point value and P208 reference point offset refer, as before, to the next zero point of the transmitter.

The meaning of the parameters P108 and P109, referencing speed, is maintained. It has to be made sure that P109 (1st reduced speed) is set in such a way that a deceleration of the axis between reference point switch and machine limit switch is possible.

P110 (2nd reduced speed) has no significance when referencing without approaching the transmitter zero point.

Adjustment of the reference point switch

The reference point switch must be adjusted in such a way that it is as close as possible to the zero crossing, for a resolver-specific referencing.

Procedure:

- move axis exactly onto the edge of the reference point cam
- run zp rho4.1
- read axis position at PHG
- set home position offset

General functions

- Servodyn-GC:
  set home position offset at the drive booster
  Home position offset =
  (360 degrees + (CDR * VZ(ms.fact) * PHG value * GF)) MOD 360
- Servodyn D:
  set home position offset via P401, subparameter home position offset
  Home position offset =
  360 degrees / GF + (PHG value MOD 360 / GF)

GF	Gear factor
VZ (ms.fact)	Sign of the measuring system valuation (P401)
CDR	CAN direction of rotation (MOOG parameter OD) +1 if positive –1 if negative

24.2.5 Correctly oriented without approaching the zero crossing

Ref mode: 3

Function

After starting referencing the axis moves in direction of the reference point until the reference point switch is recognized. When recognizing the switch, the zero position is calculated as described under point 24.2.3 and the axis is stopped with the set delay ramp. The calculated zero position is set off against the actual reference point value P207 and used as actual axis position. If a reference point offset P208 has been entered it is moved subsequently to end the referencing cycle.

Adjustment of the reference point switch

The adjustment of the reference point switch is made as described under point 24.2.3.

General functions

24.3 Reversing the direction of the Servodyn GC

To reverse the direction of a Servodyn GC (MOOG RMC) per machine parameter, both the measuring system converting factor and the set value output in P401 of the corresponding axis have to be entered as negative value.

Example

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

Old machine parameter P401, axis 2

A_2 servoboard	1
A_2 CAN plug	X51
A_2 CAN module No.	1
A_2 CAN input	2
A_2 kind of drive	0
A_2 ref. mode	0
A_2 pulses/rotation	16384
A_2 ms.fact	16384.0
A_2 command output	2

New machine parameter P401, axis 2

A_2 servoboard	1
A_2 CAN plug	X51
A_2 CAN module No.	1
A_2 CAN input	2
A_2 kind of drive	0
A_2 ref. mode	0
A_2 pulses/rotation	16384
A_2 ms.fact	**–16384.0** ⇐ new value
A_2 command output	**–2** ⇐ new value

☞ **If only one of the two values is reversed the axis will no longer get into position with one travelling movement!**

General functions

24.4 Limitation of the axis speed in linear operation

In case of linear or circular movements high axis speeds can occur with a definitely fixed path speed. This is due to the coordinate transformation, particularly close to singularities (e. g. extended position of a SCARA robot).

The resulting axis speeds can in case of linear movements either be monitored

- only in manual operation or
- in the manual and automatic operation

The monitoring is made in the manual operation by means of the high JC jog speeds set in the machine parameter P114 and in the automatic operation by means of the max. axis speeds stipulated in the machine parameter P103.

If an axis exceeds its set max. value, an interpolator stop will be released and all axes move at reduced speed until the critical speed range is left again. The movement will not be aborted.

To avoid speed jumps on the one side and keep on the other side the path deviation as small as possible, a fine interpolation is made for the critical axis (axes). This will cause a speed reduction of all axes if the speed of one axis is exceeded. The amount and duration of the reduction results from the ratio of the set maximum and calculated set axis speed.

Example

Max. speed: 15 mm/clock (scaled to mm/clock)

Default: 40 mm/clock

- 40/15 = 3 (integer division); i. e. 3 clocks are required to pass the specified path without exceeding the max. speed.
- During these 3 clocks, always the same path is specified as set value, i. e. 40/3 = 13.33 mm/clock.
- This speed reduction to 33.33 % of the default is made for all axes of the kinematics.
- When exceeding the set max. speed, neither the set or programmed path speed nor the programmed path (in world coordinates) can be kept!
- Both 'feed allow' and 'feed hold' only act through the normal interpolator. I. e. during the down slope, the values determined by the interpolator with switched-on axis monitoring are possibly put out reduced. As a result, the time until the stop of the machine will be extended, but the covered path remains independent from the axis monitoring.
- In automatic operation, the last interpolation interval of a travel block can for control-internal reasons not be reduced. The unmonitored section resulting from this, results in mm/clock from the programmed speed and the set clock time (machine parameter P5).

General functions

Machine parameter for the limitation of the axis speed

The desired axis speed monitoring is set by means of parameter P306, subinquiry 6:

WC axis monitoring	
0	no monitoring
1	monitoring with movement abort –> axis speed too high
2	only in manual operation monitoring with limitation of the axis speeds –> interpolator stop in the automatic operation there is no monitoring
3	in the manual and automatic operation monitoring with limitation of the axis speeds. The limitation acts on the whole movement except the last interpolation cycle. There is no limitation in the last interpolation cycle. This setting can lead to a jerk movement at the end of the set when using PROGR_SLOPE.
4	in the manual and automatic operation monitoring with limitation of the axis speeds. The limitation acts on the whole movement.
5	If an axis exceeds its maximal value, so the axis is limited exactly to this value. All other axes are reduced by an equal proportion. Thereby it is warrented, that the programmed path is followed at the best. If several axes exceed its maximal values, the reduction is adapted to the axis, which exceeds its maximal value most of all. This axis is reduced additionally to its maximal acceleration (machine parameter P104).By the exact reduction to the maximal value a constant limitation is warrented, i.e. no speed jumps arise. The maximal axes speed is load with the actual VFactor; i.e. changes of the VFactor affects directly the limitation of the maximal value.

The following is to be heeded:

- If the axis limitation occurs during the movement, the velocity at the TCP does not match the programmed velocity. During the axis limitation, there are path inaccuracies.

General functions

24.5 Interface signals for axis velocity monitoring

For the axis velocity monitoring, in the internal interface to the PLC at the RC inputs 2x16 Bits from version VO 04H are available.

Definition in the symbol file:

Name	PLC address	Interface description
SD_CLD01_RIE	O61.0	Safety door closed kinematic 1
SD_CLD02_RCI	O61.1	Safety door closed kinematic 2
:	:	:
:	:	:
SD_CLD16_RCI	O62.7	Safety door closed kinematic 16
PK_01_RCI	O63.0	confirmation key pressed kinematic 1
PK_02_RCI	O63.1	confirmation key pressed kinematic 2
:	:	:
:	:	:
ZPK_16_RCI	O64.7	confirmation key pressed kinematic 16

The O61.0..O62.7 (Safety door closed) are not evaluated in the operating system of the rho4. They can be used in the PLC program.

In CANopen drives, the corresponding kinematic signal 'Safety door closed' (from O61.0..O62.7) and the corresponding kinematic signal 'Confirmation key' (from O63.0..O64.7) in the additional control word are transferred to the drive.

24.5.1 Velocity reducing

If the signal 'Confirmation key' of a kinematic is set (O63.0..O64.7), the axis velocities of this kinematic are limited in the linear operation to the high JC Jog velocities available in the machine parameter P314. The monitoring occurs both in manual and automatic operation on these maximum values.

If the axis limitation is active during a movement, the velocity at the Tool Center Point (TCP) does not match the programmed velocity, resulting in path inaccuracies.

General functions

24.6 Manual axis as endless axis

In case of endless axes a distinction is made between main and manual axes. This is set with the machine parameter P303:

Axis type		
P303 =	0	normal main axis
	1	normal manual axis
	2	at present without significance
	3	endless axis as main axis
	4	endless axis as manual axis

Example

A kinematic with five endless axes, all having a modulo value of 360 (P311). There are one master and four slave axes.

Zero is set as robot type (P306: RTYP = 0). The master axis must be the first axis of the kinematic.

The master axis is defined as main axis (P303=3) and the slave axes as manual axes (P303=4):

Axis 1	P303	=	3	endless axis as main axis
Axis 2	P303	=	4	endless axis as manual axis
Axis 3	P303	=	4	endless axis as manual axis
Axis 4	P303	=	4	endless axis as manual axis
Axis 5	P303	=	4	endless axis as manual axis

If now a LINEAR movement is made with the specified path speed, the speeds of the slave axes depend on the speed of the master axis:

```
V=500

MOVE LINEAR VIA (20,    0,   0, 360,   0)

MOVE LINEAR VIA (90,  120,   0,   0,   0)

MOVE LINEAR VIA (105, 150,   0,   0,   0)

MOVE LINEAR VIA (190, 300, 160,   0, 180)

MOVE LINEAR VIA (200, 320, 180, 160, 240)

MOVE LINEAR VIA (220, 360, 260, 200, 360)

MOVE LINEAR VIA (280,   0, 360, 260,   0)

MOVE LINEAR VIA (360,   0,   0, 320,   0)
```

General functions

The master axis (axis 1) travels the entire movement clock at the constant speed of 500 degrees/s. The speeds of the slave axes are coupled to this.

General functions

24.7 Working room monitoring for cartesian kinematics

In applications with cartesian kinematics, it is possible to perform a relatively safe collision consideration concerning static obstacles for the most movements occuring in pratice at the moment of the preparation of the set.

In this way, it is possible to recognize in time at least gross programming errors arising through wrong teaching or leaving out a point. A large part of the expensive collision damages attributed to these causes can be avoided. Example: taking parts from injection moulding machines in the moving area of the handling robot.

To avoid collisions, locked cuboid-formed workspaces monitored at the moment of the set preparation of a movement instruction can be programmed from version VO04H in the rho4. The control checks the coordinates of the tool center points (TCP) on a straight line in the space that is clearly defined through the start and target point of the movement instruction. Since in cartesian systems, an approximately straight path also results in the PTP operation, a reliable protection can be reached by considering tolerances also with PTP interpolation.

☞ **For classic robot applications (articulated arm robot), the extension stage is not appropriate. A possibility would be a limited use with programs containing in areas with collision risk only linear movements and space coordinate points.**

24.7.1 Locked workspaces

programming

Up to 32 locked workspaces can be programmed as cuboids that can be allocated to different kinematics. A workspace can only be used by a kinematic and must be defined in the WC system of this kinematic.

In applications with several kinematics having a common obstacle, a workspace must be defined for every kinematic. The limits are entered as minimum and maximum authorized X,Y,Z coordinates in the coordinate system of the concerned kinematic.

The allocation to the kinematic is performed through the entry of the kinematic number in the data set of the workspace. If the kinematic number of a data set =0, this means that the concerned workspace is not engaged yet.

The limit values of the workspaces can be set and read per machine parameter and library function.

When setting per library function, the new limit values are immediately effective after executing the function. The next movement set to be prepared is checked for the new limit values.

It must be remembered that the move set prepared up to that moment were checked with the old, previously valid limit values. This means that in case of a dynamic change of the workspace limits during the running time of a move process there still can be from the adoption moment of the new limit values up to 11 move sets with old limit values that become active only at a later moment.

To avoid collisions, all move processes are therefore stopped and reselected in the case of dynamic modifications of limit values or at least stopped with a WAIT–UNTIL–INPUT command until the library function for setting the workspace limit values is successfully completed.

The saving of the limit values in the machine parameters is done with the shut-down of the control.

Entry of the workspace data

The workspace data is entered with the help of the machine parameter **P39**, enabling programming and display of up to 32 locked workspaces.

Setting and displaying the workspace data with the PHG

The workspaces get automatically the numbers 1 to 32 during the PHG entry of the machine parameter P39. The workspace number is displayed in the PHG display (PHG mode 7.8.1, 7.8.2).

The locked area of a workspace has the form of a cuboid. The limits of this quader are defined by entering six space coordinate values (Xmin, Xmax, Ymin, Ymax, Zmin, Zmax).

Each workspace must be allocated to a fixed kinematic through the entry of a kinematic number. The monitoring of the different workspaces can be switched off through the entry of the kinematic number = 0.

For each workspace, 7 values are entered:

1	Kin-No	Number of the allocated kinematic
2	Xmin	smallest X-coordinate of the locked workspace
3	Xmax	highest X coordinate of the locked workspace
4	Ymin	smallest Y coordinate of the locked workspace
5	Ymax	highest Y coordinate of the locked workspace
6	Zmin	smallest Z coordinate of the locked workspace
7	Zmax	highest Z coordinate of the locked workspace

Authorized value ranges:

- For 1, Kin No integral values from 0 to 16 are allowed. Kin No = 0 means that the monitoring for the defined workspace is inactive.
- For 2 to 7 real numbers with value range –999999.00 to +999999.00 are allowed. The limit values describe locked workspaces (cuboids) with the unit [mm] in fixed space coordinate system of the kinematic allocated through 'Kin No'.

Setting and displaying workspace data with library functions

The setting and display of the workspaces can be performed through Windows applications and active BAPS processes via the following rho4 library functions:

- rMPSAllP0039
- rMPGAllP0039

The supply of the transfer parameters of these functions and the corresponding data types are described in the following header files (see also example application 'Switcher'):

- rmps.h
- rmpg.h
- rmps.inc
- rmpg.inc

☞ **Workspaces set or modified via library functions affect running move processes without a restart of the control unit being necessary.**

In running processes, all movement sets prepared up to the calling moment of the library functions are moved with the workspace data previously valid. The adoption of the workspaces set per library function into the machine parameters occurs with the shut down of the control.

General functions

24.7.2 Functioning of the workspace monitoring

The workspace monitoring is effective in the automatic operation. In the manual operation, there is no monitoring of the locked workspaces.

Monitoring in the set preparation

The control calculates in the set preparation if the line AB which is unequivocally defined in the space through the start point A and final point B of a movement set, has a point of intersection or contact in common with a side of the locked workspace. This calculation is performed for every move command and all workspaces assigned to the kinematic.

If a movement process recognizes the violation of a locked workspace in the next set, this process moves the kinematic up to the start point of the faulty set and then generates a running time error. In this way, the process assumes the state 'Error' and does not carry out any other activities any longer. Until its complete deletion, important data such as the source line of the set with collision risk can still be displayed.

In the state messages, the number of the violated workspace is also displayed.

The stopped user process must be deleted by the user and restarted after elimination of the cause.

Status message for collision recognition

The status message of a probable collision with one of the locked workspaces can be displayed at the PHG (PHG mode 7.2). The message shows the number of the violated workspace:

```
workspace violation.
WSP No:      2
(Code=137600)
```

WSP No	Number of locked workspace, the limits of which are violated. The process running the kinematic with collision risk stops at the starting point of the critical movement set, see also PHG mode 7.3.2 (process display). The QLL line number of the set with collision risk can be displayed here. The display of this information is also possible with the help of the rho4 library functions and with ROPS4.

General functions

24.7.3 Tolerance zones

At the moment of the set preparation, it is assumed for simplification that the path to be run from point A to point B is a straight line.

The path actually interpolated later by the control shows more or less strong deviations compared to this model – due to programmable options such as overshooting, different slope and interpolation types.

Moreover, the movement of the tool center points of the real machine show another deviation due to the after-running and mass inertia of the axes compared to the set path interpolated by the control.

For these reasons, a high safety can only be reached when the possible deviations are compensated through sufficient tolerance zones in the workspaces to be protected.

Interpolation type

The path interpolated actually only matches during the interpolation type LINEAR with the straight line calculated in advance in the room.

In sets with PTP interpolation, the user must plan enough tolerances in the protection zones since even for cartesian mechanics, path deviations from the ideal assumption of a straight line arise through tools (grippers).

In sets with CIRCULAR interpolation, the workspace monitoring is also active, but is only partly appropriate as a rough monitoring.

Since for this interpolation type too, a straight line between the start and final point of the set is assumed, the uncertainty of a collision prediction is in this case especially high.

Overshooting and slope type

Path deviations resulting from overshoot radia or different acceleration types must be taken into account through tolerances in the workspaces to be protected.

General functions

24.8 Direct approach of points in the teach-in mode

The function 'direct approach of points in the teach-in mode' permits an easy change of already taught points. When commissioning a packaging line, the picking and placing positions are e. g. already taught. The user wants to make minor corrections of the positions. Since the positions are far away from each other it is time-consuming to move manually from the picking to the placing position. If the user can move directly to the placing position by means of the move order, he can save a lot of time.

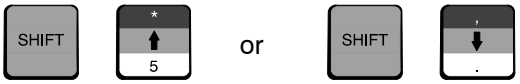
24.8.1 Operation

Select the teach-in mode.

The teach-in mode can be selected at the PHG via

- mode 3.1.3 (program. BAPS, progr. BAPS program, teach-in) or
- mode 4.2 (Define/Teach, Teach)

After having activated the teach-in mode, select the desired point. The selection is made by 'scrolling' with



respectively by entering the point name.

After having selected the point, actuate

Then press the permission key at the PHG , keeping it in the middle position. Keeping

additionally will start the movement to the desired point.

The robot moves to the given and already taught coordinate values of the point.

The movement is aborted immediately when the <enter> or permission key is no longer pressed. The movement can only be started by a reselection as described before.

General functions

PHG display:

MOVE
@POSITION

Execute with: ENTER

If the movement should be made linear, in addition to

, the key

at the PHG has to be actuated.

PHG display:

MOVE LINEAR
POSITION2

Execute with: ENTER

Speed and acceleration are derived from the machine parameters for JOG speeds, group P100.

The speed override via VFACTOR, AFACTOR and DFACTOR can be set by means of mode 11.4, 11.5 resp. 11.6 'help functions' and via interface.

With the key combination

+

it is possible to switch to the position display during the movement.

When the robot has reached the point, a new value can be taught in for the selected point.

General functions

24.8.2 Overwrite protection of already teached-in points

During the teaching-in of points, the user is, when pressing

informed by the message 'Pkt overwrite: ENTER' that the currently selected point will be overwritten when pressing

again. When actuating any other key, the old point value will remain unchanged.

☞ **The direct approach of a teach point is to be made linear. If the robot, e. g. SR6, can be changed from right-arm operation to left-arm operation, i. e. has a sign change on axis 1, the rho4 will display the message 'pt.n.reach.w.intpol'. To move to the desired point, PTP has to be moved in this case!**

General functions

24.9 Configuration identification for PLC program

With the configuration identification, the user is given the possibility to use the same PLC program for different machine configurations.

Machine parameter P2 defines the configuration identification 'machine configuration'.

For this parameter, a value between 0 and 255 can be entered. The parameter default is 0.

The value in P2 is put out at the RC-internal interface 8-bit coded and can be used in the PLC program according to the program flow control.

Signal assignment of the RC outputs

Ct. No.	PLC symbol name	rho4 interface address	Signal description
208 to 215	MA_TYP_0_RCO to MA_TYP_7_RCO	I 26.0 to I 26.7	**Machine configuration, bits 0 to 7:** The value set in machine parameter 2 is transmitted coded to the interface.

Example

For two rho4.1 controls with different machine configurations, a PLC program has to be created that can be used for the different machine configurations.

Control	Machine configuration
1	one kinematic with 3 axes
2	two kinematics with 4 axes each

Parameter P2 of the 1st control contains a 0 for the machine configuration; P2 of the 2nd control contains a 2.

General functions

Extracts of the corresponding locations in the PLC program

Syntax	Description
AN -MA_TYP_0 AN -MA_TYP_1 AN -MA_TYP_2 AN -MA_TYP_3 AN -MA_TYP_4..7 = -USER_42_RCI	decoding of the machine configuration put out by the control USER_42_RCI = control 1 (P2 = 0)
AN -MA_TYP_0 A -MA_TYP_1 AN -MA_TYP_2 AN -MA_TYP_3 AN -MA_TYP_4..7 = -USER_43_RCI	 USER_43_.RCI = control 2 (P2 = 2)
AN -M1_END4_P A -USER_43_RCI = -USER_7_RCI	limit switch 4th axis + control 2, kin. 1
AN -M1_END4_N A -USER_43_RCI = -USER_8_RCI	limit switch 4th axis – control 2, kin. 1
AN -M2_END4_P A -USER_43_RCI = -USER_15_RCI	limit switch 4th axis + control 2, kin. 2
AN -M2_END4_N A -USER_43_RCI = -USER_16_RCI	limit switch 4th axis – control 2, kin. 2
A -USER_42_RCI JPC -JOG_ST1	jump to JOG keys control 1
A -DRGRP_1 A -JOG1MPHG AN -JOG1PPHG = -JOG_1_M	JOG keys at PHG control 2 jogging 1st axis neg. at RC
A -DRGRP_2 A -JOG4PPHG AN -JOG4MPHG = -JOG_8_P . A -LOGIC1 JPC -BRAKEEN	 jogging 8th axis pos. at RC

General functions

Syntax		Description
	-JOG_ST1	JOG keys at PHG control 1
A	-DRGRP_1	
A	-JOG1MPHG	
AN	-JOG1PPHG	
=	-JOG_1_M	jogging 1st axis neg. at RC
	.	
A	-DRGRP_1	
A	-JOG3PPHG	
AN	-JOG3MPHG	
=	-JOG_3_P	jogging 3rd axis pos. at RC
	-BRAKEEN	time for brake enable M1
A	-SVE_BETN	
AN	-M1_DRIVE	
R	-TRAVOFFS1	

General functions

24.10 Output of A–V–D–FACTOR at the rho4 interface

The actual effective global Acceleration– Deceleration– and Velocity–Factors are given out at the rho4 interface. The conditions described in chapter ”function description” must be considered.
For example, the description for the VFACTOR follows. The corresponding functionality applies also to the AFACTOR and DFACTOR.

