

Python Runtime App

Python Runtime Environment for ctrlX CORE 01VRS

Copyright

© Bosch Rexroth AG 2023

All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.

Liability

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

DOK-XCORE*-PYR*****V01-AP01-EN-P

DC-AE/EPI5 (MiSc/PiaSt)

Table of contents

1	About this documentation	5
2	Important directions on use	7
2.1	Intended use.	7
2.1.1	Introduction.	7
2.1.2	Areas of use and application	7
2.2	Unintended use.	8
3	Safety instructions	9
4	Introduction and overview	11
4.1	Python runtime environment.	11
4.2	Integrated library - motion.	13
4.2.1	Python Misc.	13
4.2.2	Python states.	13
4.3	Integrated library - datalayer.	15
4.3.1	Function (nodes) = datalayer.browse (<path>).....	15
4.3.2	Function <value> = datalayer.read (<path>).....	16
4.3.3	Function datalayer.write(<path>, <value>).....	16
4.3.4	Function datalayer.create(<path>, <value>).....	16
4.3.5	Function datalayer.remove(<path>).....	16
4.3.6	Function datalayer.read_json(<path>).....	16
4.3.7	Function datalayer.write_json(<path>, <value>).....	16
4.3.8	Function datalayer.create_json(<path>, <value>).....	17
4.3.9	Further information.	17
5	Commands	19
5.1	Axis commands.	19
5.2	Kinematic commands.	27
5.3	Kinematic command options.	33
5.4	General commands.	36
5.5	Additional information.	38
6	Related documentation	41
6.1	Overview.	41
6.2	ctrlX AUTOMATION.	41
6.3	ctrlX WORKS.	41
6.4	ctrlX CORE.	42
6.5	ctrlX CORE Apps.	42
7	Service and support	47
8	Index	49

1 About this documentation

Editions of this documentation

Edition	Release date	Note
01	2023-07	First edition Python Runtime App Version PYR-V-0120

2 Important directions on use

2.1 Intended use

2.1.1 Introduction

Rexroth products are developed and manufactured to the state-of-the-art.

The products are tested prior to delivery to ensure operational safety and reliability.

⚠ WARNING

Personal injury and damage to property due to incorrect use of products!

The products may only be used as intended.

Failure to use the products as intended may cause situations resulting in property damage and personal injury.

NOTICE

Damages resulting from unintended use

Rexroth As the manufacturer does not assume any warranty, liability or compensatory claims for damages resulting from unintended use of the products. The user alone shall bear the risks of an unintended use of the products.

Before using Rexroth products, make sure that all the prerequisites for an intended use of the products are met:

- Personnel that in any way, shape or form uses Rexroth products must first read and understand the relevant safety instructions and be familiar with their intended use
- Leave hardware products in their original state, i.e., do not make any structural modifications. It is not permitted to decompile software products or alter source codes
- Do not install damaged or defective products or commission them
- It has to be ensured that the products have been installed as described in the relevant documentation

2.1.2 Areas of use and application

Products of the ctrlX series are suitable for Motion/Logic applications.

NOTICE

Products of the ctrlX series may only be used with the accessories, mounting parts, and other components specified in this documentation. Components that are not expressly mentioned must neither be attached nor connected. The same applies to cables and lines.

Only to be operated with the hardware component configurations and combinations expressly specified and with the software and firmware specified in the corresponding documentations and functional descriptions.

Products of the ctrlX series are suitable for single-axis as well as for multi-axis drive and control tasks. Device types with different equipment and interfaces are available for using the system in specific applications.

Typical areas of application:

- Building automation
- IoT and Security Gateway or Device
- Handling & Robotic

Controls of the ctrlX CORE series may only be operated under the mounting and installation conditions, in the position of normal use and under the ambient conditions (temperature, degree of protection, humidity, EMC, etc.) specified in the related documentations.

2.2 Unintended use

"Unintended use" refers to using the ctrlX products outside of the above-mentioned areas of application or under operating conditions and technical data other than described and specified in the documentation.

ctrlX products must not be used if they are exposed to following conditions:

- Operating conditions that do not meet the specified ambient conditions. Operation under water, under extreme temperature fluctuations or under extreme maximum temperatures is prohibited
- Applications that have not been expressly authorized by Rexroth




3 Safety instructions

The Safety instructions contained in the available application documentation feature specific signal words (DANGER, WARNING, CAUTION or NOTICE) and, where required, a safety alert symbol (in accordance with ANSI Z535.6-2006).

The signal word is meant to draw the reader's attention to the safety instruction and identifies the hazard severity.

The safety alert symbol (a triangle with an exclamation point), which precedes the signal words DANGER, WARNING and CAUTION, is used to alert the reader to personal injury hazards.