Function description

The global VFACTOR is written dynamic in a chronological screen of the clock start–time to the RC–Outputs VFACTOR_0_RCO ... VFACTOR_7_RCO of the rho4 interface. These RC–Outputs are put out cyclical after each PLC–IO– exchange in the PLC of the rho4.1 , resp. the PCLrho4.0 of the rho4.0.

Signals at the rho4 interface

The following signals are put out at the rho4 interface:

Current No.	Symbol PCL	rho4 Standard Interface Address	Signal description
776	VFACTOR_0_RCO	I 97.0	VFACTOR output at the interface Bit 0 (value 1)
.	.	.	
783	VFACTOR_7_RCO	I 97.7	VFACTOR output at the interface Bit 7 (value 128)
784	AFACTOR_0_RCO	I 98.0	AFACTOR output at the interface Bit 0 (value 1)
.	.	.	
791	AFACTOR_7_RCO	I 98.7	AFACTOR output at the interface Bit 7 (value 128)
792	DFACTOR_0_RCO	I 99.0	DFACTOR output at the interface Bit 0 (value 1)
.	.	.	
799	DFACTOR_7_RCO	I 99.7	DFACTOR output at the interface Bit 7 (value 128)

General functions

Range of global VFACTOR

The range for the output of the global VFACTOR depends on the range global VFACTOR in machine parameter **P23 Range global VFACTOR**.

The range for the output of the global AFACTOR and DFACTOR depends on the range global A–/DFACTOR in machine parameter **P22 Range global A–/DFACTOR**.

Example:
Machine parameter **P23 Range globale VFACTOR** ist set to the following values:

P23 GLOB.RANGE VFACT		**P23 GLOB.RANGE VFACT**	
minimum	**0.001**	**maximum**	**9.999**

The adjusted VFACTOR is 7.0

From them, it results a resolution per Bit at the interface of

(maximum – minimum) / 255 = 0,03920784

Calculated value:

(VFACTOR – minimum) / 0,03920784 = 178,5102104

Rounded value:

178,5102104 rounded to 179

Value output at the rho4 interface:

VFACTOR_0_RCO = 1 (value 1)
VFACTOR_1_RCO = 1 (value 2)
VFACTOR_2_RCO = 0 (value 4)
VFACTOR_3_RCO = 0 (value 8)
VFACTOR_4_RCO = 1 (value 16)
VFACTOR_5_RCO = 1 (value 32)
VFACTOR_6_RCO = 0 (value 64)
VFACTOR_7_RCO = 1 (value 128)

Note: The value range is only valid with activated option **A/D/V–Factor IF**

PLC und PCLrho4.0 programs

The signals listed above are included into the PLC– and PCLrho4.0 standard programs and are available from version VO06B.

General functions

24.11 Multifunction I/O on Servodyn D drive modules

The digital output OUT1 on each Servodyn D drive module connected to the rho4 is transferred from the rho4 interface to the drive modules, MF_OUTxx_RCI. Furthermore, the digital inputs IN1 to IN4 of the Servodyn D drive modules are transferred into the rho4 interface, MF_INPUTxx_RCO.

Integration of the Servodyn D I/O into the PLC interface

The assignment of the digital input and output signals on the Servodyn D drive boosters to the PLC interface is made via machine parameter P36, multifunction I/O.

Addresses for multifunction inputs	
0	Basic adjustment The Servodyn D inputs are copied to the given input addresses, I81.0 to I92.7.
16 to 175	Input of the start address for the Servodyn D inputs The length of the required blocks is determined by the number of axes. Block length = number of axes * 4 bits.
–1	Switching-off of the multifunction inputs The digital inputs of the drive boosters are not copied to the PLC interface.

Addresses for multifunction outputs	
0	Basic adjustment The given output addresses, O56.0 to O58.7, are copied to the Servodyn D outputs.
16 to 175	Input of the start address for die Servodyn D outputs The length of the required blocks is determined by the number of axes. Block length = number of axes * bits.
–1	Switching-off of the multifunction outputs The digital outputs of the drive boosters are not operated.

Example

2 kinematics:	1st kinematic	3 axes
	2nd kinematic	5 axes
	Total	8 axes

Adjustment P36: address for multifunction inputs: 85

General functions

PLC input addresses		
Axis 1	In1	I85.0
	In2	I85.1
	In3	I85.2
	In4	I85.3
Axis 2	In1	I85.4
	to	to
	In4	I85.7
Axis 3	In1	I86.0
	etc.	etc.
Axis 4	In1	I86.4
to	etc.	etc.
Axis 8	In1	I88.4
	In2	I88.5
	In3	I88.6
	In4	I88.7

Adjustment P36: address for multifunction outputs: 60

PLC output addresses		
Axis 1	Out1	I60.0
Axis 2	Out2	I60.1
Axis 3	Out3	I60.2
Axis 4	Out4	I60.3
Axis 5	Out5	I60.4
Axis 6	Out6	I60.5
Axis 7	Out7	I60.6
Axis 8	Out8	I60.7

General functions

24.12 Asynchronous inputs

In previous versions of the operating system, always an internal synchronization between the block-preparing movement task, i. e. a user process, and a kinematic-specific block processing, i. e. the control of the axis, is carried out in case of an input inquiry, such as in the 'IF .. THEN' command. The control attempts to prepare as many as possible, up to 11, kinematic-specific blocks of a process in parallel with the execution of a travel block. The block preparation normally precedes the block execution. The calculation of arithmetic terms are thus executed at a much earlier time than the processing of MOVE blocks located in the BAPS program before these arithmetic operations take place. This asynchronity must be eliminated when using inputs, i. e. a synchronization is necessary as soon as an input is read. This function 'asynchron inputs' permits to maintain this asynchronity for input inquiries. This means that inputs defined as asynchronous are read and processed immediately at the time of the block preparation. This normally leads to a time shifting in handling several BAPS commands. Asynchronous inputs should for this reason only be used when other BAPS solutions are insufficient.

Declaration

The declaration of asynchronous inputs is made by indicating an offset of 1000 in the channel number. The following table shows the admissible input types and their channel numbers for synchronous and asynchronous input inquiries.

Input type	Channel number (synchronous)	Channel number (asynchronous)
BINARY	1 to 199	1001 to 1199
INTGER	401 to 416	1401 to 1416

The use of asynchronous inputs in BAPS is shown hereafter by means of the group of the binary inputs.

The other input types are to be used in the same way. The declaration of asynchronous binary inputs is made via the channel numbers 1001 to 1199. Physically the same inputs are addressed with these channel numbers as with the channel numbers 1 to 199.

INPUT:1001 = ASYN_INP_1, 1002 = ASYN_INP_2

Asynchronous inputs in the BAPS program

For the use of inputs in the BAPS program, the same BAPS syntax applies as for normal binary inputs. It is also allowed to declare in this case the same physical input once as normal synchronous input and additionally as asynchronous input. In the BAPS program, the input can be inquired either with or without synchronization.

Example

```
1     PROGRAM syn_asyn
2     INPUT:1 = i1, 1001 = i1_asyn
3     BINARY:bin_1, bin_2
4     BEGIN
5     MOVE LINEAR VIA P1
6     bin_1 = i1
7     MOVE LINEAR VIA P2
8     MOVE LINEAR VIA P3
9     MOVE LINEAR VIA P4
10    bin_2 = i1_asyn
11    MOVE LINEAR VIA P5
12    MOVE LINEAR VIA P6
13    MOVE LINEAR VIA P7
14    PROGRAM_END
```

When used first, the binary input 1 is read synchronously to the program flow and assigned to variable bin_1 in the example. Block preparation is stopped between line 5 and 7. The reading of the inputs is carried out 2 interpolation clocks (machine parameter P5) before the set value output of end point P1, synchronously to the movement. The blocks P2, P3, P4, P5, P6, P7 are only prepared after having read the inputs.

When used for the second time, the input is used as asynchronous input. The block preparation does in this case not wait until P4 is approached. The input is already read during the travel of the kinematic to point P2. The blocks P5, P6, P7 can also be prepared while the kinematic moves to point P2, if the user task is assigned sufficient computing time.

Application example of asynchronous inputs

When operating the control with high-speed packaging machines, this function offers a possibility to react after an adjustable number of travel blocks to external events, without affecting set values. In this application the course of a curve is followed by means of the special function 'MOVE_FILE' stored in a .bnr file, see section 16.2. The values of the file result in combination with the set interpolation clock in a firm speed profile. When using an asynchronous input it is possible to change in the travel loop of the user program without jerk to another .bnr file, i. e. without affecting set values. The block lead, see chapter 17, is limited to 5 blocks in the example. This means that the speed change will only be effective after 5 MOVE_FILE blocks. If, instead of the asynchronous output, a normal binary output is used, the probability that the set values are affected is very high since in this case the block preparation synchronizes the reading time of the input.

When using asynchronous inputs, the following rule applies:

The higher the block lead has been set, the lower the probability that the set values are affected and the longer it takes until the input is effective on the kinematic blocks.

General functions

Block lead	Probability that the set values are affected	Response time to asynchronous input
High bock advance	low	slow
Small block lead		fast

Example

```
;;CONTROL=rho4                                    ;compiler instructions

;;KINEMATICS:(1=p_machine)

;;JC_NAMES=A1,A2,A3,A4

PROGRAM asyn_on

SPC_FCT:45=move_file(VALUE INTEGER:              kin_no
                     BNR_FILE:                   curve_x
                     VALUE INTEGER:              basis
                     JC_POINT:                   @modulo_flag
                     VALUE ARRAY[1..6] INTEGER: reserve)

SPC_FCT:46=blck_advance(VALUE INTEGER: kin_no
                        VALUE INTEGER: block_number)

INPUT: 2=I2,1001=v_switch

BNR_FILE: MF_1,MF_2,MF_3,
          MF_1_s,MF_2_s,MF_3_s,                        ;slow
          MF_1_f,MF_2_f,MF_3_f                         ;fast

JC_POINT:@mod_flag

ARRAY[1..6] INTEGER: reserve
            INTEGER: i

CONST: PC_raster  =2;
       head_length=80

BEGIN

   ;;KINEMATICS=p_machine

   blck_advance(1,5)                          ;5 blocks of lead for kinematic 1

   MOVE WITH V_PTP VIA @(57,0,12,12)

   READ_BEGIN MF_1_s, head_length             ;open .bnr files

   READ_BEGIN MF_1_f, head_length

   READ_BEGIN MF_2_s, head_length

   READ_BEGIN MF_2_f, head_length

   READ_BEGIN MF_3_s, head_length

   READ_BEGIN MF_3_f, head_length
```

General functions

```
   LOOP:                                    ;travel clock
      IF v_switch=1
      THEN
      BEGIN                                ;files for high speed depart
         @mod_flag=@(1.0,0,0)
         move_file(1,MF_1_f,PC_raster,@mod_flag,reserve)
         @mod_flag=@(0.00.10.1)
         move_file(1,MF_2_f,PC_raster,@mod_flag,reserve)
         @mod_flag=@(0,1,0,0)
         move_file(1,MF_3_f,PC_raster,@mod_flag,reserve)
      END
      ELSE
      BEGIN                                ;files for low speed depart
         @mod_flag=@(1.0,0,0)
         move_file(1,MF_1_s,PC_raster,@mod_flag,reserve)
         @mod_flag=@(0.00.10.1)
         move_file(1,MF_2_s,PC_raster,@mod_flag,reserve)
         @mod_flag=@(0,1,0,0)
         move_file(1,MF_3_s,PC_raster,@mod_flag,reserve)
      END
   IF I2=0
   THEN JUMP LOOP
   CLOSE MF_1_s                        ;close .bnr files
   CLOSE MF_2_s
   CLOSE MF_3_s
   CLOSE MF_1_f
   CLOSE MF_2_f
   CLOSE MF_3_f
PROGRAM_END
```

General functions

24.13 DSS coupling via the drive point of the rho4

From the DSS version V2.02.04 (DSS~Diagnostic and Service System) and the rho4 version VO04H, a direct coupling to the rho4 is possible via the drive interface (SERCOS, CANopen). Up to 4 bus participants can be selected at the same time. Additional hardware or changing the connection of a V24 cable is not necessary. The required DSS software can be directly installed on the rho4.1.

For the remote diagnostic, the DSS software can be installed on a diagnostic PC to communicate within a network with the rho4. With a rho4.1, a gateway must be started on the rho4.1 in analogy to the online ROPS coupling. For the communication, the rho4 must be completely started and the field busses must be initialized with their participants. For the drive startup without rho4 there is also the possibility to communicate via V24 connection.

24.13.1 Realization

The DSS is able to communicate via the rho4 with CANopen and SERCOS participants. The DSS checks during the installation the availability of the file 'rho4fkt.dll'. If the file is on the local computer, the file path is marked for the DSS and during the start of the DSS, a connection to the rho4 is also available.

The interface parameters are entered in a dialog box:

Host Name	rho4
Port:	rho_Functions_1 (also possible rho_Functions_2, rho_Functions_3, rho_Functions_4)
Timeout	e.g. 100 s

After the interface is quit, the connection to the control unit is established. If it is successful, a selection box appears with the available CANopen and SERCOS participants. From the latter, it is possible at present to select at the same time 4 ones for the processing through the DSS.

Besides the local operation, the DSS can also use as basis the rho4 within a network. The following modifications are necessary:

- On the computer, on which the DSS is to be installed, the rho4fct.dll must be present in the directory 'C:\Winnt\system32'.
- The product version of the rho4fkt.dll under Properties\Version\Product version must be higher than VO04E.

With a rho4.1, the file gateway.ini (as default in the directory \Bosch\Rho4\WinExe) must be adapted as follows:

```
ConnetionNo = 1
1 = rho_Functions_1
```

After the saving of the file, the gateway (in the rho4 program group) can be started. On the computer with the DSS, the IP address of the rho4 can be entered into the file 'hosts' in connection with an alias name. After the start of the DSS, the rho4 connection is selected and its interface parameters modified. In the selection box for the host name, the alias for the IP address of the rho4 should be now entered. The further operation is the same as with the local coupling.

General functions

24.14 Deletion of write/read buffer

Introduction

The communication between two units via a serial interface often causes the problem that when starting the communication, characters of a communication aborted before or of switch-on errors can be stored, which can lead to malfunctions or misinterpretations.

It should therefore be possible to delete characters from out of the BAPS program stored intermediately in the control without having to use the interface disable signal.

24.14.1 Syntax

To do so, the language elements READ_BEGIN and WRITE_BEGIN used so far only for files are used as BAPS language elements.

```
READ_BEGIN(file_variable)[,line_number] resp.

WRITE_BEGIN(file_variable)[,line_number]
```

As file variable, a unit name (PHG, V24_1 to V24_4) or a file variable can be used to which a unit has been assigned by means of the ASSIGN function. If no file variable is indicated, the PHG is used by the BAPS compiler.

The line number that can be indicated in a file is not evaluated in this case.

24.14.2 Deletion of write buffer

When writing onto an interface, characters can be stored

- on the interface processor
- in intermediate storages of the output process belonging to the process
- in intermediate storages of the output process belonging to other processes
- in intermediate storages of the communication partner.

Characters stored in intermediate storages of the communication partner cannot be deleted by the WRITE_BEGIN command. In such cases, commands have to be provided for the communication (reset, restart) that ask the communication partner to delete his stored data.

The characters stored in other output processes are not deleted. If these characters are to be deleted, the associated process has to be stopped. Outputs of the same process, which are already prepared, will be deleted.

Also outputs that have already been transmitted to the interface processor from the same process are deleted as far as possible. This then leads to cut-off outputs.

☞ **When calling the WRITE_BEGIN command, the condition of the unit is set to zero for this process.**

☞ **If the protocol 3964R has been set for this unit, the prepared outputs will be deleted.**

☞ **Outputs in the driver process will not be cut-off, they will be completed to restrict the complications on the driver level as much as possible. However, no repetitions required by the protocol will be carried out. If the data transmission has not started yet, the transmission will be ended after a successful establishment of the connection.**

☞ **An error in this transmission does not change the CONDITION; it will remain zero.**

☞ **When restarting the communication, it has to be taken into account that the driver process aborted with WRITE_BEGIN is for four seconds (repetition delay time) entitled to repetitions, provided that a repetition would have been required.**

24.14.3 Deletion of the read buffer

When reading from an interface, characters can be stored

- in intermediate storages of the communication partner (e. g. retained by handshake) or
- on the interface processor or
- in a driver process (3964R).

The characters stored in intermediate storages of the communication partner cannot be deleted by the READ_BEGIN command.

General functions

It can also happen that after execution of the READ_BEGIN command unexpected characters are received from the communication partner, e. g. replies to previously enabled actions with longer response or processing times.

Characters on the interface processor are only deleted if the interface assigned to the unit (PHG, V24_1 to V24_4) is not assigned otherwise, i. e. if no interface switching is active.

☞ **If a read buffer is deleted while data belonging together are transferred, misinterpretations of the remaining data or protocol errors can occur.**

☞ **If the protocol 3964R of a unit has been set, only the receiving buffer of the driver process will for this reson be deleted, but not the characters on the interface processor. These characters are received by the driver, which will interpret them according to the protocol.**

☞ **When calling the READ_BEGIN command, the condition of the unit will be set to zero for this process.**

Example 1

```
1 PROGRAM rcmaster
2 FILE: barcode_read
3 GLOBAL INTEGER:barcode
4 BEGIN
5    ASSIGN barcode_read,'V24_1.   '
6    READ_BEGIN barcode_read
7    WRITE barcode_read,'send number'
8    READ barcode_read, barcode
9 PROGRAM_END
```

General functions

Example 2

```
 1 ;;PROCESS_KIND=PERMANENT
 2 PROGRAM rcslave
 3 FILE:           master
 4 GLOBAL INTEGER: part_no,part_quant,deposit_no
 5 INTEGER:        cond_pc
 6 BEGIN
 7    ASSIGN master,'V24_1.   '
 8    READ_BEGIN master
 9    beg:
10       READ master,part_no,part_quant,deposit_no
11       cond_pc=CONDITION(master)
12       IF cond_pc<0 THEN
13       BEGIN
14        WAIT 0.5                    ;whether master still sends something
15        READ_BEGIN master
16       END
17    JUMP beg
18 PROGRAM_END
```

General functions

Example 3

```
 1 ;;PROCESS_KIND=PERMANENT
 2 PROGRAM write_beg
 3 INPUT: 1=break
 4 INTEGER: cond_v24_1
 5 BEGIN
 6    WRITE V24_1,'first output'
 7    cond_v24_1=CONDITION(V24_1)
 8    WRITE V24_1,'second', ..
 9    IF break=1 THEN
10    BEGIN
11      WRITE_BEGIN V24_1
12      WRITE V24_1,'--- break ---'
13    END
14 PROGRAM_END
```

Explanation to the WRITE_BEGIN program

In case of a break, the WRITE_BEGIN command deletes in line 11 the prepared output of line 8.

Without the CONDITION inquiry in line 7, the output would be deleted in line 6 as far as still possible. The output could then e. g. look as follows: 'fir—— break ——'.

General functions

24.15 rho4 coupling via CAN bus

For coupling several rho4 controls, the digital I/O signals are used via the CAN bus. Since each control generates its own clock, there is no possibility to synchronize all controls in a common clock. For this reason it is not ensured that an incoming telegram is received in each clock. To be able to carry out a data exchange between rho4 controls nevertheless, the time monitoring has been tuned down especially for this application.

Connection of the controls

CAN bus has to be reserved for the connection of the rho4 controls. No other units must be connected to this bus.

Adjustment of the identifiers

Reserved identifiers for the coupling of digital I/O modules:

- outputs: 471 to 480
- inputs: 541 to 550

For the coupling of rho4 controls, only the identifiers 471 to 480 are used. To be able to carry out a data exchange, the same identifiers must be set on the corresponding controls.

Data exchange between 2 controls

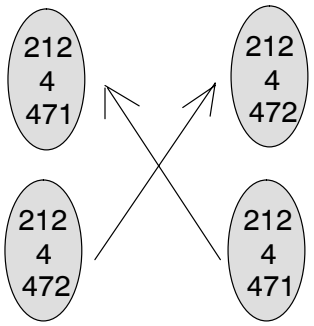
P31 address ranges of the CAN inputs

	Control 1	Control 2
Beg.addr.block 1	212	212
Length block 1	4	4
Identifier block 1	471	472

P32 address ranges of the CAN outputs

	Control 1	Control 2
Beg.addr.block 1	212	212
Length block 1	4	4
Identifier block 1	472	471

General functions



Here, the output signals bytes 212 to 215 of control 1 are transferred via identifier 472 to the inputs bytes 212 to 215 of control 2.

In reverse order, the output signals bytes 212 to 215 of control 2 are transferred via identifier 471 to the inputs bytes 212 to 215 of control 1.

As can be seen, identifiers which are at a control actually reserved for outputs are used as input identifiers. This adjustment has automatically the effect that the time monitoring for these inputs is set to 300 ms.

This monitoring time can, if required, be set via an option flag to other values, respectively the time monitoring can be switched off completely.

During the control boot up, the time monitoring remains at first inactive. It is only activated if all controls involved in the data exchange have responded. If a control has not responded within 60 seconds after the run-up of the fastest control, the break will be made with the message 'no inp-transm. CAN*x*,inp.-bl*. y* (x = number of the CAN bus, y = number of the input block).

The identifiers must be set in such a way that it is excluded that 2 controls are sending on one bus on the same identifier.

General functions

Notes:

25 Process-oriented functions

25.1 Axes which can be switched off

In some applications, e. g. jointing works it is necessary to take one or several axes out of the position control circuit to move these axes, e. g. by extermal means, such as a hydraulic or pneumatic cylinder. For this type of problem, the rho4 offers the possibility to switch off individual axes via axis-specific DRIVE ON signals.

These are the signals

- DRIVE ON all axes. Collective signal for all axes.
- DRIVE ON 1st to 24th axis. Axis-specific DRIVE ON signals.

☞ **The axis-specific DRIVE ON signals are only effective when the collective signal has not been set.**

25.1.1 Control-internal effect of the DRIVE ON signals

Drive on = 1 (axis switched on)

The axis is in the position control circuit. Travelling with foreign set values is not possible. Servo error monitoring is active.

Drive on = 0 (axis switched off)

The axis is not within the control circuit. The axis positions are refreshed internally. No servo error is released when the axis is moved by a foreign set value. When switching-on the axis again, the set position = actual position is set, proceeding with the current position value.

Process-oriented functions

25.1.2 Application possibilities

The possibility to switch one or several axes on or off via DRIVE ON signals exists both in the setting (manual) and the automatic mode.

Setting mode

When selecting the setting mode (PHG mode 2, manual), the user has the possibility to switch off DRIVE ON when jogging individual axes, which are not involved in the movement, i. e. these disabled axes can then be manipulated by influence from outside. When manipulating them by influencing them from outside, the current actual position of the axes is refreshed and shown on the display of the PHG.

Axes that are not switched off can be travelled in parallel via PHG operation in JC and WC. If, when doing so, a movement of an axis that has been switched off is released, the control will abort with the error message 'DRIVE ON not avail.'.

Automatic mode

In the automatic mode it is also possible to switching off individual axes. It must, however, be ensured that the DRIVE ON signals are switched ON and OFF synchronously to the running BAPS program. To do so, it is necessary to harmonize the BAPS and PLC programs accordingly. Some application examples are explained hereafter.

Safety instructions

If axes are moved with switched off DRIVE ON signal, a synchronization to the current actual position must take place before the next programmed axis movement, since otherwise set value jumps or servo errors can occur.

Read POS can be achieved by a dummy MOVE UNTIL or MOVE_REL UNTIL instruction, e. g. MOVE_REL TO sig = 1 TO P_var. In this case, the condition (sig = 1) must have been fulfilled before reaching the position programmed in P_var.

☞ **The instruction MOVE TO @POS, to read the current actual position cannot be used at this place.**

Process-oriented functions

Example 1

```
;;CONTROL=rho4                                ;compiler instructions

;;KINEMATICS:(1=sr6)

;;sr6.JC_NAMES=a_1,a_2,a_3,a_4

;;sr6.WC_NAMES=a_X,a_Y,a_Z,a_a

PROGRAM fsw_O3

OUTPUT:                                      ;user outputs (1 to 4) are copied in

1=driveon_01,                                ;the PLC program to the DRIVE ON

2=driveon_02,                                ;input signals 1st to 4th axis if the

3=driveon_03,                                ;user output 17 (auto_don) is set.

4=driveon_04,                                ;If auto_don is not set, the

17=auto_don                                  ;signals are set from the outside via
                                             ;switches, see associated PLC program

INPUT:

25=drive1_ok,                                ;feedback of the DRIVE ON signals

26=drive2_ok,

27=drive3_ok,

28=drive4_ok

BEGIN

   ;;KINEMATICS=sr6

   driveon_01=1                              ;switching-over to automatic

   driveon_02=1                              ;DRIVE ON control via user outputs

   driveon_03=1

   driveon_04=1

   auto_don=1

   loop:

      WAIT 0.5                               ;wait until all drives are switched on

      IF NOT (drive1_ok AND
              drive2_ok AND
              drive3_ok AND
              drive4_ok)

      THEN JUMP loop
```

Process-oriented functions

```
   MOVE LINEAR TO beg_pos

   driveon_03=0                      ;switch off 3rd axis

   WAIT 5

   MOVE LINEAR TO end_pos            ;Travel movement without involvement of 3rd
                                     ;axis. It can at the same time be travelled
                                     ;with foreign set value.

   driveon_03=1                      ;switch 3rd axis on again

   loop2:

      WAIT 0.5                       ;wait until 3rd axis is switched on again

      IF NOT drive3_ok

      THEN JUMP loop2

   MOVE LINEAR UNTIL drive3_ok=1     ;Dummy MOVE TO to force read POS
        TO end_pos                   ;since 3rd axis is possibly moved with
                                     ;foreign set value

   MOVE TO beg_pos

   auto_don=0

   HALT

PROGRAM_END
```

☞ **No coordinate system change may take place between the last travel block and within the MOVE UNTIL block, unless the option flag 'read POS at coordinate system change' (P306) is set.**

Example 2

```
;;CONTROL=rho4                     ;compiler instructions

;;KINEMATICS:(1=sr6)

;;SR6.JC_NAMES=a_1,a_2,a_3,a_4

;;SR6.WC_NAMES=a_x,a_y,a_z,a_a

PROGRAM a1234_fb

OUTPUT:

17=auto_don

INPUT:

25=drive1_ok,
```

Process-oriented functions

```
26=drive2_ok,

27=drive3_ok,

28=drive4_ok

BEGIN

   auto_don=0                                    ;switching-over to automatic
                                                 ;DRIVE ON control via
                                                 ;user outputs

   loop:

      WAIT 0.5                                   ;wait until all drives are
                                                 ;switched on

      IF NOT (drive1_ok AND
              drive2_ok AND
              drive3_ok AND
              drive4_ok)

      THEN JUMP loop

   MOVE_REL TO drive1_ok=1 @(10,10,10,10)        ;read actual position

   MOVE TO @(-50,-50,-50,-50)                    ;move to starting position

   WAIT UNTIL drive1_ok=1                        ;travel axes if associated
                                                 ;DRIVE ON signals are set

   MOVE TO drive1_ok=1 TO @(-50,-50,-50,-50)

   MOVE TO @(50,-50,-50,-50)

   WAIT UNTIL drive2_ok=1

   MOVE TO drive2_ok=1 TO @(50,-50,-50,-50)

   MOVE TO @(50,50,-50,-50)

   WAIT UNTIL drive3_ok=1

   MOVE to drive3_ok=1 TO @(50,50,-50,-50)

   MOVE TO @(50,50,50,-50)

   WAIT UNTIL drive4_ok=1

   MOVE TO drive4_ok=1 TO @(50,50,50,-50)

   MOVE TO @(50,50,50,50)

   STOP

PROGRAM_END
```

Process-oriented functions

PCL program for examples 1 and 2 (extract) DRIVE ON, axis-wise

```
AN  -USER_17_RCO                ;inquiry user output 17

JPC MANU_DO                     ;DRIVE ON signals
                                ;influenced by BAPS program
                                ;via user outputs

A   -USER_1_RCO

=   -DRIVE_1_RCI

A   -USER_2_RCO

=   -DRIVE_2_RCI

A   -USER_3_RCO

=   -DRIVE_3_RCI

A   -USER_4_RCO

=   -DRIVE_4_RCI

JPC  DO_END

MANU_DO:                          ;DRIVE ON logic via switch

A   -USER_9_DI

=   -DRIVE_1_RCI

A   -USER_10_DI

=   -DRIVE_2_RCI

A   -USER_11_DI

=   -DRIVE_3_RCI

A   -USER_12_DI

=   -DRIVE_4_RCI

DO_END:                           ;copy of DRIVE ON signals to user inputs
```

Process-oriented functions

```
A    -DRIVE_1_RCI                    ;for inquiry in BAPS program

=    -USER_25_RCI

A    -DRIVE_2_RCI

=    -USER_26_RCI

A    -DRIVE_3_RCI

=    -USER_27_RCI

A    -DRIVE_4_RCI

=    -USER_28_RCI
```

25.1.3 Monitoring

Whenever an axis has to be moved for which no DRIVE ON signal has been set, the running program is aborted with the runtime message 'DRIVE ON not avail.'.

The axis movement is on the JC level checked after the coordinate transformation. Despite switched off-axes, it is thus possible to travel in world coordinates as long as none of the switched off-axes is involved in the movement.

Example: SR6

The 3rd axis is switched off via DRIVE ON. Axes 1 and 2 can nevertheless be travelled in WC, see example 1.

If DRIVE ON is switched off while a program is active, after the switching-on again at first a MOVE TO block for reading the POS has to be carried out. If this condition is not met, the program will in the next travel block be aborted with the runtime message 'DRIVE ON not avail.'.

DRIVE ON is only allowed to be switched on again when all axes of the respective kinematic are stopped. If this is not the case the program will also be aborted with the runtime message 'DRIVE ON not allowed'.

To avoid a runtime message by an inadmissible switching on or off of individual axes, the DRIVE ON signal change in the PCL program should be interlocked accordingly, see example 3.

Example 3 (PCL program)

Disabling of DRIVE ON signal changes with running axes

```
A    -INPOS_1_RCO

A    -INPOS_2_RCO

A    -INPOS_3_RCO
```

Process-oriented functions

```
A   -INPOS_4_RCO

JPC INPOS_AA                          ;One or several axes move
                                      ;-→ DRIVE ON must not be switched on.

A   -INPOS_1_RCO

AN  -USER_9_DI

R   -DRIVE_1_RCI

A   -INPOS_2_RCO

AN  -USER_10_DI

R   -DRIVE_2_RCI

A   -INPOS_3_RCO

AN  -USER_11_DI

R   –DRIVE_3_RCI

A   -INPOS_4_RCO

AN  -USER_12_DI

R   -DRIVE_4_RCI

JPC  DO_END

INPOS_AA:                             ;All axes are in position
                                      ;DRIVE-ON signals can be set
                                      ;accordingly to the switch position.

A   -USER_9_DI

=   -DRIVE_1_RCI

A   -USER_10_DI

=   -DRIVE_2_RCI

A   -USER_11_DI

=   -DRIVE_3_RCI

A   -USER_12_DI

=   -DRIVE_4_RCI

DO_END:
```

Process-oriented functions

25.2 Rapid measuring with probe input and ’fast inputs’

To perform measurements on workpieces, pallets or tools with high accuracy, input signals are required to which the control unit can react with the possibly smallest delay time. With the extension stage of the rho4 described below, it is possible to meet this requirement via probe input or ’fast inputs’.

Programming

A movement cannot be interrupted through a binary input when the BAPS program contains the following command:

```
MOVE ....
  UNTIL Variable Rel_Operator Output ERROR Instruction
TO ....
```

Variable	Channel number of a binary input
Rel_Operator	{ = || <> }
Expression	{ 0 || 1 }

The fast inputs are addressed via the following channel numbers:

6 1 y

y	consecutive number of the fast input
0	probe input
1..8	Fast inputs

The MOVE UNTIL command is executed as for normal binary inputs. The movement is interrupted when the state of the selected input changes to the programmed value (0 or 1).

When the fast input is used, the advantage is that the measuring position can be recorded within the position control cycle. Additional running times through reading of the interface signals (PLC running times) do not occur.

When the probe input is used, the measuring position is saved almost without time delay.

The movement is interrupted in both cases in the next interpolation cycle (Parameter P5). The measuring position can be now determined through reading of @MPOS, where @MPOS is a standard variable that was specifically defined for the application ’Fast measuring’.

For fast inputs, the interruption condition can already exist before the MOVE UNTIL- command is active. In this case, the movement will not be started. For ’probe input’, the signal change must occur while the corresponding set is active. If there is no change, the movement continues up to the programmed final point.

Process-oriented functions

The position of the last completed measurement remains saved in the variable @MPOS until the completion of the next measuring. If no measurement has been performed yet in the active program, @MPOS is not assigned, i.e. the running time error 'Point not def.' appears when reading the @MPOS.

To avoid that positions may be adopted by mistake from preceding measures or that the program may be interrupted because of a @MPOS not assigned, an error condition must be indicated in the MOVE UNTIL set (see example, page 25–9).

Only the positions of the axes that belong to the programmed kinematic are adopted. When it is accessed to a component of @MPOS that is not assigned, the program is interrupted with the running time error 'Point not def.'.

The measuring position is only available in machine coordinates. It can however be converted via the BAPS command **WC** in space coordinates.

Program example

```
;;CONTROL=RHO4
;;KINEMATICS:(1=ROBI1)
;;ROBI1.JC_NAMES=A1,A2
;;ROBI1.WC_NAMES=K1,K2

PROGRAM PROBE1

JC_POINT: @MEASVAL1,@MEASVAL2
POINT: MEASVAL1,MEASVAL2

INPUT:610=PROBE,                                    ;Probe Input
      611=HS_INP1                                   ;1st Fast Input

BEGIN
;;KINEMATIC=ROBI1

;Measuring value via probe
MOVE LINEAR TO START_POS
MOVE LINEAR UNTIL PROBE=1
     ERROR JUMP MEAS_ERR                            ;JUMP MEAS_ERR
TO END_POS1                                         ;prevents from reading
                                                    ;of @MPOS
@MEASVAL1=@MPOS                                     ;read measuring value
MEASVAL1=WC(@MPOS)                                  ;Conversion into
                                                    ;space coordinates

;Measuring value via fast input
MOVE LINEAR TO START_POS
MOVE LINEAR UNTIL HS_INP1=1
     ERROR JUMP MEAS_ERR
     TO END_POS2
@MEASVAL2=@MPOS
MEASVAL2=WC(@MPOS)
JUMP FINISHED
```

Process-oriented functions

```
MEAS_ERR: WRITE 'probe has not responded'

FINISHED: HALT

PROGRAM_END
```

Hardware prerequisites

rho4.1

For the rho4.1, only the probe (channel No. 610) is available.

One of the following PCI_RHO cards must be inserted:

- PCI_RHO_CAN (1070080377)
- PCI_RHO_SERCOS (1070078393)

The kind of the PCI_RHO card is displayed at the PHG under mode 7.3.4.

rho4.0

For rho4.0, only the fast inputs (channels No. 611 to 618) are available.

Machine parameter

`P11 PROBE-INPUT`	(only rho4.1)
`SB1 No.of Probe In:0/1`	(0 = no probe /1 = probe existing)

If a probe input is applied under P11, it will be checked during the control start-up if one of the PCI_RHO cards indicated under the hardware prerequisites is inserted. If this is not the case, the boot will be interrupted with the error message 'Incorr. highspeed-io' (code 18944).

If the probe is programmed in the user program although it was not applied via machine parameter, the error message 'Inadm. input' appears in the running time.

Realizable accuracy

Probe input:

- Reaction time: 10 us
- Accuracy [mm]: v [mm/s] /100000 [1/s] + mech. switch path of the button

Fast inputs:

- Reaction time: 0 to position control cycle (Parameter P5)
- Accuracy [mm]: (v [mm/s] / 1000) * LRTAKT [ms] + mech. switch path of the button

v	Speed in direction of the probe
LRTAKT	Position control cycle (Parameter P5)

Process-oriented functions

25.3 Coded text output

With the coded text output, the machine manufacturer has a tool by means of which he can display to the operator important instructions or currently occurred errors and conditions. The PHG2000, lines 2 and 3, serves as means of display.

Max. 256 different texts can be displayed. The selection of text to be put out is made via the RC-internal interface (RC inputs 280 to 287, rho4 interface address O35.0 to O35.7).

The transfer to the RC is made via 8 bits decimal-coded and a strobe signal (RC input 260, rho4 interface address O32.4). The selected text is put out after the take-over of the text number on the output channel PHG.

After completed output, the acknowledgement signal 'text output ok' at the RC-internal interface (RC output 156, rho4 interface address I19.4) is set.

If the output channel is assigned or not ready, the error strobe 'text output error' (RC output 157, rho4 interface address I19.5) is set.

At the PHG output, the coded text output overwrites the lines 2 and 3 of the displays. To avoid this, the output of RC messages on lines 2 and 3 can be disabled. This is done by setting 'disable outputs of system messages to PHG' (RC input 227, rho4 interface address O28.3).

All texts are contained in the file Text.dat and can be created or edited by the user offline (e.g. by the ROPS–Editor).

Example

01 = 'Text with a length of forty characters !'

coded text output at the PHG :

Text with a length of forty characters !

The max. text length for the PHG output is 40 characters. Longer texts are cut off.

Process-oriented functions

File Text.dat

The code numbers of the texts are hexadecimal. They must always be entered with two digits. The characters A to F must be capital letters.

Space characters between number and '=', resp. between '=' and text are admissible, they are ignored. The text to be put out is to be inserted in inverted commas. Controller characters in the text are not admitted. Comments are characterized by a semicolon ";". The admissible text length is 80 characters.

Structure of the file Text.dat

Code number		Assigned output text	Comment
00	=	'first text'	
01	=	'text text'	
02	=	'texttexttext'	
.			
3F	=	'text'	;comment
FF	=	'last text'	

The name of the file is for English surface Text.dat, with German surface Texte.dat.

Interface assignment of coded text output

With set RC input 260 (rho4 interface address O32.4), the HEX number at the RC inputs 280 to 287 (rho4 interface address O35.0 to O35.7) is taken over by the RC and the text output is activated. After completed output, the RC output 156 (rho4 interface address I19.4), in the case of error RC output157 (rho4 interface address I19.5) is set.

RC outputs

Current no.	PLC symbol name	rho4 interface address	Signal description
156	TXOUT_OK_RCO	I 19.4	**Acknowledgement, coded text output:** Strobe signal, is set when the selected text could be put out without errors. The time can be set via machine parameter P9, the default is 110 ms.
157	TXOUT_ER_RCO	I 19.5	**Errors with coded text output:** Strobe signal, is set when the selected text could not be put out (output channel occupied, no text available for the selected number). The time can be set via machine parameter P9, the default is 110 ms.

Process-oriented functions

RC inputs

Current no.	PLC symbol name	rho4 interface address	Signal description
260	STB_TXTO_RCI	O 32.4	**Strobe, coded text output:** Activates the coded text output when changing from ’0’ to ’1’. The signal is to be sent to the RC as strobe signal with a minimum time of 100 ms. In the case of an error, for example ’text for the selected number not available’, the RC output current no. 157, ’errors with coded text output’ is set as strobe signal (strobe time can be adjusted via machine parameter P9). When the RC output current no. 156 ’Acknowledgement coded text output’, has been selected correctly, it is set as a strobe signal.
227	DIS_PHGM_RCI	O 28.3	**Disable outputs of system messages to PHG:** This permits to avoid that a coded text output or user texts (WRITE PHG...) is overwritten by the RC.
280 to 287	TXTOUT_0_RCI to TXTOUT_7_RCI	O 35.0 to O 35.7	**Coded text output, bits 1 to 8 (bit significance 1 to 128):** The signals are taken over in the RC with the strobe signal (RC input current no. 260) as hexadecimal coded text number (00’H to FF’H). The associated text is stored in the reference file Text.dat. 256 different texts can be selected.

Restrictions and messages

Controller characters in the text are not admitted.

If the file Text.dat is missing or if a text cannot be put out since the output channel is occupied, the text output error strobe is set.

☞ **As output channel only PHG is admitted at the moment.**

Process-oriented functions

25.4 Coded state output

With the function 'coded state output', all status messages and warnings recognized by the operating system of the rho4, are put out binary-coded at the digital interface of the rho4.

Hereafter, the structure of the code numbers, the interface assignment and the way how code numbers are put out at the interface, are described.

The exact meaning of the code numbers, the status messages and warnings that are displayed at the PHG, a possible error cause as well as instructions for the elimination of errors are to be found in the manual status messages and warnings.

The occurring messages are sorted by numbers.

a = 1 to 24, axis index

k = 1 to 16, kinematic index

p = 1 to 16, PPO index

x = 1 to 40, CAN input block index

y = 1 to 40, CAN output block index

25.4.1 Coded state output of runtime messages

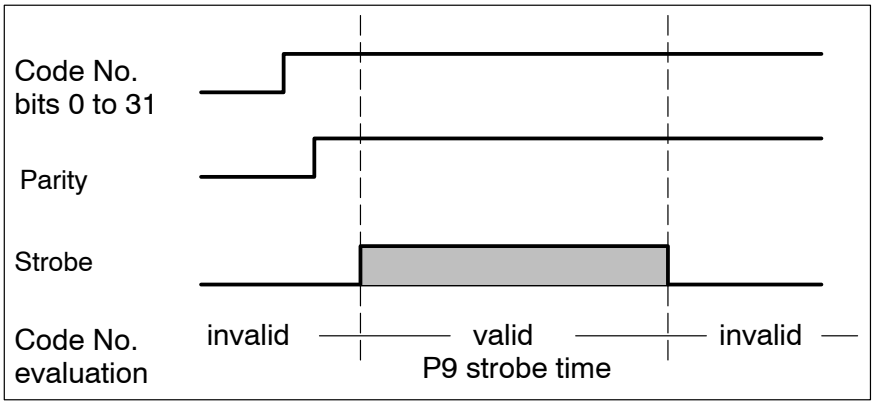
By runtime messages we understand messages which are cleared after having eliminated the cause and set the basic setting, e. g. point variable is undefined.