The Safety instructions in this documentation are designed as follows:

 DANGER	In case of non-compliance with this safety instruction, death or serious injury will occur.
 WARNING	In case of non-compliance with this safety instruction, death or serious injury could occur.
 CAUTION	In case of non-compliance with this safety instruction, minor or moderate injury could occur.
NOTICE	In case of non-compliance with this safety instruction, property damage could occur.

4 Introduction and overview

4.1 Python runtime environment

Python Runtime App for ctrlX CORE - Basics

The Python Runtime app contains a Python interpreter that has been specifically extended to interact highly efficiently with ctrlX MOTION. Thus, scripts specifying motion commands can be processed and can react to the state of ctrlX MOTION.

Script control via the script manager

A generic script manager is integrated in ctrlX CORE. The script manager can create and control script interpreter instances. A description of the script manager can be found in the ctrlX CORE Runtime application manual documentation, chapter [↗ Script parser/interpreter](#).

To run a Python script in the Python runtime environment, these steps have to be executed:

1. ➤ Copy the script into the solution (see section "Search paths for Python modules")
2. ➤ Creating a Python interpreter instance via script manager
3. ➤ Run script in this instance (see section "Script execution")
(The Python interpreter instance can be used repeatedly and only needs to be created once).

Integrated libraries

Two libraries are integrated into the Python runtime environment:

- motion - Specifying commands and reading out of ctrlX MOTION states, see [↗ Chapter 4.2 Integrated library - motion on page 13](#)
- datalayer - Simple access to the Data Layer, see [↗ Chapter 4.3 Integrated library - datalayer on page 15](#)

In addition, all Python libraries that are integrated in Python by default (e.g. sys) are available.

Troubleshooting

The error handling of the two libraries is realized via Python exceptions. Each time a function call fails, an exception is triggered (typically a RuntimeError). This exception can be recorded and the user can use individual troubleshooting. If the exception is not recorded, the script is canceled and all attached (see motion.attach_obj()) kinematics and axes are stop (with a minor error).

Name arguments

The name arguments in the functions of both libraries can be used on demand. All named arguments can be used as unnamed arguments (the sequence is relevant)

Additional Python libraries

Only Python libraries consisting of pure Python scripts can be used in the Python runtime environment. Libraries that require additional compiled objects are not supported due to the security concept.

If a Python library consisting of pure Python scripts is to be used, it has to be copied to the appropriate search paths (see section "Search paths for Python modules"). This can be done, for example, via "Manage app data", see

➔ <https://docs.automation.boschrexroth.com/doc/878085629/window-manage-app-data/latest/en/>. If a Python library is to be used on a ctrlX CORE, a complete Python runtime environment (including all required compiled objects) can be integrated into a customized app. In this case, the ctrlX MOTION has to be commanded via Data Layer or data has to be queried via Data Layer.

It is recommended to use REST calls for this purpose (e.g. with the Requests library, ➔ <https://requests.readthedocs.io/en/latest/>). The required data (both for commanding and for querying states) are used as JSON objects. The integrated json library ➔ (<https://docs.python.org/3/library/json.html>) facilitates working with this data.

Minimal example for REST calls:

```
import requests
#IP address
ip_addr = "192.168.1.1"
# get bearer token
bearer_addr = "https://" + ip_addr + "/identity-manager/api/v2/auth/token"
command_data =
{"name": "boschrexroth", "password": "boschrexroth"}
res = requests.post(bearer_addr, json=command_data,
verify=False)
token = res.json()["access_token"]

# send command
res = requests.post('https://' + ip_addr + '/automation/api/v2/nodes/motion/axs/' + "Axis1" + '/cmd/pos-abs',
                    json={"type": "object",
                        "value": {"axsPos": 10,
                                "buffered": False,
                                "lim": {"vel": 10,
                                        "acc": 10,
                                        "dec": 10,
                                        "jrkAcc": 0,
                                        "jrkDec": 0}}},
                    headers={ 'Authorization': 'Bearer ' +
                            token },
                    verify=False)

#get CmdId
cmdId = res.json()["value"]
print(cmdId)
```

Search paths for Python modules

The configuration path for the script manager is provided in \$SNAP_COMMON (/var/nap/rexroth-automationcore/common) by the "rexroth-automationcore" app.

Python scripts without relative or absolute path specification are searched in \$SNAP_COMMON/solutions.

Script execution

Script execution

For the "robot" instance, process the \$SNAP_COMMON/solutions/activeSolution/script/loadWorkpiece.py script.

Under the Data Layer node *script/instances/robot/cmd/file*, the payload {"name": "activeSolution/script/loadWorkpiece.py", "param": "pallet1"} is posted.

Optionally, the absolute path specification can be realized using "/var/snap/rexroth-automationcore/common/solutions/activeSolution/script/loadWorkpiece.py".

Imported Python modules are searched in the following order:

Current script directory

- ./

Application modules:

- \$SNAP_COMMON/solutions/activeConfiguration/scripts/user
- \$SNAP_COMMON/solutions/activeConfiguration/scripts/oem
- \$SNAP_COMMON/solutions/activeConfiguration/scripts/bosch

Libraries, if available at the time of instance generation:

- \$SNAP_COMMON/solutions/activeConfiguration/scripts/libraries/user
- \$SNAP_COMMON/solutions/activeConfiguration/scripts/libraries/oem
- \$SNAP_COMMON/solutions/activeConfiguration/scripts/libraries/bosch

4.2 Integrated library - motion

Python functions

- Exceptions are used for troubleshooting
 - Each time a function call fails, exception is triggered (typically a RuntimeError)
 - This exception can be recorded and the user can use individual troubleshooting
 - If the exception is not recorded, the script is canceled and all attached (see motion.attach_obj()) kinematics and axes are stop (with a minor error)
- The named arguments can be used on demand. All named arguments can be used an unnamed arguments (the sequence is relevant)

See documentation ctrlX CORE Runtime Application Manual, chapter [Script parser/interpreter](#).

4.2.1 Python Misc

motion.attach_obj(<objNames>)

Attaches one or several Motion objects to this runner instance.

- <objNames> - strings, Motion object names, comma separated

motion.detach_obj(<objNames>)

Removes one or several Motion objects from this runner instance.

- <objNames> - strings, Motion object names, comma separated

4.2.2 Python states

<ipoValues> = motion.get_axs_ipo_values(<axsName>)

Gets the interpolated values of a single axis.

- <axsName> - string, axis object name
- <ipoValues> - dictionary with 'pos', 'vel', 'acc', 'jrk' with float values

<actValues> = motion.get_axs_act_values(<axsName>)

Gets the actual values of a single axis.

- <axsName> - string, axis object name
- <actValues> - dictionary with 'pos', 'distLeft', 'vel', 'acc', 'torque' with float values (currently, only 'pos' is supported)

<state> = motion.get_axs_plc_open_state(<axsName>)

Gets the PLC Open state of a single axis.

- <axsName> - string, axis object name
- <state> - string, current state of PLC Open

<brakingDistance> = motion.get_axs_braking_distance(<axsName>)

Gets the braking distance of a single axis.

- <axsName> - string, axis object name
- <brakingDistance> - double, current braking distance

<ipoValues> = motion.get_axs_ipo_values(<axsName>)

Gets the interpolated values of a kinematics.

- <kinName> - string, kinematic object name
- <ipoValues> - dictionary with 'pos', 'vel', 'acc', 'jrk' with float values

<ipoData> = motion.get_kin_ipo_add_data(<kinName>)

Gets additional interpolation data of a kinematics of the active command.

- <kinName> - string, kinematic object name
- <ipoData> - dictionary with 'dist_from_start', 'dist_to_target', 'time_from_start', 'time_to_target' with float values

<ipoPos> = motion.get_kin_ipo_pos(kin = <kinName>, [coord_sys_out = <coordSys>])

Gets the interpolated position of the kinematics in a coordinate system.

- <kinName> - string, kinematic object name
- <coordSys> - string, coordinate system ('PCS', 'WCS', 'MCS' or 'ACS'), default is 'PCS'
- <ipoPos> - tuple of the float value with interpolated position in the coordinate system.

<actPos> = motion.get_kin_act_pos(kin = <kinName>, [coord_sys_out = <coordSys>])

Gets the actual position of the kinematics in a coordinate system.

- <kinName> - string, kinematic object name
- <coordSys> - string, coordinate system ('PCS', 'WCS', 'MCS' or 'ACS'), default is 'PCS'
- <ipoPos> - tuple of the float value with actual position in the coordinate system.

<outPos> = motion.coord_transform(kin = <kinName>, pos = <inPos>, coord_sys_in = <coordSysIn>, coord_sys_out = <coordSysOut>)

Transforms a kinematic position from one coordinate system to another coordinate system. The transformation is based on the current kinematic state (e.g. which PCS is active).

- <kinName> - string, kinematic object name
- <inPos> - tuple of float value or float list (max. 16 values), position to be transformed
- <coordSysIn> - string, coordinate system of <inPos> ('PCS', 'WCS', 'MCS' or 'ACS')
- <coordSysOut> - string, coordinate system ('PCS', 'WCS', 'MCS' or 'ACS')
- <outPos> - tuple of float value with transformed position

<state> = motion.get_kin_plc_open_state(<kinName>)

Gets the PLC Open states of a kinematics.