The operating system of the rho4 stores up to 64 runtime messages. These messages are put out, as described hereafter, via the bit 0 to 31, current number 168 to 199, symbol name STATUS_0_RCO to STATUS_31_RCO.

Output principle

If a runtime message occurs, it is put out coded at the interface via code number bits 0 to 31 with strobe signal, i. e., the code message can only be evaluated when the strobe signal is given, current no. 166, STB_COER_RCO.

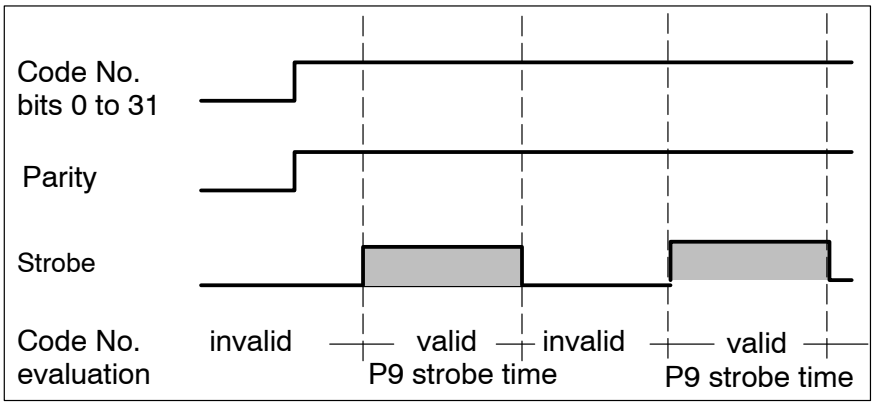
Process-oriented functions



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

Output of several code numbers

If several messages occur, they will be displayed one after the other. The strobe signal becomes zero after the expiration of the set strobe time. It remains again 0 for this strobe time. Then the next code number is put out. The strobe signal then goes again to 1, see diagram.



This sequence is continued until the last code number has been put out.

Parity

The parity bit, number 167, PTY_COER_RCO, is formed for even parity. It is formed for bit 0 to 31 of the code number and is only valid with given strobe signal.

Process-oriented functions

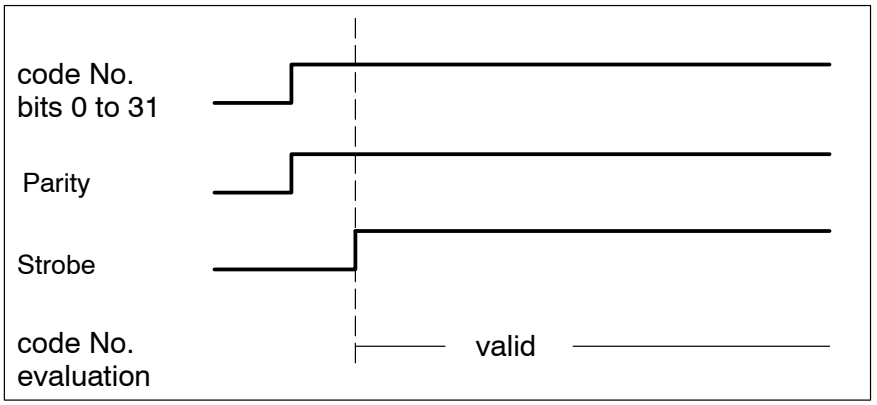
Behaviour in case of consecutive messages

When a message is already available (e. g. Emergency stop) and one or several further messages follow (e. g. interpolator stop warning axis 1, servo error axis 1), then all code numbers which are still available will in case of new occurring messages be put out again. By means of the basic setting at the interface or the basic setting at the PHG2000 with mode 11.1, the messages can be deleted again after having remedied the cause.

25.4.2 Coded state output of system messages

System messages are also put out coded at the interface of the rho4. System messages can, when the cause has been remedied, only be cleared with the RC restart.

Contrary to the coded state output of runtime messages, in case of which the strobe signal is only given with system strobe time, the strobe signal is available after having put out the code number 'static', i. e. it is only deleted after having pressed a key at the PHG.



The parity signal is formed for even parity via all 32 state code bits. It is only valid with given strobe signal.

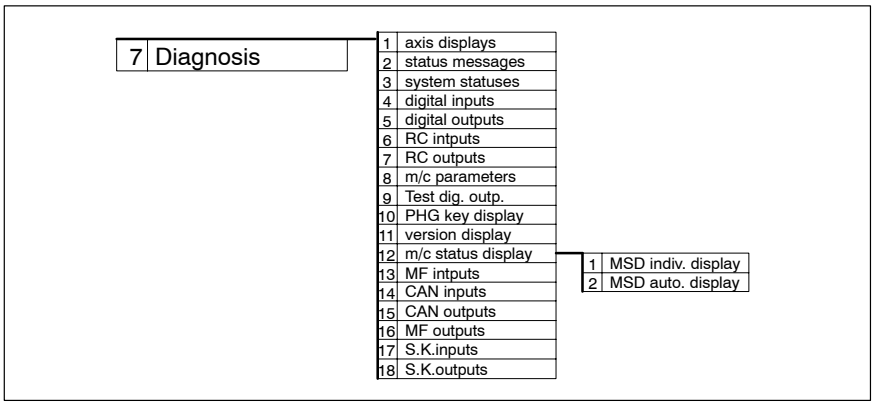
Process-oriented functions

25.5 Machine status display

With the machine status display (MSD), the user can give instructions to the operator via conditions of the system. To keep the expenditure in the PLC program as little as possible and most of the statuses can normally also be used as user inputs in BAPS programs, for each of the first 99 user inputs, an associated text can be stored in the file MSD.dat.

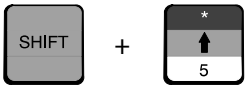
To be able to make the operator aware of a new status, the 'MSD display activated' (RC output 163, rho4 interface address I20.3) is available. As long as this signal is set to 1, the machine status is displayed in line 4 of the PHG2000, under mode 7.12.

Via the PHG diagnosis mode, submode MSD (mode 7.12), all active statuses, i. e. the associated RC input is 1, can be displayed one after the other.

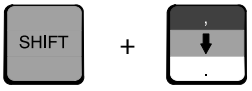


Two display possibilities are available.

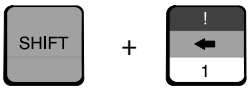
1 Manual with scrolling via



or



2 Automatic and cyclical every second until the sequence is interrupted with



In both possibilities, two statuses are displayed simultaneously in line 2 and line 3.

Process-oriented functions

Example

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

Machine status display at the PHG2000

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

Texts exceeding the line end of a PHG line are cut off.

MSD.dat

The MSD.dat is structured in a similar form as the ASCII file EXPROG.dat (external program selection).

The numbers of the statuses and their associated texts are BCD-coded. They must always be entered with two digits. Space characters between number and '=', resp. between '=' and text are admissible, they are ignored. The MSD text must be limited by inverted commas. Controller characters in the text are not admitted. Comments are characterized by a semicolon ";". The admissible text length is 80 characters.

Structure of the file MSD.dat

Code value		Assigned output text	Comment
01	=	'first text'	
02	=	'texttexttext'	;Comment line for additional ;information
20	=	'text'	;Comment until line end as far as ;desired
32	=	'text'	
99	=	'last text'	

With a German surface, the file name is MZA.dat, with an English surface MSD.dat (machine status display).

Process-oriented functions

MSD interface assignment

In addition to their function as user inputs (1 to 199), the RC inputs 1072 to 1270 (rho4 interface addresses O134.0 to O158.6) can be used for the machine status display.

When selecting the diagnosis function 'machine statuses' (mode 7.12), submode 1 or 2, the RC output 163 (rho4 interface address I20.3) is set and reset again when leaving submode 12. The user can thus recognize in the PLC program, whether the MSD display has been activated by the operator.

RC outputs

Current no.	PLC symbol name	rho4- interface address	Signal description
163	MSD_ACTI_RCO	I 20.3	**MSD display has been activated:** Is set by the RC to 1 when under PHG DIAGNOSIS (mode 7) in the submode'machine statuses' (mode 12) one of the two display types has been activated. Is reset to 0 when mode 12 is left again.

RC inputs

Current no.	PLC symbol name	rho4- interface address	Signal description
1072 to 1270	USER_1_RCI to USER_199_RCI	O 134.0 to O 158.6	**User inputs 1 to 199:** These binary interface signals can be inquired from BAPS. The inputs 1 to 99 are evaluated in parallel as machine statuses.

Restrictions and messages

Controller characters in the text are not admitted.

With missing MSD file, a corresponding message at the PHG is put out at the time of display.

Process-oriented functions

25.6 Belt-synchronous

The synchronization instructions ensure that the controlled machine takes the correct position with regard to position and orientation. The belt can move forward or backward without changing its speed or it can stop.

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

With this extension stage, four different belt synchronization types are available in the rho4.

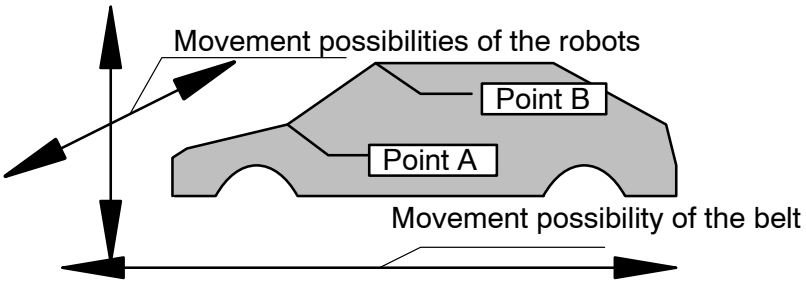
- Belt kind 1:
  Belt synchronization with belt-parallel travel of the kinematic. Only LINEAR and CIRCULAR movements are permissible here. Speeds(V) and accelerations (A) in the program are relative to the parts moving with the belt (see BAPS3 Programming instructions manual, chapter Belt-synchronous)
- Belt kind 2:
  Belt synchronization without belt-parallel travel of the kinematic. The programmed speed in the belt-synchronous section is not important. It results from the current belt speed. Speed changes are executed with the programmed acceleration (see subsection Belt synchronization without belt-parallel travel)
- Belt kind 3:
  “Cam disk interpolation”. Cam disk interpolation is a variant of belt synchronization without belt-parallel travel (belt kind 2). Speed changes are, however, not executed with the programmed accelerations. They result from a function curve and the current belt speed (see subsection Belt synchronization kind 3 (cam disk interpolation))
- Belt kind 4:
  Belt kind 4 is an expansion to the belt synchronization with belt-parallel travel (belt kind 1).
  Belt-synchronous movements are also permitted with belt kind 4. The effect is that the belt is softly coupled in the first belt-synchronous travel block and softly uncoupled in the last belt-synchronous travel block. Travel blocks can be passed during transition into the SYNCHRON section. Travel blocks can also be passed when leaving the SYNCHRON section.
  (see subsection Belt synchronization kind 4)

Process-oriented functions

25.6.1 Belt synchronization without belt-parallel travelling possibility

Hereafter the function, the parameters and the programming of the belt synchronization kind 2 are described. The belt synchronization kind 2 can be selected via special function 21.

In this type of synchronization the kinematic has no belt-parallel travelling possibility.



In the synchronous part, the robot is moved in such a way that the belt and the robot reach the programmed end point together.

The programmed speed can in this case not be taken into account. It depends on the belt speed and the angle of slope which results from the position of the supporting points and their belt components.

The programmed acceleration is in this type of synchronization taken into account. The consideration of the programmed acceleration treats on the one hand the robot with care, but leads on the other hand also to speed-dependent deviations from the programmed path.

Operating conditions for the belt synchronization kind 2

For the SYNCHRON program part of the belt synchronization kind 2 the following special conditions apply:

- When the kinematic is travelled belt-synchronously, no other process can move this process in the meantime. It has to wait until the synchronous part is finished.
- If runtime errors occur, the READY contact is opened so that also the belt itself can be stopped. In this case, the runtime message ’Emergency stop input’, code 16640 is generated additionally.
- If the first belt-synchronous block gets only active when the belt is already further than the belt value (incl. the indicated tolerance) indicated in the last travel block before ’belt synchron’, the message ’BS-tol.with 1.Synch.’, codes 23040 to 23055 is put out. The reasons for this can be:
  - program started too late
  - travel movements take longer since the axes are not in position
  - travel section through end point of the last program longer
  - belt travels too fast

Process-oriented functions

- If a belt-synchronous block gets only active when the belt is already further than the belt value (incl. tolerance) indicated in the previous belt-synchronous travel block, the message 'BS-tol.with nth MOVE', codes 22272 to 22287 is put out. The reasons for this can be:
  - input inquiries delaying the block preparation
  - many computations, loops
  - processes of higher priority
  - access to occupied resources, e. g. kinematic, units
  - insufficient time for the block preparation (machine parameter P5)

☞ **If a belt-synchronous block gets already active when the belt is still before the belt value (less tolerance) indicated in the previous belt-synchronous travel block, no message is put out. It is waited until the belt reaches the range of tolerance!**

- If the end of a belt-synchronous block is not reached until the belt is further than the belt value (incl. the indicated tolerance) indicated in this travel block, the message 'BS-tol. at block end', codes 23296 to 23311 is put out. The reasons for this can be:
  - geometry too steep
  - insufficient programmed acceleration or deceleration values
  - too long jerk limitation times at $\sin^2$ slope
  - too short point sequence
  - during the control at the beginning of the block, the tolerance time was still active, e. g. by retriggering

- The messages 'BS-tol.with 1.Synch.', 'BS-tol.with nth MOVE' and 'BS-tol. at block end' are only put out if the tolerance time indicated via SPC_FCT:15(belt_param) is not yet expired.
- If the kinematic is prevented from moving by 'FEED ALLOW', the program will abort with the kinematic-dependent runtime message 'BS: no feed allow', codes 16256 to 16271. Via the option bit 'belt synchron without belt-parallel axis' the effect can be determined by means of 'FEED ALLOW'. If value 1 is entered for the option bit ,the 'FEED ALLOW' signal is effective. If a 3 is entered, the 'FEED ALLOW' signal has no effect on the belt synchronization.

☞ **The monitoring is only effective when the kinematic is actually moved!**

- The interpolation type PTP is not admissible. The program aborts at PTP with the runtime message 'PTP not allowed here', code 138496.
- In case of the first belt-synchronous travel movement, the movement has to wait until the belt has reached the belt value programmed at the last travel block before the belt-synchronous part.

- If the belt moves before the first belt-synchronous travel movement against the programmed belt direction, the program is aborted with the kinematic-dependent runtime message ’belt dir. change prog’, code 147072.
- The size of the tolerance range can be changed by calling SPC_FCT: 21=belt_kind (VALUE INTEGER: no. VALUE INTEGER: kind), see chapter 9.
- The belt difference of two successive travel blocks must not be 0 (e. g. MOVE_REL(0,0)), otherwise the program aborts with the runtime message ’belt distance is zero’**,** code 147328.
- The belt difference must have the same signs for each travel block within the belt-synchronous program section, otherwise the message ’belt dir. change prog’, code 147072 is output.
- A MOVE_REL block as first travel block in the SYNCHRON part is permitted here.
- The commands SYNCHRON and SYNCHRON_END do not stop the block preparation here.
- The kinematic-dependent runtime messages and ’Emergency stop input’ are displayed in the process display as ’other messages’.

☞ **The measuring system conversion factor of the belt should be selected as high as possible when using belt kind 2, since the measuring value is here not only used for the position acquisition, but also to acquire the belt speed. With this speed, the probable arrival of the belt at the next supporting point is determined. A measuring system conversion factor that is too low leads to frequent speed corrections and thus to a higher computing load!**

25.6.2 Belt synchronization kind 3 (cam disk interpolation)

Operating conditions for the belt synchronization kind 3

For the synchronous program part of the belt synchronization kind 3, the following special conditions apply:

- If the kinematic is moved belt-synchronously, no other process can move this kinematic. It has to wait until the synchronous part is finished.
- If runtime messages occur, the READY contact is opened so that in case of need also the belt itself can be stopped. In this case the runtime message ’Emergency stop input’, code 16640 is generated additionally.

Process-oriented functions

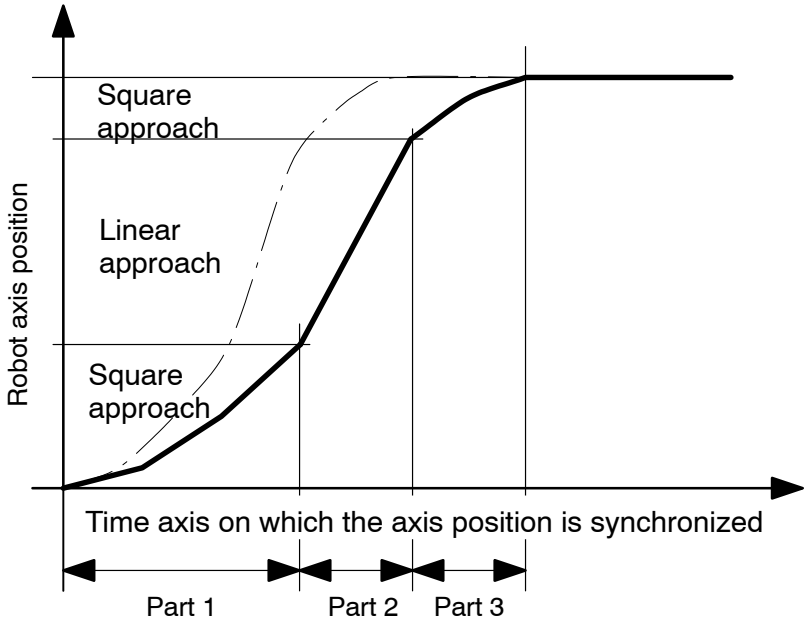
- If the kinematic is prevented from moving by 'FEED ALLOW', the program will abort with the kinematic-dependent runtime message 'BS: no feed allow', codes 16256 to 16271. Via the option bit 'belt synchron without belt-parallel axis' the effect can be determined by means of 'FEED ALLOW'. If value 1 is entered for the option bit ,the 'FEED ALLOW' signal is effective. If 3 is entered, the 'FEED ALLOW' signal has no effect on the belt synchronization.

☞ **The monitoring is only effective when the kinematic is actually moved!**

- The interpolation types PTP and CIRCULAR are inadmissible. The program aborts in case of PTP with the runtime message 'PTP not allowed here', code 138496 and in case of CIRCULAR with 'beltsync. n. possible.', code 141952.
- In case of the first belt-synchronous travel movement, the movement has to wait until the belt has reached the belt value programmed at the last travel block before the belt-synchronous part.
- If the belt moves before the first belt-synchronous travel movement against the programmed belt direction, the program is aborted with the kinematic-dependent runtime message 'belt dir. change prog', code 147072.
- If the belt moves during a belt-synchronous travel movement against the programmed belt direction, the kinematic also travels within this partial block against the programmed moving direction. However, the program does not run backward.
- At the travelling block transitions, the deviation from the programmed belt position is checked. If this deviation exceeds the path made by the belt within a specific number, 5 as standard, of interpolation clocks, the program aborts with a kinematic-dependent runtime message. The size of the tolerance range can be changed by calling 'SPC_FCT:21= belt_kind(VALUE INTEGER: no. VALUE INTEGER: kind)'.
- The belt difference between two successive travel blocks must not be 0, e. g. MOVE_REL (0, 0), otherwise the program aborts with the runtime message 'beltdistance is zero', code 147328.
- The belt difference must have the same signs for each travel block within the belt-synchronous program section, otherwise the message 'belt dir. change prog', code 147072 is put out.
- MOVE_REL as first travel block in the synchronous part is permitted here.
- The commands SYNCHRON and SYNCHRON_END do not stop the block preparation here.
- The kinematic-dependent runtime messages and 'Emergency stop input' are displayed in the process display as 'other messages'.

Process-oriented functions

Description of the parameterized course of the curve



The transitions between the part sections are smooth. The slope of the second part section and thus also of the transitions is computed from these marginal conditions. The size of the first and third part section can be programmed in wide ranges.

The size of the second part section results from the programmed overall length.

Programming of the part sections

For the determination of the course of the curve are required

- the overall belt path
- the belt path of the first part section
- the belt path of the third part section
- the path sections for the robot coordinates

The overall belt path results from the difference of the programmed belt coordinates of the end point and the belt coordinate of point approached last.

For the programming of the belt path of the first part section, the meaning of the acceleration default not required here is redefined. The value from A, kinematic.AFACTOR and global AFACTOR, thus corresponds to this belt path.

For the programming of the belt path of the third part section, the meaning of the brake default not required here is redefined. The value from D, kinematic.DFACTOR and global DFACTOR, thus corresponds to this belt path.

All coordinates are, according to the path differences programmed for them, moved during the travel movement along the curve default, i. e. possibly also in opposite direction. For the programming the MOVE_REL command, e. g. MOVE_REL(100,200,0,0,200), can be used.

Special cases

If a part section is smaller than one percent of the overall length, it is set to this minimum value. If the total of the first and third part sections exceeds the overall length, the second part section is set to the minimum value and the rest of the overall part split proportionally to the first and third part sections.

Difference to normal travel blocks

In case of travel blocks with ramp slope by blocks, the robot also generates a three-part course of the curve. The size of the part sections depends in this case, however, from the programmed speed and acceleration / deceleration.

In case of belt kind 3, these sizes are, however, directly coupled to the belt position. They are thus independent from the programmed speed.

In case of high speeds or short belt paths, it is, of course, here necessary, too, to take the scanning effects into account.

Belt offset time

For the belt synchronization kinds 1 and 4, position errors on the running belt due to lags of the concerned axes with belt-parallel movement possibility can be compensated via the machine parameter P507 'Belt time offset'.

For the belt synchronization kinds 2 and 3 (without belt-parallel movement possibility), position errors to the running belt can arise through reaction times between position registration of the belt and the positioning of the axes.

For the axes coupled via a function curve in the belt synchronization kind 3 'Cam disk interpolation', it is possible as an option to reduce the position error due to reaction times from version VO03F.

Activation

The machine P507 'Belt time offset' is activated globally during the cam disk interpolation via the bit 3, value 8, of the option byte for the cam disk interpolation.

The value of the machine parameter P507 'Belt offset time' in SYNCHRON sections with the belt synchronization kind3 becomes active for all belts. The machine parameter P507 'Belt offset time' can be set separately for every belt. With the value 0, it is deactivated.

Process-oriented functions

Via the bit 2, value 4 of the option byte for the cam disk interpolation, the rho4 can be caused to use as a belt speed a sliding arithmetical mean value consisting of up to 10 measuring values.

The extension stage cam disk interpolation is activated via the bit 0, value 1, of the option byte for the cam disk interpolation.

It follows the hexadecimal combinations 0, 1, 9, 0D. The adjusted value can be read at the machine parameter P28.

Example:
Extension stage active, belt offset time effective, determination of the mean value of the belt speed

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	1	1	0	1

The hexadecimal value 0D for the option byte for the cam disk interpolation must then be set.

Mode of operation of the belt offset time

The machine parameter P507 'Belt time offset' is a time in milliseconds. The adjusted value is rounded to the multiple of the machine parameter P5 'Clock-Start time'.

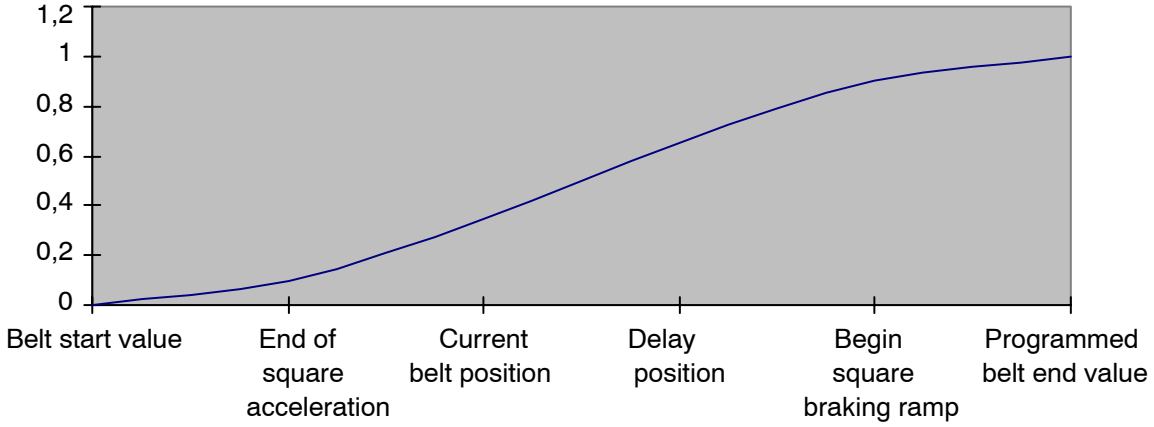
The belt speed is determinated through:

- the difference of two belt positions
- the sliding arithmetical mean value composed of up to 10 measuring values

The screening of the measuring values in increments per scanning interval is to be taken into account. It follows:

P401 Measuring system evaluation [Inkr./mm] * Belt speed [mm/s] * P5 Clockstart time [ms] / 1000

For the cam disk interpolation, the position of the axes is determined via a parametrized function run from the current belt position.



Process-oriented functions

Through the belt offset time, the position of the axes is not determined with the current belt position but with a preceding belt position. The concerned path is given by:

Offset time * Belt speed

Variations or griddings of the belt speed lead to position variations weighed with the belt offset time.

☞ **If the machine parameter P5 'Clock start time' is modified, it must be checked if the machine parameter P507 'Belt offset time' must also be changed.**

Functioning of the belt speed determination

The mean value determination of the belt speed acts

- for the belt synchronization kind 2 permanently
- for the belt synchronization kind 3 as an option

For the belt synchronization kind 3, the parameter 'Belt_Run' (Belt runs) of the special function SPC_FCT : 15 = Belt_Param (...) has no effect on the function of the mean value determination. The parameter 'Belt_Stop' (belt stops) the special function SPC_FCT : 15 = Belt_Param (...) keeps on having effect.

When selecting the belt synchronization kind 2 or 3 via the special function SPC_FCT : 21 = Belt_Kind (...), the mean value determination begins again.

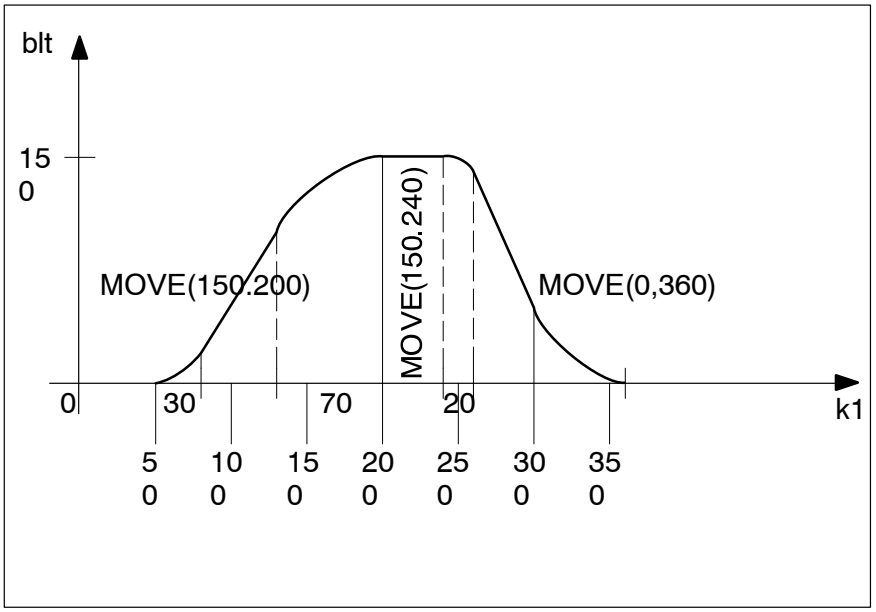
Example program

```
 1  ;;KINEMATICS:1=rob1
 2  ;;rob1.WC_NAMES=k1,blt
 3  ;;rob1.JC_NAMES=k1,blt
 4  PROGRAM belt_kind3
 5  SPC_FCT:21=belt_kind(VALUE INTEGER: belt_no
                         VALUE INTEGER: kind_belt)
 6  rob1.BELT:501=round_axis
 7  BEGIN
 8     belt_kind(1,3)
 9     A=10
10     loop:
11     SYNC round_axis>=360.0
```

Process-oriented functions

```
12      MOVE LINEAR VIA (0,50)
13      SYNCHRON rob1 round_axis
14         AFACTOR=0.3                  ;A*AFACTOR=30
15         DFACTOR=0.7                  ;A*DFACTOR=70
16         MOVE(150.200)
17         MOVE(150,240)                ;corresponds to waiting until rd axis>=240
18         A=100
19         AFACTOR=0.2                  ;A*AFACTOR=20
20         DFACTOR=0.6                  ;A*DFACTOR=60
21         MOVE(0,360)
22      SYNCHRON_END
23      JUMP loop
24   PROGRAM_END
```

Explanations to the example



The BAPS keywords are in capital letters. Other freely selectable names in small letters.

Line 5 contains the declaration of special function 21 to switch between the belt synchronization kinds. It is called in line 8.

The reset of the belt counter is made via the SYNC command in line 11.

In line 12, rob1 is moved to position 0. The belt value 50 has no significance yet for this movement. It is nevertheless important for the first belt-synchronous travel block in line 16. There rob1 is waiting until the belt value has reached the value 50. If the belt moves away from the begin value, in this case into the negative direction, the program aborts with the message 'belt dir. change prog', code 147072.

If the belt value 50 has been exceeded, the position of rob1 directly depends on the belt value. In each scanning step, the position is determined via the course of the belt value curve.

In line 14 (19) the belt path of the first part section is fixed to the value 30 (20). In line 15 (20) the belt path of the third part section is fixed to the value 70 (60). The values in brackets refer to the MOVE instruction in line 21.

From this, the following sequence results for the MOVE instruction in line 16:

- from the belt value 50 to the belt value 50+30, the position of rob1 is square-dependent from the belt value.
- from 80 to 200–70 it is linear-dependent.
- from 200–70 to 200 it is again square-dependent.
- at the end rob1 is at (150,200).

Restrictions

- In the belt-synchronous program section, the meanings of the acceleration and deceleration values change.
- If the global or kinematic-specific values for AFACTOR or DFACTOR are reduced here, the acceleration resp. deceleration will be increased.
- The premature initiation of the block preparation has not been taken into account. For this reason, it is possible that program branchings lead because of input signals to program aborts.
- The value for the programmed acceleration A is internally limited to $2000/(P5)^2$, e.g. 20 for P5 = 10 ms. If smaller values for the path are to be programmed, this has to be done via correspondingly small values for AFACTOR and DFACTOR.

Process-oriented functions

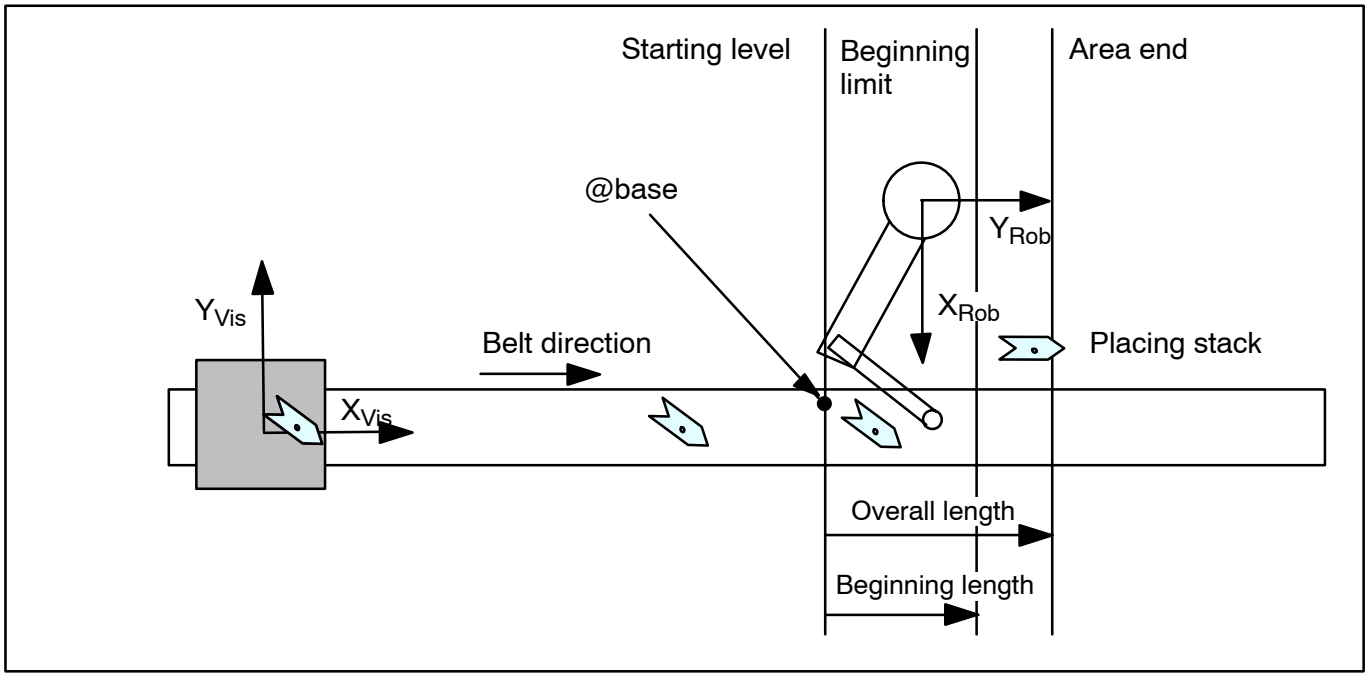
25.6.3 Belt-synchronous, belt kind 4

The belt synchronization kind 4 is an expansion of belt kind 1 for kinematics with a movement possibility in the belt direction.

It enables

- soft coupling and uncoupling of the belt synchronization
- passing when coupling and uncoupling the belt synchronization
- monitoring of the belt working area limits

The discription relates to the more user-friendly user interface that is available from version VO05 onwards.



The image shows the same part at four different locations in the process:

- Being recorded by the vision system
- On the way to the robot
- In the belt-synchronous working area
- On the placing stack

The belt direction is set with the machine parameter P503 belt coupling factor. In this case the belt is turned +90 degrees from the robot coordinate system, thus coord1=0, coord2=1, coord3=0.

The starting level passes via the machine coordinate point @base and is vertical to the belt direction. The beginning limit and the area end are parallel to the starting level.

Process-oriented functions

At the point where the image is taken in the vision coordinate system, the part is turned –30 degrees at Xvis050, Yvis015. The position would be useless if the belt counter = 4512 was not recorded at the same time as the image.

The position has to be converted into the robot coordinate system for processing. The result is Xrob=285, Yrob=–930 and a turning angle of +60 degrees. In this case, the corresponding belt counter = 4512 also has to be stored.

In the second image of the part, the belt has moved 600 mm further to value 5112. The part is now located in the robot coordinate system at Xrob = 285, Yrob = –330, turning angle = +60 degrees.
Even if the values in the robot coordinate system have changed, the robot coordinate system still describes, along with the belt counter, the same belt-synchronous position.

On the basis of the belt-synchronous position at the time when the image is taken, the belt value where the part will pass the starting level is determined. In this case, the belt value is 5312.

The belt counter is already at 5377 during the first movement in the SYNCHRON section. The part is located at Xrob = 285, Yrob = –65, turning angle = +60 degrees, and thus already in the belt-synchronous working area. It can be gripped and turned and set on the placing stack.

The path of a part is presented here. In reality, the vision system would recognize other parts and their positions with the corresponding belt counter in the meantime.

A minimal possible distance between parts or a maximum possible belt speed results from the processing and movement times. A starting level that is very far ahead enables the robot to approach the part from a long distance. The longer travelling movements to the placing stack that are then necessary, can, however, eat up the apparent extra time.
The cycle can be positively influenced with a well laid-out system of belt, robot and processing positions.

Overview about the function extensions

The belt kind 4 uses the BAPS3 language symbols SYNCHRON and SYNCHRON_END. It must be selected before via the special function 21 Belt_Kind (...).

The following special functions are useful for belt kind 4:

- SPC_FCT: 53 =Belt_Area (...)
  to describe the geometric position of the belt-synchronous working area.

  This SPC_FCT has to be programmed for use with belt kind 4.

  The other SPC_FCT are auxiliary functions.

Process-oriented functions

- SPC_FCT: 54 =Belt_V (...)
  to determine the current belt speed.
- SPC_FCT: 55 =Belt_V_Sim (...)
  to test the belt-synchronous programs with various simulated belt speeds.
- SPC_FCT: 47 =exc_define (...)
  Here: to activate an automatic handling of exceptional situations when the belt area has been exceeded.
- SPC_FCT: 48 =exc_detect (...)
  Here: to detect the exceptional situations activated with the SPC_FCT: 47 exc_define (...).
- SPC_FCT: 52 = Belt_Ptp_Fac (...)
  to read (and set) the factors for the geometric reactions of the axis motions to the belt motion. They depend on the position of the belt in the working area of the robot. The factors are automatically determined when SPC_FCT: 53 = Belt_Area (...) is called up.
- SPC_FCT: 51 = Belt_Calc (...)
  to set and read the process-related belt counter calculation value. Programming the belt counter calculation value is no longer necessary. Calling up the function leads to the runtime error message 'Inadmiss. SPC_FCT' , code 131840, from version 5 onwards.

Safety instructions

For belt-synchronous applications, each program run or motion obstruction, as well as the READY signal of the control, must cause the belt to stop. Otherwise collisions betwen the parts being moved by the belt and the kinematic or parts moved by the kinematic can occur.

In the creation of the AUTOMATIC program, special cases are also to be taken into account such as new start of the control, belt stop, belt running too rapidly, etc.

☞ **It must be ensured that the reset of a belt counter does not occur when a process is running synchronously to this belt.**

Application instructions

The belt kind 4, from version VO05 onwards, should be used for new applications with belt-parallel moving possibility instead of belt kind 1, because for belt-synchronous applications in Pick-and-Place use areas, there is the possibility:

- of using PTP movements in SYNCHRON sections
- to change between SYNCHRON sections and non-synchronous sections harmonously and at the best appropriate time.

For the use of the control function "Spatial passing" (see rho4 manual 'BAPS3 Programming instructions', chapter 'Spatial passing'), it must be considered that the input inquiries prevent a passing.

Process-oriented functions

In the creation of forking for the moving to waiting positions or for program loops the following fact should be taken into account:

- If the motion starting from a fixed position passes a position and leads to a belt-synchronous position, the belt-synchronous position must be already known at the start from the fixed position.

For the change of old applications with the belt kind 1 into the belt kind 4, heed the following:

- The belt kinds 1 and 4 behave much differently at the beginning and the end of the belt-synchronous section. A mere change in the programming of the belt kind leads rapidly to incorrect motions in extreme arm positions.
- In an existing program for the belt kind 1, the movement before SYNCHRON and the movement after SYNCHRON_END will be usually included into the belt-synchronous section of a program for belt kind 4. Additional provisional points can easily cancel the time advantage.
- Through the additional travel blocks at the beginning and at the end of the belt-synchronous section, the duraction of the belt-synchronous section increases. It is to be ensured that the belt counter will not be reset during the belt-synchronous section.
- For the belt kind 4, the final point of the last point of the belt-synchronous section is already a fixed position in the space. The programmed belt value must be 0.0.
- For the belt kind 4, the programmed belt value will be taken into account within the SYNCHRON section in the case of a MOVE_REL. If a MOVE_REL has been porgrammed with a belt value different from zero in the belt synchronization kind 1, then a zero must be programmed for the use of the belt kind 4.

Selection of the belt synchronization kind

After the boot of the control, the belt synchronization kind 1 is active. The kinds of belt synchronization can be selected with the special function 21., see chapter 9 'Belt kind'.

Special function:

```
SPC_FCT : 21=Belt_Kind (VALUE INTEGER: Belt_No
                        VALUE INTEGER: Kind_Belt)
```

Example for calling the belt kind for switching the first belt to belt kind 4 "PTP with belt-synchronous":

Belt_Kind (1,4)

If the belt runs in the negative direction, the value -4 has to be programmed as Kind_belt.

If the synchronization kind of a belt is changed, this setting applies to all processes until they are modified again or the control boots again.

The parameterization of the other belt synchronization kinds is described in the manual control functions chapter 9 belt kind.

Process-oriented functions

A SYNCHRON section begins and ends in a BAPS3 program through the following commands:

- SYNCHRON [kinematic name] Belt variable
- SYNCHRON_END [kinematic name] Belt variable

☞ **It is not necessary to enter the kinematic name, since the belt variable is assigned with the declaration of a kinematic.**

During the SYNCHRON section, the kinematic is synchronized with the belt motion.

With belt kind 1, the synchronization is started stepwise with the command SYNCHRON and ends stepwise with the command SYNCHRON_END. The control function ”Spatial passing” is not effective for the transitions from the non-SYNCHRON section to the SYNCHRON section and vice-versa.

For the belt kind 4, the synchronization is started softly in the first belt-synchronous travel block and ends softly in the last belt-synchronous travel block. The kinematic is entierely synchronized with the belt motion only between the first and the last belt-synchronous travel block. When the belt variable changes, the robot position changes according to the belt direction given in the machine parameter P503, even if no movement is programmed.

The control function ”Spatial passing” can be also used in belt kind 4 for the transitions into the SYNCHRON section and out of the SYNCHRON section. In the passing areas of the transitions, the belt synchronization has partly still/already effects. The programmed ”Spatial passing” is not effective within the SYCHRON section. Deactivating the ”Spatial passing” in the program is recommended.

Last travel block before the SYNCHRON section

When the last travel block before the SYNCHRON section and the first belt-synchronous travel block are passed, the robot begins already in the passing area of the last travel block before the SYNCHRON section to synchronize with the belt.

First belt-synchronous travel block

The first belt-synchronous movement is started if the belt-synchronous destination point is in the working area, which has a fixed position and is defined by SPC_FCT : 53 = Belt_Area (...). If the belt-synchronous destination point is not yet in the belt area, then there is a waiting period until the destination point reaches the belt area (indirect WAIT UNTIL BLT).

Process-oriented functions

If the last travel block before the SYNCHRON section and the first belt-synchronous travel block are passed, the passing area depends on reaching the belt start value. It might then decrease in relation to the programming and adopt the value zero in the extreme case.

☞ **In the first belt-synchronous travel block, the coupling of the robot motion in relation to the belt motion is increased sinequadratically and proportionately to the distance covered from 0% to 100%. In this travel block, the movement is not yet completely belt-synchronous.**

Last belt-synchronous travel block

In the last belt-synchronous travel block, the coupling of the robot motion in relation to the belt motion is decreased proportionately to the distance covered from 100% to 0%. In this travel block, the movement is no longer completely belt-synchronous.

☞ **The destination of the last belt-synchronous travel block is a fixed position in the space. It is characterized in that the belt component contains a 0.0 ID. Only one travel block may have such an identification in the SYNCHRON section.**

First travel block after the SYNCHRON_END

If the last belt-synchronous travel block adn the first travel block are passed after the SYNCHRON_END, the robot still moves partly belt-synchronously in the passing area of the first travel block after the SYNCHRON_END.

Special cases

Only one travel block in the SYNCHRON section

If the SYNCHRON section consists of one travel block only, the belt motion at the beginning of this travel block will be transferred stepweise to the robot motion and decreased proportionately to the distance covered down to zero during the travel block.

Two travel blocks in the SYNCHRON section

If the SYNCHRON section consists of two travel blocks, the belt motion will be transferred proportionately to the distance covered to the robot motion during the first travel block and decreased proportionately down to zero during the second travel block.
If these two travel blocks are passed, the belt movement at the beginning of the second travel block, which was till then proportionally switched, is stepwise completely switched on and then proportionally decreased to zero.

Process-oriented functions

WAIT at the beginning or at the end of the SYNCHRON section

If instructions are programmed between the SYNCHRON and the first belt-synchronous travel block, which execution time is not negligible, e.g WAIT, then the robot does not move yet SYNCHRON in this period. The same applies to instructions between the last belt-synchronous travel block and the SYNCHRON_END.

The Belt counter

BAPS3 declaration of the belt input channel

For a BAPS3 program, a belt is kinematic-related input channel of kind INPUT REAL. The channel number indicated there is not checked by the BAPS3 translater. The IRD interpreter is informed via the belt channel number offset 500 that the INPUT REAL used is a belt.

In the BAPSplus development environment, the belts are defined at the signals.

Within the rho4, the belt consecutive numbering begins with 1 in the first belt of the first kinematic. Kinematics may have no or several belts. The belt number is also required for several special functions as transfer paramater.

The belt channel number results from the addition of the belt number and the belt channel number offset of 500.

Example

```
                                        ;Robi_1 has the belts 1 and 2

Robi_1.BELT:501=KIN01BLT01,
            502=KIN01BLT02
```

☞ **If several kinematics are arranged with belts in an application on a rho4, it can be useful to arrange the belt numbering via CONST, so that later modifications of the belt number should have a small influence on the program sources.**

Process-oriented functions

Example

```
                                        ;Robi_1 and Robi_3 have 2 belts,
                                        ;Robi_2 has 1 belt

CONST:        BLTCHANN_OFF=500,
              BELT_KIND_PTP=4,
              NRKIN01BLT01=1,
              NRKIN01BLT02=2,
              NRKIN02BLT01=3,
              NRKIN03BLT01=4,
              NRKIN03BLT02=5

Robi_1.BELT: BLTCHANN_OFF + NRKIN01BLT01 = KIN01BLT01,
              BLTCHANN_OFF+NRKIN01BLT02=KIN01BLT02

Robi_2.BELT: BLTCHANN_OFF + NRKIN01BLT01 = KIN02BLT01

Robi_3.BELT: BLTCHANN_OFF + NRKIN03BLT01 = KIN03BLT01,
              BNDCHANN_OFF+NRKIN03BLT02=KIN03BLT02
```

The selection of the belt kind could then e.g be the following in the program:

Belt_Kind (NRKIN01BLT01, BELT_KIND_PTP)

☞ **The names of the BELT channels should be chosen the same as for the JC_NAMES and WC_NAMES.**

```
;;sr6_2B.JC_NAMES=A_1,A_2,A_3,A_4,B01,B02
;;sr6_2B.WC_NAMES=X_A,Y_A,Z_A,C_A,B01,B02

...

sr6_2B.BELT:501=B01,502=B02
```

Machine parameters

The machine parameters for belts are grouped in the kinematic-dependent machine parameter group P500.

A measuring system must be assigned to each belt in the machine parameter P401. The same measuring system can be assigned to several ”logical” belts defined in the P500 group via the P401. Regulated axes can also be arranged as belt measuring systems.

☞ **For belt transmitters at the CAN bus, P30 must also be adopted if required. See also manual machine parameter.**

Parameter no.	Machine parameter converter key words
P501	KinBeltNumb.Kinx
P503	BeltDirCos.Kinx.Beltb.Coordz
P505	BeltCoLimMin.Kinx.Beltb

Process-oriented functions

Parameter no.	Machine parameter converter key words
P505	Belt.CoLimMaxKinx.Beltb
P506	BeltName.Kinx.Beltb
P507	BeltTimeOff.Kinx.Beltb
P508	BeltSimVel.Kinx.Beltb

The belt coupling factors

The direction of the straight line in the space, along which the belt synchronization can shift a point, the belt coupling factor is communicated to the rho4 via the machine parameter P503.

Indication for the (up to) three space axes X, Y, Z:

- The cosine value * the angle, which is included between the belt and the corresponding WC_NAMES
- The root of the sum of the squares from the cosine value has to be as close to 1 as possible. If the value is greater than 1, this corresponds to an increase in the length of the belt direction, and a decrease if the value is less than 1.

☞ **Use setting Deg (not (new–) Grad) at the windows pocket calculator.**

Example 1

The belt runs exactly parallel to the X axis. The following values are to be set with the MP converter:

```
P503.BeltDirCos.Kin1.Belt1.Coord1=1      ;Cos (0 deg)
P503.BeltDirCos.Kin1.Belt1.Coord2=0      ;Cos (90 deg)
P503.BeltDirCos.Kin1.Belt1.Coord3=0
```

Example 2

The belt runs exactly parallel to the Y axis. The following values are to be set with the MP converter:

```
P503.BeltDirCos.Kin1.Belt1.Coord1=0      ;Cos (90 deg)
P503.BeltDirCos.Kin1.Belt1.Coord2=1      ;Cos (0 deg)
P503.BeltDirCos.Kin1.Belt1.Coord3=0
```

Example 3

The belt forms with X coordinate an angle of 30 degrees and with the Y coordinate an angle of 60 degrees. The following values are to be set with the MP converter:

```
P503.BeltDirCos.Kin1.Belt1.Coord1=0,866025      ;Cos;Cos (30 deg)
P503.BeltDirCos.Kin1.Belt1.Coord2=0.5           ;Cos (60deg)
P503.BeltDirCos.Kin1.Belt1.Coord3=0
```

For angles in the range from 90 degrees to 270 degrees, the cosinus value takes negative values.

If the belt coupling factors are cosinus values of an angle, the values are then between −1 and +1.

Application instruction for P401.MeaSysFactor.Kin*x*.Belt*b*

The measuring system factor is usually set in a way that the belt value can be directly read in the millimeter range. For some functions, within rho4, the current belt speed will be calculated. For the selection of the resolution of the transmitter, it must be taken into account that enough increments per scanning interval (P5) are available.

Calculation:

Max. belt speed [Incr. / scann. interval] = P401.MeaSysFactor [Incr. / mm] * max. belt speed.[mm / sec] * P5.Clock start time [millisec] / 1000.

If the result is a value of e.g. 20 incr. / scanning interval, the belt speed expected by the rho4 changes in steps of 5%.

Analogy consideration between axes and belts

For axes, there are further machine parameters influencing the position. The following table refers to similar functions for belts.

Process-oriented functions

P402 Reference point direction	A belt is moved normaly in one direction
P403 Effectiveness reference point switch	The RC input signals BELTxxRES_RCI O59.0 to O60.7 react to positive impulse edges. For SYNC belt variables, input binar = 0 or 1 a reaction is possible to a negative or positive signal state.
P207 Reference point actual value P208 Reference point offset	Via SPC_FCT: 28 = Belt_Set (...)
P202 Software end switch positive WC P203 Software end switch negative WC	P505 Limit values for belt counter. Min. and max. value = 0, the monitoring switches off. It has also effect on the adoption of the belt counter during teaching.
P311 Modulovalue for endless axes P208 Reference point offset	Via SYNC belt variable >= real value

RC input signals

With the RC input signals BELTxxRES_RCI O59.0 to O60.7, it is possible to cause a reset of the belt counter to the active belt counter reset value for each of the max. 16 belts separately. The signals react to a positive edge.

☞ **It must be ensured that the reset of a belt counter does not occur when a process is running synchronously to this belt. If a kinematic has several belts, a belt not used can be reset.**

With the RC input signal O32.7 BELT_SIM_RCI, the belt simulation for all belts can be switched on at the same time. The belt counters change their values with the speed set in the machine parameter P508 or with the velocity changed via SPC_FCT : 55 = Belt_V_Sim (...). For a more realistic simulation, the signal should be connected with similar release signals, as the real belt.

For safety reasons, tests can be performed in the BAPSplus test system in the SYNCHRON section only when the signal BELT_SIM_RCI is set. If the test is to be performed when the belt is stopped, a simulation speed of zero has to be selected.

Reset of a belt counter

If a belt runs endlessly in a direction, computation inaccuracies occur when the powers of ten of the belt counter value increase. In such cases, a reset of the belt counter at appropriate places is therefore required.

Also after the control is booted, a reset of the belt counter is usually required.

Process-oriented functions

The resetting of a belt counter to the belt counter reset value can be performed

- from the PCL program via the RC input signals BELTxxRES_RCI O59.0 to O60.7 intended for this purpose
- from a BAPS3 program with the command SYNC.

State of the belt counter after start–up of the control

After the start–up of the control, the belt counter value is on zero for incremental measuring systems. The reference to the real belt position is still missing. It is created in many applications by an input signal of a light barrier or proximity switch when the belt brings a new part in the work area.

In the use of absolute measuring systems as belt transmitter, the measuring systems have passed the zero position because of the permanent running of the belt in a direction. This leads to a cyclically absolute position. After the start–up of the control, the belt counter value is then on such a cyclically absolute value. Only when it is ensured that the parts are in these cyclical spaces on the belt, it can be worked with the belt counter value without a reference to the actual part position.

For machines for which the belt positions of parts are determined much before the work area of the robot, e.g through a vision system, saved belt positions must be, if necessary before the start, converted into the new belt counter value or cancelled.

Belt counter reset value

The belt counter reset value is the value to which a belt counter is set when a reset is performed. After the start of the control, the belt counter reset value has the value zero. This belt counter reset value can be changed from a BAPS3 program by calling up special function 28. The change applies globally for all processes until the next change or boot.

Declaration

```
SPC_FCT:28 = Belt_Set  (VALUE INTEGER: Belt_No
                        VALUE DEZ: Reset_Value )
```

Belt_No	Number of the belt for which the value is set (1 to 16)
Reset_Value	Value in [mm] to which the internal belt counter reset value is to be set.

The function can only be carried out in the AUTOMATIC mode.

Resetting a belt counter via RC input signals BELTxxRES_RCI

With a positive edge at the RC input signals BELT01RES_RCI up to BELT16RES_RCI (O59.0 .. O60.7), the corresponding belt counter values can be set to the corresponding belt counter reset value.

The resetting of a belt counter via the RC input signals intended for this purpose acts directly, independently from the operating mode MANUAL / AUTOMATIC.

☞ **It must be ensured that the reset of a belt counter does not occur when a process is running synchronously to this belt. If a kinematic has several belts, a belt not used can be reset.**

Resetting of a belt counter via SYNC with RC input signal

With the BAPS3 command 'SYNC belt variable, input binary = Binary value', a belt counter can be reset from a program.

This function can only be carried out in the AUTOMATIC mode.

☞ **The SYNC command may also be used when the program is running synchronously to the belt that is to be reset. (See also section 25.6.4 Belt synchronization with endless belt)**

If the INPUT BINARY does not have the programmed binary value, the program execution will be stopped until the INPUT BINARY corresponds to the programmed binary value. At that moment, the belt counter value is then reset to the belt counter reset value.

If the INPUT BINARY has already the programmed binary value at the time of the execution of the SYNC command, the belt counter value is reset at this moment to the belt counter reset value. There will be no edge analysis. A belt counter value reset in this way is usually useless.

☞ **If such a case is to arise, it should be got under control through an appropriate program branching.**

Resetting of a belt counter via SYNC with belt counter value

With the BAPS3 command "SYNC belt variable >= REAL value" or "SYNC belt variable <= REAL value", the resetting of a belt counter can be triggered from a program in dependence on the belt counter itself.

This kind of belt counter resetting is especially useful when the sections on the belt have the same length. It can be compared to the modulo value of an endless axis.

☞ **The SYNC command may also be used when the program is running synchronously to the belt that is to be reset. (See also section 25.6.4 Belt synchronization with endless belt)**

If, with the command 'SYNC belt variable >= REAL value', the current belt counter value is smaller than the programmed REAL value, the program execution is stopped until the current belt counter value is greater than or equal to the programmed REAL value. At that moment, the belt counter value is set to the belt counter reset value plus the current belt counter value minus the programmed REAL value.

If, with the command 'SYNC belt variable >= REAL value', the current belt counter value is already greater than or equal to the programmed REAL value, the program execution is not stopped. The new belt counter value follows accordingly.

Example

The belt counter should always be in the range of 100 .. 300. This is achieved through a belt counter reset value = 100 and a REAL value = 300.

Through cyclically calling SYNC B01 >= 300, the belt counter is maintained in the indicated range.

- 1st case:
  The current belt counter value is at 301.7
  After execution of the command 'SYNC B01 >= 300', the new belt counter value is then 301,7 – 300 + 100 = 101,7

- 2nd case:
  The current belt counter value is at 7999,9
  This case can arise when the program calling cyclically 'SYNC B01 >= 300', is not (yet) running when the belt is running.
  With a single execution of the command SYNC B01 >= 300 it follows the new belt counter value 7999,9 – 300 + 100 = 7799,9

  The belt counter could be brought by calling 39 times a 'SYNC belt variable >= 300' or a 'SYNC belt variable >= 7900' into this range (to 199,9).

  - The required belt counter reset value for a single call can be calculated in general through the formulas:
    N = Integer part

    Integer part = ((current belt counter value – range lower limit) / (range upper limit – range lower limit))

    REAL value = N * (range upper limit – range lower limit) + range lower limit

  Here:

  Range lower limit = belt counter – reset value = 100

  Range upper limit = 300

  Integer part = ((7999,9 – 100) / (300 – 100)) = 39,4995

N = 39

REAL value = 39 * (300–100) + 100 = 7900

Hence: SYNC B01 >= 7900

Belt rate time for belt kind 4

For the belt synchronization kind 4, positioning errors occur comparatively to the running belt because of the after-runnings of the concerned axes with belt-parallel movement possibility.

Via the machine parameter P507 Belt rate time, they can be compensated. The value of the belt rate time is rounded off internally to the multiple of the machine parameter P5 Clock Start time (belt rate action factor).

The rho4 determines from the belt rate time factor and the current belt speed a position rate action.

The current belt speed is determined from the position difference via a scanning interval (machine parameter P5 clock start time). The screening of the measuring values in increments per scanning interval is to be taken into account. It follows:

P401.MeasuringSystemEvaluation [Incr./mm] * Belt speed [mm/s] * P5.ClockStartTime [ms] / 1000

When the scanning duration decreases, the belt speed is more strongly screened and the belt rate time factor becomes higher (P507).

Via the position rate action, the screening of the belt speed is to be observed as set position fluctuation.

To reduce this effect, it is possible from version VO03F to determine the belt speed as an option from taking the average value of up to 10 measuring values. This occurs via the bit 2, weight 4 of the option byte for the belt input.

Example

Extension phase active and averaging the belt speed

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	0	1	0	1

The value 5 must then be set at the option byte for the belt input. Useful values are 0, 1, 5.

The special function SPC_FCT : 15 = Belt_Param (...) has no influence on averaging.

Process-oriented functions

Positions for belt-synchronous

Positions that are to be run belt-synchronously do not have any fixed position in the robot coordinate system. The actual position in the robot system of coordinates depends on the belt value to the execution moment.

The belt value, for which the point values of the robot coordinate systems apply, is therefore saved together with the values of the robot system of coordinates. The number of the components of a point increases by the number of the belts of the kinematic. Axis number and coordinate number of a kinematic must be equal.

Belt-synchronous points are overdetermined in a mathematical point of view. The same position can be indicated either through the belt value or through corresponding coordinates.

Example: Kinematic declaration of a 4-axis scara robot with 2 belts

```
KINEMATICS (1=sr6_2B)

;;sr6_2B.JC_NAMES=A_1,A_2,A_3,A_4,B01,B02
;;sr6_2B.WC_NAMES=X_A,Y_A,Z_A,C_A,B01,B02

sr6_2B.BELT:501=B01,502=B02
```

☞ **The names of the belts should be chosen equal for the JC_NAMES, WC_NAMES and BELT channels.**

When the belt number increases, all points used in the program must be increased by components for the belts. In general, the additional component for the belts can be set to 0.

The number of components of the points in the PKT files is also increased. The additional component for the belts is automatically set to 0 for these points.

Outside the SYNCHRON sections, the value of the belt components for movements have no meaning. Exception: The last travel block before SYNCHRON for the belt synchronization kinds 2 and 3.

Teaching of points

A belt coordinate is adopted when during the point teaching when the monitoring of the belt limit values with the machine parameter P505 is active. In this case, the belt counter value must have been reset at least once since the start of the control via the RC input signal of the corresponding belt.

If the RC input signal of a belt has not yet been actuated, or if the belt counter is outside the values declared with P505, the warning ”Value of beltcounter” appears.

If the monitoring of the belt limit values with the value 0 as minimum and maximum belt value is deactivated, the value 0 is adopted for the teaching of points for this belt coordinate.

☞ **The belt value of a point is not taken into account when going directly to the teach points.**

Calculation of a position with different belt positions

Belt-synchronous points are overdetermined in a mathematical point of view. The same position can be indicated:

- through the belt value
- through corresponding coordinates

If e.g. all belt-synchronous points are to have the same belt value, the coordinates of points that have been taught at different belt values, can be converted.

With the formula

$$X\_C_{new} = X\_C_{old} + COS\_X * (B01_{new} - B01_{old})$$

the x component of a point can be calculated as an example.

X_C_{old}	the X-component so far
COS_X	is the corresponding machine parameter P503.Belt coupling factor
$B01_{new}$	the required belt value
$B01_{old}$	the belt value so far

For the calculation of the resulting fixed final position of the last belt-synchronous travel block, it follows:

$$X\_C_{fix} = X\_C_{prog} + COS\_X * (\text{"Belt counter current value"} - B01_{prog})$$

Display of the programmed final point

The programmed final point of a movement can be displayed at the PHG:

- via Mode 7.1
- in Rops4-Online via kinematic axis positions
- via the rho function 2033 rKGAxEPos

The values displayed there include only axis resp. coordinate values and no belt values. In belt-synchronous program sections, the programmed belts values are included in the axis resp. coordinate values.

Inpos signals

The Inpos signals

- INPOS_AA_RCO I16.0
- INPOS_1_RCO to INPOS_24_RCO I39.0 to I41.7

Process-oriented functions

indicate that the current hunting is smaller than the inposition area set in the machine parameter 201.

☞ **In belt-synchronous program sections, the belt motion can lead to the fact that an axis is outside the area, even if a motion is completed.**

A MOVE TO is completed only if option “BS: TO wo.INPOS” is active (=1) (is displayed in machine parameter 28).
The hunting in belt-synchronous program section can be reduced via the machine parameter P507.BeltTimeOff.KinxBeltb.

Speeds

Speeds for LINEAR and CIRCULAR

For belt-synchronous LINEAR and CIRCULAR movements, the given speeds are the relative speeds between the workpiece moving with the belt and the robot.

Automatic speed adjustment for PTP movements

For PTP movements, the given speeds for the axes are deduced from the machine parameters via factors. For belt-synchronous movements, the effect of an axis can be additionally strengthened or weakened through the belt movement.

The extent of this effect depends on 2 application depending factors:

- The current belt speed
- The position of the belt-synchronous working area in the working area of the robot.

The axis geometry factors indicate by how many degrees or mm an axis moves when the belt is moved by 1 mm. They can be positive or negative. The factors are automatically determined when the SPC_FCT Belt_Area (...) is called up.

They can be read or overwritten with SPC_FCT 52 = Belt_Ptp_Fac.

Declaration

```
SPC_FCT:52=Belt_Ptp_Fac (VALUE INTEGER: Belt_No
                           JC_POINT:    @Min_Ptp_Fac
                         JC_POINT:    @Min_Ptp_Fac)
```

Belt_No	Number of the belt for which the values are to be set (1..16) or read (–1..–16)
@Min_Ptp_Fac	Minimum axis geometry factor
@Max_Ptp_Fac	Maximum axis geometry factor

Process-oriented functions

The axis geometry factors indicate by how many degrees or mm an axis moves when the belt is moved by 1 mm. They can be positive or negative.

Several kinematics

Variables of type JC_POINT are kinematic related. If this special function should be used for several kinematics, it must be declared under several names e.g. BeltPtpFac01, but the same number (52 = ...) for each kinematic with kinematic-related type JC_POINT (e.g. Robi_1.JC_POINT).

Example

```
SPC_FCT:52=BeltPtpFac01      (VALUE INTEGER:     Belt_No
                              Robi_1.JC_POINT:   @MinPtpFac01
                              Robi_1.JC_POINT:   @MaxPtpFac01)

SPC_FCT:52=BeltPtpFac 02      (VALUE INTEGER :   Belt_No
                               Robi_2.JC_POINT:  @MinPtpFac02
                               Robi_2.JC_POINT:  @MaxPtpFac02)
```

Several belts within a kinematic

If a kinematic has several belts, the special function can be used with the same declaration for each belt of the kinematic.

Effects of the axis geometry factors

By calling SPC_FCT : 53 = Belt_Area (...), the axis geometry factors are automatically determined.

The general request is that the axes that are moved through the belt, must be able to have a (clearly) higher speed than the

max. belt speed * max. geometry factor

If an axis moves in direction of the belt, it can run with a higher speed than indicated in the machine paramters. The given speed can be increased by the

min. belt speed * min. geometry factor

If an axis moves behind the belt, it may only be given a smaller speed than indicated in the machine parameters, to ensure that the resulting speed does not exceed the maximum speed of the axis. The speed must be reduced by the

max. belt speed * max. geometry factor

Process-oriented functions

Test of belt-synchronous programs

The test of belt-synchronous programs is very complex because of the simultaneous movement of the robot and the belt. It should be therefore done gradually. It is to especially pay attention to the possiblity of the belt simulation (BELT_SIM_RCI) and possibly also in combination with the axis simulation (P401.SlotNum.Kin*x*.Axis*y*=X99). The following procedure can be used as a suggestion:

- Belt stationary, small VFACTOR
- Program run for loops, first/last passage, interruptions, restart
- Different positions at which the belt is, with small VFACTOR. Also with clearly too small or clearly too high belt counter values
- Different positions at which the belt is, with normal VFACTOR
- Determination of the program execution time and conversion in min. and max. belt path
- Small belt speed with normal VFACTOR
- Normal belt speed with normal VFACTOR
- Test of the max. belt speed

BapsPlus Test system

For safety reasons, in the BAPSplus test system in the SYNCHRON section, single steps and interruption are only allowed when the signal BELT_SIM_RCI is set. If the test is to be done when the belt is stationary, zero must be selected as simulation speed (P508).

If the porgram execution is interrupted in the SYNCHRON section with a real belt through the BAPSplus-Test system through single step or interruption, the program is interrupted with the state message 'BP not all. (belt-s)', (code 141696), and the READY contact opened.

Reactions from other functions

Parameterization of the belt characteristic

The special function 15 Belt_Param (...) has no effect on the belt kinds 1 and 4.

RC input signals advance / move release

If the processing of travel blocks in the SYNCHRON section of the belt kind 1 or 4 is prevented by

- the collecting signals O16.6 TRAVELAK_RCI
- the collecting signals O17.0 FEEDE_AK_RCI
- the corresponding kinematic single signals

the belt synchronization remains active and the kinematic can be moved at the same time through the belt. Under certain circumstances it can lead to the triggering of the moving area monitoring.

Program interruption / Reset

If a process is interrupted, stopped or reset in the SYNCHRON section, the belt synchronization is ended abruptly.

Process-oriented functions

Reactions on other functions

Limit values for the belt counter

The belt counter is checked if the limit values are observed

- P505.MinLimitvalueBelt.Kin*x*.Belt*b*
- P505.MaxLimitvalueBelt.Kin*x*.Belt*b*

for belt kinds 1 and 4 already at the preparation time of the command SYNCHRON. If the belt counter is outside the given limits, the process is interrupted with the state message ”Value of beltcounter”, (code 141824).

If at the preparation time of the command SYNCHRON, no useful conclusion can be done about the belt counter, the preparation time can be delayed through input inquiries (e.g. WAIT UNTIL Belt variable).

But the monitoring can also be switched off by setting the min. and max. limit value to the value zero. This must be done when several SYNCHRON sections are to be run without a resetting of the belt counter and therefore when no useful limits can be set.

Parallel programming

☞ **It is not permissible to program in the SYNCHRON section parallel pathes with moving instructions of the same kinematic. In such a case, the moving instruction of the subprocess is not executed and therefore the main process is not ended.**

Standard functions WC(), JC()

The standard functions WC() and JC() forward the programmed belt values. Within the SYNCHRON section, the belt values are not used.

WC_SYSTEM

The conversion of the programmed belt value with the values of the machine parameter P503 “Belt coupling factors” into the involved coordinates occurs in WC_UR.

Conversion Right/Left-armed

During PTP Belt-synchronous, a change of the arm side is not allowed.

READ/WRITE

If in READ/WRITE file or device (V24_x, WIN_x, PLC) the type POINT or JC_POINT is used, the data quantity read or written changes through the additional components for the belt values.

For universal communication routines, it is recommended to do without the types POINT and JC_POINT and if required to replace them through component-like access or ARRAY[...] REAL.

Process-oriented functions

Overview of the example programs

- An example for a Scara robot application is given in ROPS4 in the directory ...\BOSCH\rho4\Example\Baps\BA4. A server process generates the positions of non-existant parts. A client process processes the parts. You can find a short description of this in the main.txt file.
- PhgBdSet for setting the belt counter from the PHG.

State messages

State messages arise in a process while the process is in the SYNCHRON section, the READY contact opens for safety reasons besides the state message.

Exceptions to this are the state messages 'BS Begin Limit' and 'BS Take Limit'. A quick restart is provided for with these messages via SPC_FCT : 47 = exc_define (...).

The following table contains an extract of the state messages that can appear in connection with the use of the belt kind 4.

State No.	PHG text	Cause	Instruction
19584 to 19599	BS–Begin Limit Belt : b	It was determined at the beginning of the execution of the first belt-synchronous travel block in belt kind 4, that the end of this travel block could not be reached before the belt counter exceeded the limit value for the latest beginning that was set in SPC_FCT : 53 = BeltArea.	Part no longer reachable. Check execution and travel time between the decision to move the part and the beginning of SYNCHRON. Reduce belt speed if necessary. Change start position to one closer to the belt if necessary. Program restart via SPC_FCT : 47 = EXC_DEFINE (..) if necessary.
19712 to 19727	BS–Take Limit Belt : b	It was determined during the execution of the belt-synchronous program section of belt kind 4 that the belt counter exceeded the set limit value for the belt area length that was set by SPC_FCT 53 = Belt_Area (...) or would exceed it while executing a travel block.	The belt-synchronous processing cannot be completely finished. Check the processing time needed. Reduce belt speed if necessary. Start earlier with processing or the latest start if necessary. Program restart via SPC_FCT : 47 = EXC_DEFINE (..) if necessary.
21760 to 21775	BS-workroom limit Kinem.: k	Kinematic is in singular position	Move kinematic into valid range in Manual. Check programmed points and belt position

Process-oriented functions

State No.	PHG text	Cause	Instruction
22528 to 22543	BS-ambiguity Kinem.: k	Programmed point and interpolation direction do not agree	Check belt position
131968 to 131983	BS–working area Belt : b	SPC_FCT: 53 = BeltArea (...) was called up with incorrect parameters or not called up in the belt-synchronous program section or before use of belt kind 4.	Check parameters. Area end outside of the travel range?
132096 to 132111	BS –End ambiguity Belt : b	An error occurred during recognition of the last belt-synchronous travel block.	Error location SYNCHRON_END: No last travel block was recognized. Error location travel block: A second last travel block was recognized.
141696	BP not all.	A breakpoint in the BAPS plus testsystem was reached during a belt synchronization was active	Set no interruptions in a SYNCHRON section or use the interface signal BELT_SIM_RCI (Belt simulation)
141824	Value of beltcounter	At the preparation time of the SYNCHRON command the belt counter was not reset by the corresponding interface signal BELTxxRES_RCI or the beltcounter is outside the range defined in machine parameter P505.	Reset belt counter. Increase positive flank or range or switch off the monitoring with min=0, max=0
143360	no belt–option	Belt synchronization is disabled.	Option for purchase. Can be enabled by Bosch customer service.
147200	wrong belt-kind/-tol	By calling the SPC_FCT 21, a non-authorized belt kind or a non-authorized belt tolerance have been programmed.	Check BAPS program
147712	Inadmiss. Belt-No	A belt with the indicated number in SPC_FCT 15, 21, 28, 51 or 52 is not available.	Check BAPS program

Process-oriented functions

Program example PhgBtSet.QLL

```
;--------------------
;Example program
;PhgBdSet for setting the belt counter from the PHG
;--------------------
;The program determines the current belt counter reset value via the
;SYNC belt = belt and sets the belt counter to the desired value via the
;SYNC belt
;The number of axes and the current belt number have to be adjusted
;with a different configuration.
;--------------------

;;SER_IO_STOP-
;;KINEMATICS:(1=SR6_BLT)
;;SR6_BLT.WC_NAMES=K01,K02,K03,K04,BLT
;;SR6_BLT.JC_NAMES=A01,A02,A03,A04,BLT

PROGRAM PhgBtSet

CONST:
    CurBltNo = 1

REAL :  BltResetVal, BltNom

BELT :  500+CurBltNo = BLT

BEGIN

  WRITE PHG, CLS, 'set belt counter',
                  'via SYNC Blt >= xx',
                  '(Reset o.-1=abort)',
                  'Nom.Value='

  READ PHG, BltNom

  IF ( ABS(-1.0-BltNom) < 0.001 ) THEN
    BEGIN
      WRITE PHG, CLS, 'unchanged', BLT
      STOP
    END

  ;Determine belt counter reset value
  SYNC BLT >= BLT
  BltResetVal = BLT

  IF BltNom >= BltResetVal
    THEN SYNC BLT >= (BltResetVal - (BltNom - BltResetVal) )
    ELSE SYNC BLT <= (BltResetVal - (BltNom - BltResetVal) )

  WRITE PHG, CLS, 'belt counter=', BLT

PROGRAM_END
```

Process-oriented functions

25.6.4 Belt synchronization with endless belt

General information

In some applications it is necessary that an endless axis is permanently moved in synchronization with an other uncontrolled axis (belt). For this purpose it is necessary that the master axis (belt) is reset cyclically to avoid an internal counter overflow.

Function

The BAPS command SYNC 'beltname' resets the belt counter. This command was so far only allowed to be programmed outside the synchronous movement.

By means of the function described here it is possible to also program the command SYNC 'beltname' within the synchronous condition. The belt counter is reset without interfering with the synchronous movement.

Example 1

```
;;CONTROL=rho4
;;KINEMATICS:(1=robi1)
;;robi1.JC_NAMES=a_1, blt
;;robi1.WC_NAMES=k_1,bt1

PROGRAM bs_vs

INPUT:    1=i1

robi1.BELT: 501=bt1

INTEGER:    mod_value

;;KINEMATICS=robi1
;;INT=LINEAR
```

Process-oriented functions

```
BEGIN

   AFACTOR=1, DFACTOR=1, VFACTOR=1, V=100
   mod_value=1000
   SYNC bt1, i1=1

   MOVE LINEAR TO POS
    SYNCHRON robi1 bt1

    loop:

    SYNC bt1 >= mod_value
    WAIT 1
    IF i1=1
    THEN JUMP loop

    SYNCHRON_END bt1

PROGRAM_END
```

With the command SYNC bt1>=10000 this limit value, when exceeding the programmed limit value, in the example mod_value=1000, is deducted from the current belt counter value. The belt-parallel axis a_1 follows the belt continuously, also during the reset.

The program is stopped in the SYNC block until the programmed SYNC condition is met. The program is only continued after having executed the SYNC command.

Example 2

```
   .
   .
   SYNC bt1, i1=1

   MOVE LINEAR TO POS
   SYNCHRON robi1 bt1

      loop:

      IF @POS.bt1 < mod_value THEN JUMP no_sync
      SYNC bt1 = mod_value

      no_sync:

      MOVE_REL LINEAR (10.0)
      MOVE_REL LINEAR (-10.0)

      IF i1=1 THEN JUMP loop

    SYNCHRON_END bt1

PROGRAM_END
```

In example 2, the command SYNC bt1>=1000 is only active when the programmed limit value for the belt counter has been exceeded. It is then possible to execute further BAPS commands within the loop. In the example, the belt-parallel axis moves relative to the belt forward and backward.

Restrictions

When resetting the belt counter within the belt-synchronous movement, the following restrictions have to be taken into account:

- A programming of absolute positions on the belt, as offered by the expansion level of the belt synchronization, is not possible since the belt counter value is no longer available as absolute value. A teach-in of positions relative to the belt is not possible.
- The belt-parallel axis must be declared as endless axis.
- Within the belt-synchronous movement, only MOVE_REL commands for the belt-parallel axis are reasonable.
- Within the synchronous movement, the belt value must not be reset depending on an input signal, e. g. SYNC bt1, i1=1.

Process-oriented functions

25.7 Tool change

With the function tool change it is possible to adapt the coordinate transformations of the individual kinematics

- to the effective tool (e. g. with bundle or revolver grippers)
- to assembly inaccuracies

The path control always refers to the Tool Center Point (TCP) of the active tool (gripper).

Coordinate transformation

To be able to work with different tools (grippers) in the same world coordinate system, the coordinate transformation is split into several parts.

(1) The first part takes into account the robot kinematic to the flange which is determined via machine parameters. This is determined by the robot type (P306), axis number (P302), arm lengths (P307) and coupling factors (P308). The flange cannot be changed by BAPS.

(2) The second part takes the kinematic of the tool from the flange to the Tool Center Point into account. There are 2 possibilities of tool definition, which can also be combined with each other:

(2a) A default tool can be defined via machine parameter P309 (flange coordinates). This is active after the start-up of the control. The zero point of the flange coordinate system is within the flange determined by (1). As flange coordinates three translations
(FL_X,FL_Y,FL_Z)
and three rotations
(FL_O1,FL_O2,FL_O3) are entered.

(2b) You can define several tools using the tool.dat file. None of these tool coordinate systems are active after the control has been booted. They have to be activated from BAPS and can be changed online during operation. They replace the standard tool (2a).

For the general description again three translations
(G_X,G_Y,G_Z)
and three rotations
(G_O1,G_O2,G_O3) are used.

The zero point of the tool coordinate system is determined by (1).

Remarks

The orientation of the flange or of the tool coordinate system and the definition of the orientations depend on the robot type. For further details, please refer to the corresponding transformation documentation.

FL_Z and G_Z show, however, in case of all kinematics into the working direction of the flange resp. of the tool.

In case of the standard tool, the orientations are defined as follows:

FL_O1 (G_O1), rotation around the axis FL_Z (G_Z)

FL_O2 (G_O2), rotation around the resulting axis FL_Y' (G_Y')

FL_O3 (G_O3), rotation around the resulting axis FL_X" (G_X")

Which of the orientations

FL_O1, FL_O2, FL_O3, G_O1, G_O2, G_O3

will actually enter into the transformation, depends on the robot type, especially on the number of axes and thus the number of the degrees of freedom of the robot.

25.7.1 Structure of the file TOOLS.DAT

As mentioned before, all tool coordinate systems coming into consideration are stored in the file Tools.dat.

TOOLS.DAT is the reserved name for the file to be created by the user himself. The filename depends on the selected language in machine parameter P10.

The individual tools are given a name that can be freely selected by the user. Under this name, the associated coordinates are then stored.

The file TOOLS.DAT is edited as ASCII file in the robot operating system or offline. For each line, one tool name and all associated coordinate values are entered as follows:

Syntax

```
tool_name = G_X G_Y G_Z G_O1 G_O2 G_O3
```

whereby G_X , ... , G_O3 represents the corresponding numeric value.

The tool_name must be at the beginning of the line and its length must not have more than twelve characters. It can be freely selected.

The tool_name and the coordinate values are to be separated by equal signs '='.

Process-oriented functions

The order of the individual coordinates: at first G_X, then G_Y, ... at last G_O3, has to be maintained by all means. The individual coordinate data are to be separated by space characters (the number is unlimited). They are decimal values, whereby the decimal point has not in any case to be set. For the coordinate values, only the entries ’0’, ’1’ to ’9’, ’+’, ’–’ and ’.’ are permitted.

Missing values must be preceeded by explicit zeros. If for one tool less than six coordinates are entered in one line, the message ’format error in DAT’ will be put out at the runtime.

Comments at the end of the line are permitted. They must start with ’;’. Complete comment lines are also allowed. They too must start with ’;’.

Empty lines are redundant.

The translations (these are the first three values) are entered in [mm] and the rotations (the last three values) in [degrees].

Example for file TOOLS.DAT

Syntax	**Description**
IC_GRIP_10 = 10 2.5 5 1 2 3	;IC grippers No. 10 and No. 11
IC_GRIP_11 = -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 50 200 0 0 0	;bundle gripper No. 5 behind right
OFF = 0 0 0 0 0 0	;dummy gripper for switching off

Process-oriented functions

25.7.2 Tool selection in the movement program

The selection of a specific tool is made in the BAPS program with the command

TOOL kin1 grp_name
kin1 = name of the kinematic, for which the tool is activated resp. default kinematic, if no name is indicated.
grp_name = in TOOLS.DAT defined name of the tool (gripper) to be activated.

Example

In a magazine are e. g. to be several bundle grippers.

The individual grippers of the bundle grippers No. 5 are assumed to be defined in the file TOOLS.DAT with the following names:

BUNDLE5_F_L,
BUNDLE5_F_R,
BUNDLE5_B_L,
BUNDLE5_B_R

The selection of the left gripper behind the bundle gripper No. 5 for the kinematic SCARA_1 is then made by

TOOL SCARA_1 BUNDLE5_B_L

i. e. in the coordinate transformation of the SCARA_1, the values of the tool coordinate system BUNDLE5_B_L are inserted.

A tool change is only possible in the automatic mode.

A gripper remains effective until the next call of TOOL.

At the program end, the tool programmed last remains active.

In case of a program abort (e. g. EMERGENCY STOP input, auto-manual switching, reset, etc.) the tool which is active until the time of cancellation remains active.

After the start-up of the control (depending on the machine parameter P309) there is

(a) no tool is active when all flange coordinates in P309 are identically equal to zero

(b) the tool with the name 'MP tool' active when at least one of the flange coordinates in P309 is different from zero.

There is no direct command to switch off the gripper or the tool.

Process-oriented functions

The desired effect can, however, be easily realized by defining for example a dummy gripper with names 'off' and the associated coordinates G_X = ... = G_O3 = 0 in TOOLS.DAT (see example). The BAPS command 'TOOL off' will then lead to a switching off.

When teaching-in in the operation mode Define/Teach In (mode 4.2) or when jogging in the manual mode (mode 2), it is possible via a corresponding program to select the desired tool through external program selection.

For example by means of the following program

```
PROGRAM BUNDLE5FL
BEGIN
  TOOL SCARA_1 BUNDLE5_F_L
PROGRAM_END
```

By this command sequence the TCP of the Scara_1 and thus the world coordinates are realized internally. They then refer to the point of the front left gripper of bundle gripper No. 5.

This is done without travel movement since the machine coordinates have remained unchanged.

The following additional messages are put out in the case of error at the runtime

"no TOOLS.DAT"
"no tool name"

Under

(mode 7, mode 1, 'shift ->'),
(mode 2, 'shift ->') and
(mode 4, mode 2, 'shift ->')

each the names and coordinates of the currently active tools (gripper) of the individual kinematics are displayed.

☞ **When starting the program, it has in any case to be made sure that the correct tool is activated. If programs with wrong tools are started, unexpected movements can be the consequence. The same effect can occur if a WC point in the program is approached and taught in with different tools.**

In the following program examples P309 is each identical with zero, i. e. the origin of the tool coordinate system is in the flange.

Program example

```
0 TOOL Scara_1 BUNDLE_F_L
1 MOVE Scara_1 LINEAR corner_left
2 TOOL Scara_1 BUNDLE_F_R
3 MOVE Scara_1 LINEAR corner_right
4 TOOL Scara_1 BUNDLE_B_L
5 MOVE Scara_1 LINEAR placing
6 TOOL Scara_1 OFF
```

Process-oriented functions

associated file TOOLS.DAT

BUNDLE_F_L	=	–50	–50	200	0	0	0
BUNDLE_F_R	=	–50	50	200	0	0	0
BUNDLE_B_L	=	50	–50	200	0	0	0
BUNDLE_B_R	=	50	50	200	0	0	0
OFF	=	0	0	0	0	0	0

Process-oriented functions

Effect of the command sequence on scara_1

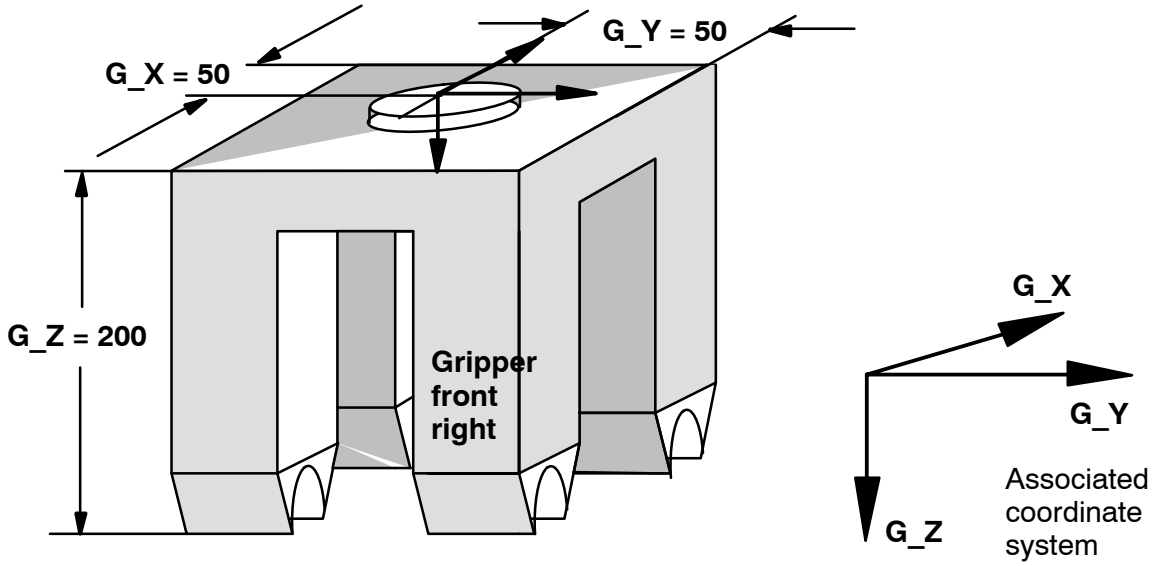
(1) Point corner_left is approached by the left front gripper.

(3) Point corner_right is approached by the right front gripper.

(5) The point PLACING is approached by the left back gripper.

(6) The Tool Center Point is now within the flange. Bundle gripper

Bundle gripper



Program example

The working direction of the grippers is always directed vertically downward. Gripper No. 2 is turned down uncontrolled by means of digital outputs.

```
;;KINEMATICS=portal_2

TOOL revolv1_g1

MOVE LINEAR start_pos ;switch over mechanics so that gripper 2 points downward
                      ;(e. g. by setting digital output signals)

TOOL revolv1_g2

MOVE LINEAR dest_pos

TOOL off
```

associated file TOOLS.DAT

revolv1_g1	=	0	0	100	0	0	0
revolv1_g2	=	0	0	92	0	0	0
off	=	0	0	0	0	0	0

Effect of the command sequence

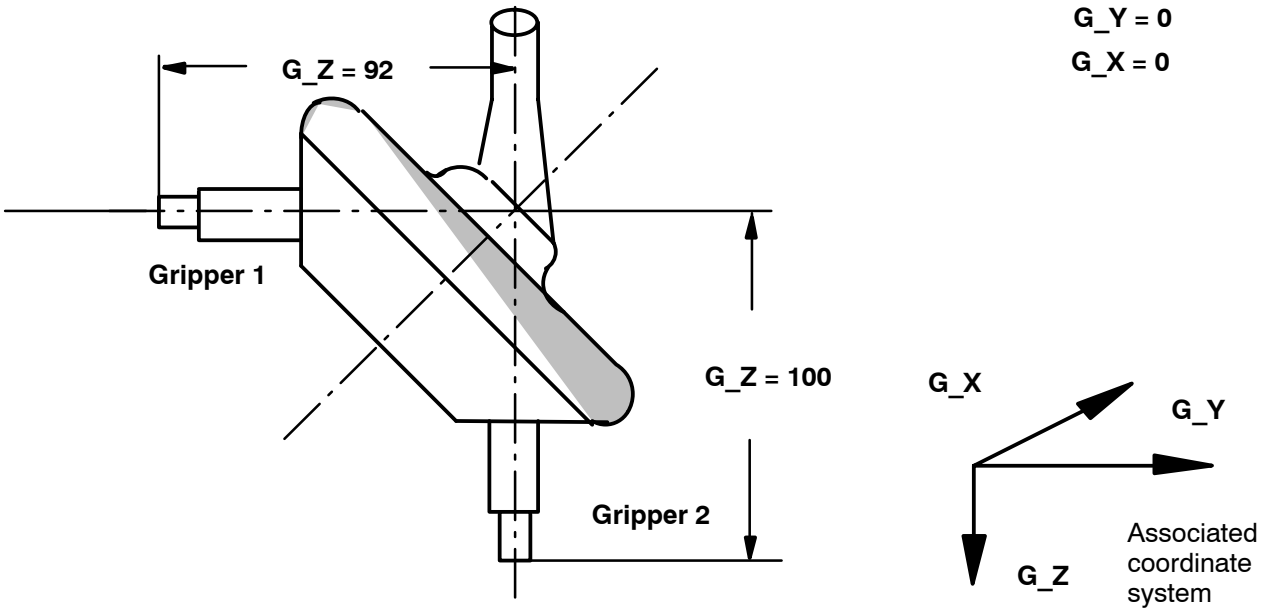
The START_POS point is approached from PORTAL_2 (default kinematic) by gripper 1. Switch over mechanics so that gripper 2 points downwards (e.g. by setting digital output signals).

(4) The DEST_POS point is approached by gripper 2.

(5) The Tool Center Point of the PORTAL_2 is now within the flange.

Process-oriented functions

Revolver gripper



Process-oriented functions

25.7.3 Selection and function in manual mode

The function 'tool coordinate system' acts for each kinematic global, i. e. exceeding program and process limits. If individual part programs are called at several places of the overall process, it has to be made sure, as far as the program technique is concerned, that they always work with the same tool coordinate system.

As mentioned before, a change of the tool coordinate system is only possible via the selection in a non-permanent BAPS program. Since after the program end the tool coordinate system actived last remains active.

In manual mode it is possible by the PHG2000, to display the active tool coordinate system, resp. to select a special one.

In PHG-Mode 2 (Manual) and 4.2 (Teach In), the 2. left key of the second row (from the top) is used as a function key (softkey, keycode "UNTIL"). Entering this key, the following mask is displayed:

TOOLS.DAT	_X O1	_Y O2	_Z O3
BUNDLE_F_L	100,123 40.345	200.123 50.567	300.234 60.321
BUNDLE_F_R	654,111 77.777	543.222 88.888	432.333 99.999
BUNDLE_B_L	454,111 77.777	343.222 88.888	332.333 99.999
BUNDLE_B_R	120,000 44.444	222.222 55.555	333.333 66.666
REVOLV1_G1	254,111 77.777	143.222 88.888	232.333 99.999
REVOLV1_G2	100,123 40.345	200.123 50.567	300.234 60.321
PALLET_2	110.123 45.345	220.123 55.567	330.234 65.321

In the last two lines the active local tool is displayed.

In the middle part of the mask the content of the system file TOOLS.DAT is displayed.

The invers displayed tool can be activated by the <Enter> - key. Thereby the permission key (deadman) must be released.

Using the following key combinations makes it possible to navigate with the cursor within the tool list:

Process-oriented functions

		Standard key assignment
< Cursor–Up >	moves cursor up	< Shift > < 5 >
< Cursor–Down >	moves cursor down	< Shift > < . >
< '<' >	moves cursor one page up	< Alt > < 7 >
< '>' >	moves cursor one page down	< Alt > < 8 >
< BEGIN >	moves Cursor to begin of the list	< BEGIN >
< END >	moves cursor to end of the list	< END >
< Cursor–Left >	exit mask	< Shift > < 1 >

Process-oriented functions

25.8 External program/process disselection

With this expansion level the user can stop his programs/processes specifically also via PLC, analog to the external program selection.

Via an 8-bit data channel, a parity bit and a strobe signal, the process to be stopped can be selected from the file Exprog.dat as with the external program selection.

In this case the same signals as with the external program selection are used.

As acknowledgement the process is either aborted or, if this was not possible, 'process abort error' is transmitted as strobe signal to the interface.

In case of a correct process cancellation, the process is stopped directly without ending an active movement block.

If the main process is stopped, all other sub-processes started with it will also be stopped at the same time.

Permanent processes are, independently from the signal 'permanent processes should remain active', aborted as normal processes.

RC input signals

For the selection of the processess to be stopped, the 8 data bits of the external program selection, the cancellation signal (strobe) as well as a parity bit are required.

Process-oriented functions

RC inputs

Cur. No.	PLC symbol name	rho4 interface address	Signal description
256	STBEXTPS_RCI	O 32.0	**Strobe, external program cancellation:** In case of signal change from '0' to '1' the process selected via the data channel 'external program selection' and Exprog.dat is aborted. Correct cancellation is acknowledged by the strobe signal (RCO no. 164). If the program could not be cancelled, the control will signal this by means of the strobe signal (RCO no. 165).
258	PTY_EXTP_RCI	O 32.2	**Parity, external program selection/cancellation:** Has to be set according to the parity set via machine parameter P4. In case of wrong parity with external program selection, the error signal RCO159 is set, in case of an external process cancellation the error signal RCO165 is set.
264 to 271	EXTPRG_0_RCI to EXTPRG_7_RCI	O 33.0 to O 33.7	**External program selection/cancellation, bits 1 to 8 (bit significance 1 to 128):** 8-bit wide data channel for external selection or cancellation of programs/process. The data are interpreted in the control as 2-digit hex. figures (value 00'H to FF'H). By means of a reference list (Exprog.dat) this hex figure is then used to determine an associated program name and select or abort this program, if it exists. It is possible to select 256 different programs. In case of error: output error strobe (RC output no. 159).

RC output signals

In case of a successful cancellation of the selected process, the acknowledgement signal 'process aborted' is set.

If the selected process could not be aborted because e. g. no process with a corresponding name exists or because of a wrong parity, the signal 'process abort error' is set.

The strobe times of these signals can be set via the machine parameter P9.

Restrictions

It is not admitted to select a process via external program selection and stop at the same time another or the same process via 'external process abort'. This means that both strobes must not be given at the same time.

A corresponding interlocking has to be realized by the user in his adaptation program (PLC).

Structure of Exprog.dat

In the file Exprog.dat an assignment between code value and the program is defined which is to be aborted at the given code value and strobe. The default of the code value to be hexadecimal.

Process-oriented functions

Example of an Exprog.dat

Code value		Program name (name of the .ird file)	Comment
00	=	Prog1	;comment separated by semicolon
01	=	ProgXY	
02	=	ProgAB	
.			
FF	=	ProgFF	

Process-oriented functions

25.9 File and user memory functions

25.9.1 Change file attributes

Mode 9.9 file attributes

Under this mode, the attributes of the files located in the user memory can be modified via the PHG.

☞ **The attribute 'Hidden' can only be set when the supervisor password (mode 7.8.4) is set.**

Admissible attributes are:

R (Read)	File can only be opened for reading
W (Write)	File can only be opened for writing
D (Delete)	File can only be deleted
H (Hidden)	File is not displayed in the control in 'RC list' neither in online

The available attributes are R, W, D.

☞ **All attributes are effective only with file operations in the control.**

25.10 User memory functions via PHG

Mode 9.6 Save UsMemory

The whole user memory of the rho4 is saved on the hard disk (dump). The files cannot be read under Windows since the whole user memory is saved as binary file.

Mode 9.7 Read UsMemory

The copy existing on the hard disk (dump) of the rho4 user memory is back saved in the control. Files that may exit in the rho4 will be overwritten or deleted.

Process-oriented functions

25.11 Save user memory via rho4 function (only rho4.1)

The user memory of the rho4.1 robot control (real time) is saved completely on the hard disk at each shut down. At the start of the control, this backup will be copied back into the user memory. Moreover a backup can be performed with the help of the PHG function 'Store Us. memory' (Mode 9.6), however only in the case of EMERGENCY STOP.

The backup occurs in every case as dump, i.e. all user programs are written on the hard disk in a file. An access via Windows or DOS functions on user files is not possible.

To be able to perform a user-controlled saving of the user memory via the library functions, the rho4 library from version VO 04H makes the function rSSaveUSMEM available.

The use of this function is useful e.g. for machines in permanent operation (no shut down of the control) to enable an archiving of the momentary state or the modifications. The user memory saved in this way is loaded at the next start/restart of the robot control when a shut down error has been detected.

25.11.1 Rho4Fkt: rSSaveUSMEM()

The function rSSaveUSMEM() of the rho4 library creates a copy of the user memory in the RAM of the real time core when it is called. After the creation of the copy, it is written in a second step on the hard disk of the PC.

The function can be called from Windows and via BAPS.

25.11.2 Status of the user memory via rho4Fkt: rSStateUSMEM()

To determine if saving the user memory is necessary, its status can be requested with the help of the function rSStateUSMEM().

Are supplied as return values:

- Number of the completed write accesses on DAT-files
- Number of the completed write accesses on PKT-files*
- Number of the completed write accesses on IRD-files*
- Number of the completed write accesses on QLL-files
- Number of the completed write accesses in total, except DAT, PKT, IRD, QLL

* Programs (IRD- and PKT-files) that are processed, are only read-open, i.e. modifications for these file types are not captured.

The function can be called from Windows and via BAPS.

Process-oriented functions

25.11.3 Configuration of the function rSSaveUSMEM

The function can be configured in the initialisation file. An example file 'Winrho4.ini' is in the directory c:\Bosch\rho4\origin. All entries in this file correspond to the default values.

The function rSSaveUSMEM can be deactivated with the entry 'rSSaveAktiv = 0', in this way the rho4.1 needs 4MB RAM less.

With the key 'LoadAlways = 1' it is determined if in the case of a detected shut down error the control should always take the backup files (~dfak_.sav, ~dfa_.sav) or only when they are younger than the dump files (~dfak_.bin, ~dfa_.bin).

```
;;-------------------------------------------------------------

;; Configuration of function rSSaveUSMEM since version VO04x

;;-------------------------------------------------------------

[SaveUserMemory]

        ;; Disable/enable function rSSaveUsmem();at minimum 64MB RAM needed

        ;; Default: rSSaveUSMEM = 1 means enabled

rSSaveAktiv = 1

        ;; if the lockfile is not found, then load always

        ;; the user memory saved with rSSaveUSMEM(),

        ;; even if it's older than the last good known

        ;; memory dump

        ;; Default: LoadAlways  = 0 don't do that

LoadAlways  = 0
```

The file 'Winrho4.ini' must be in the directory 'c:\Bosch\rho4\winexe'. Modifications are only effective after the restart of the application 'Winrho4.exe'.

Process-oriented functions

25.11.4 Combination rSSaveUSMEM and restart of the RC

rSSaveUSMEM creates two backup files on the hard disk of the PC. When the control is shut down, bin files are created.

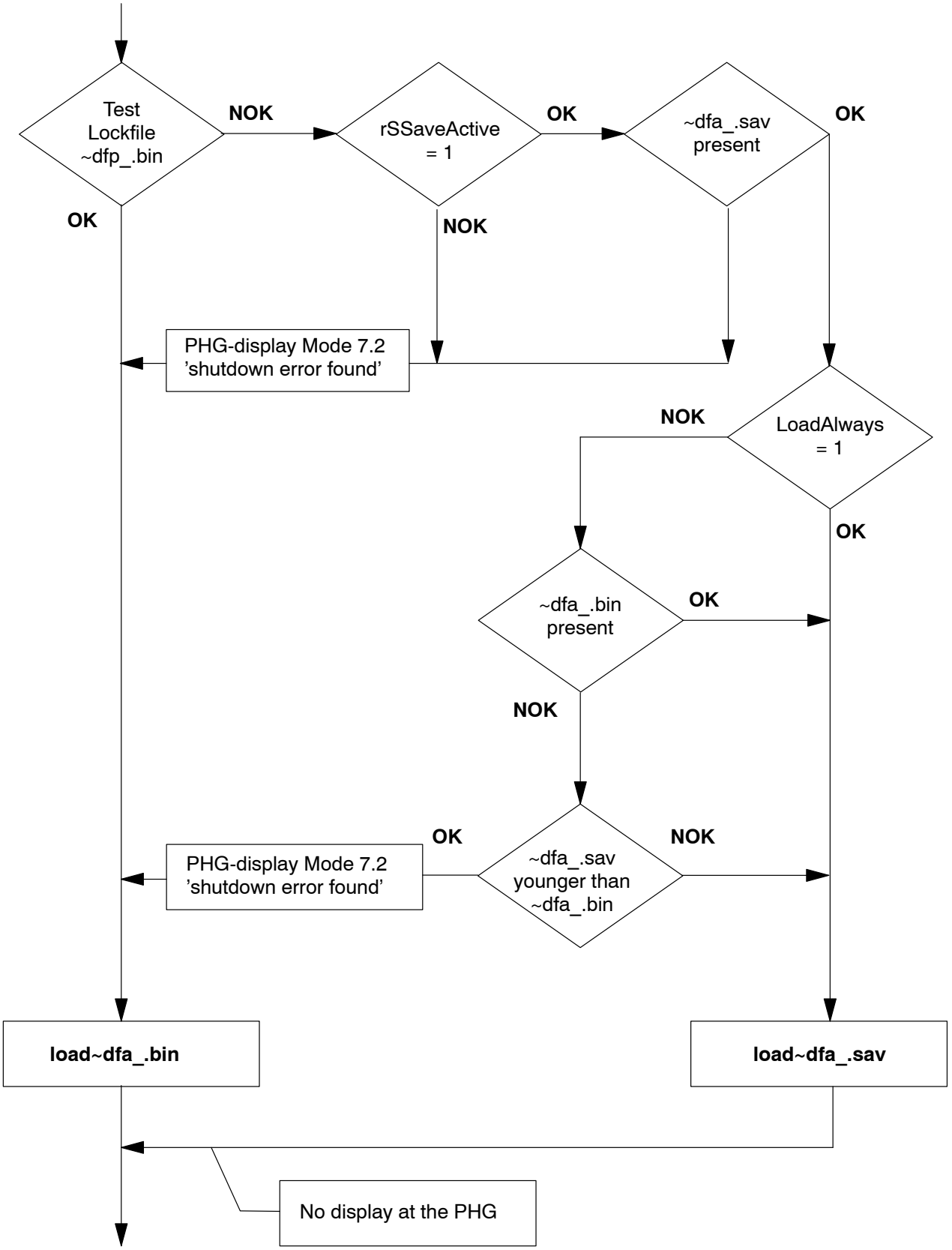
	User memory	Machine parameter	Lockfile
rSSaveUSMEM created	~dfa_.sav ~dfak_.sav	————	————
Shutdown RC created	~dfa_.bin ~dfak_.bin	~dfm_.bin	~dfp_.bin

At the next boot of the control unit, it is checked if the last shutdown ran correctly.

If this was the case, the Bin files (~dfa_.bin, ~dfak_.bin) are then loaded and the backup files (~dfa_.sav, ~dfak_.sav) are not taken into account. If the Lockfile (~dfp_.bin) is not present, the backup failes are loaded depending on the settings in the 'Winrho4.ini'.

Process-oriented functions

25.11.5 Structogramm of the startup logic



Process-oriented functions

Notes:

BAPS3 keywords

26 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 subprogram names in a BAPS3 program.**

Deutsch	Englisch
@	@
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
KREIS	CIRCULAR

BAPS3 keywords

Deutsch	**Englisch**
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_ANFANG	WRITE_BEGIN
SCHREIBE_ENDE	WRITE_END
SEMAPHOR	SEMAPHORE
SONST	ELSE
SOWIE	ALSO
SPRUNG	JUMP
SPZ_FKT	SPC_FCT
START	START

BAPS3 keywords

Deutsch	Englisch
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

Deutsch	**Englisch**
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

Deutsch	Englisch
@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

Deutsch	**Englisch**
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_CALC
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

Deutsch	**Englisch**
CLS	CLS
RK_UR	WC_UR
VERSION	VERSION

BAPS3 keywords

General BAPS3 keyword list

Deutsch	Englisch
@	@
@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

Deutsch	**Englisch**
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

Deutsch	Englisch
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_CALC
RK_SYSTEM	WC_SYSTEM
RK_UR	WC_UR
RSPRUNG	RETURN
RUNDUNG	ROUND
SATZ_SLOPE	BLOCK_SLOPE
SCHLIESSE	CLOSE
SCHREIBE	WRITE
SCHREIBE_ANFANG	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

Deutsch	Englisch
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

Deutsch	Englisch
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:

Annex

A Annex

A.1 Abbreviations

Abkürzung	Bedeutung
Abbreviation	**Description**
Aux.fct.	Auxiliary function
C:	Drive name, in this case drive C (hard disk drive)
CCOMP	Cross Compensation
CEST	Central European Summer Time
CET	Central European Time
ESD	Electro-Static Discharge Abbreviation for all terms relating to electro-static discharge, e.g. ESD protection, ESD hazards, etc.
Fx	Function key with number x
GOM	Group Operating Mode
HP	Main Program (’Hauptprogramm’)
LSEC	Lead Screw Error Compensation
MDI	”Manual Data Input” mode
MP	MACODA configuration parameter
MSD	Machine-Status Display
MTB	Machine-Tool Builder
NC, CNC	Numerical Control (Computerized Numerical Control)
OI	Operator Interface
PE	Protective Earth
PLC	Programmable Logic Controller
SK	Softkey
SP	Subprogram
UTC	Universal Time Coordinated (formerly GMT)

Annex

A.2 Index

Annex

Annex

Notes:

Rexroth
Bosch Group

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 93 52-40-50 60
Fax +49 93 52-40-49 41
service.svc@boschrexroth.de
www.boschrexroth.com

Printed in Germany
DOK-RHO*4*-STEUERFUNK*-PR07-EN-P