- <kinName> - string, kinematic object name
- <state> - string, current state of PLC Open

<state> = motion.get_cmd_state(obj=<objName>, id=<cmdID>)

Gets the status of a command and of the command ID. Note: when entering an invalid Command ID, the returned state can return any kind of state.

- <objName> - string, Motion object name
- <cmdID> - uint64, Command ID
- <state> - string, current status ('CREATED', 'PREPARED', 'ACTIVE', 'DONE', 'ABORTED', 'OUTDATED')

<cmdID> = motion.get_cmd_act_id(obj=<objName>)

Gets the ID of the active command of the Motion object

- <objName> - string, Motion object name
- <cmdID> - int, ID of the active command of the Motion object

<cmdSrc> = motion.get_cmd_act_src_data(obj=<objName>)

Gets the source information of the active command of the Motion object

- <objName> - string, Motion object name
- <cmdSrc> - dictionary with "description" of the fields (field name, string), 'type' (source type, string) and the 'line' (line number, int) of the active Motion object command.

<Value> = motion.get_override(obj=<objName>)

Current override value of the Motion object.

- <objName> - string, Motion object name
- <Value> - float, current override value of the Motion object

4.3 Integrated library - datalayer

Python functions for Data Layer Access

4.3.1 Function (nodes) = datalayer.browse (<path>)

(nodes) = datalayer.browse (<path>)

The function browses through a specified Data Layer path and returns all nodes of a level below this path.

- <path>
String, Data Layer path to be browsed through
- (nodes)

Tuple of node names

4.3.2 Function **<value> = datalayer.read (<path>)**

`<value> = datalayer.read (<path>)`

The function reads an element of the Data Layer and returns its value. Only simple types are supported.

- `<path>`
String, Data Layer path to be read
- `<value>`
Value node (only simple values are supported)

4.3.3 Function **datalayer.write(<path>, <value>)**

`datalayer.write(<path>, <value>)`

The function writes an element of the Data Layer. Only simple types are supported.

- `<path>`
String, Data Layer path to be written
- `<value>`
Value to be written (only simple values are supported)

4.3.4 Function **datalayer.create(<path>, <value>)**

`datalayer.create(<path>, <value>)`

The function creates an element with the specified value in the Data Layer. Only simple types are supported.

- `<path>`
String, Data Layer path to be created
- `<value>`
Value node (only simple values are supported)

4.3.5 Function **datalayer.remove(<path>)**

`datalayer.remove(<path>)`

The function deletes an element of the Data Layer.

- `<path>`
String, Data Layer path to be deleted

4.3.6 Function **datalayer.read_json(<path>)**

`datalayer.read_json(<path>)`

The function reads an element of the Data Layer and returns its value as JSON string.

- `<path>`
String, Data Layer path to be read
- `<value>`
Value of the node as JSON string

4.3.7 Function **datalayer.write_json(<path>, <value>)**

`datalayer.write_json(<path>, <value>)`

The function writes an element of the Data Layer. It is specified as JSON string.

- `<path>`

String, Data Layer path to be written

- `<value>`

Value to write the JSON string

4.3.8 **Function `datalayer.create_json(<path>, <value>)`**

`datalayer.create_json(<path>, <value>)`

The function creates an element with the specified value in the Data Layer. The value is a JSON string

- `<path>`

String, Data Layer path to be created

- `<value>`

Value of the node JSON string

- `<result>`

Result during the creation as JSON string (only if successful)

4.3.9 **Further information**

Further information

Documentation ctrlX CORE Runtime Application Manual, chapter ➔ [Script parser/interpreter](#)

5 Commands

5.1 Axis commands

➔ **AxsAddToKin** - Add axis to kinematics

Python:

```
<cmdID> = motion.axs_cmd_add_to_kin( axs=<axsName>,  
kin=<kinName> [,buffered=<buffered>] )
```

Generates an AddToKin command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#).

- <axsName> - string, axis object name
- <kinName> - string, kinematic object name
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ **AxsPosRemoveFromKin** - Remove axis from kinematics

Python:

```
<cmdID> = motion.axs_cmd_remove_from_kin( axs=<axsName>)
```

Generates a RemoveFromKin command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <axsName> - string, axis object name

➔ **AxsAddToGantry** - Add axis to gantry

Python:

```
<cmdID> = motion.axs_cmd_add_to_gantry ( slave=<slaveName>,  
master=<masterName> [,buffered=<buffered>] )
```

Generates an AddtoGantry command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#).

- <slaveName> - string, axis object name
- <masterName> - string, gantry master axis object name
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ **AxsRemoveFromGantry** - Remove axis from gantry

Python:

```
<cmdID> =  
motion.axs_cmd_remove_from_gantry( slave=<slaveName> )
```

Generates a RemoveFromGantry command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <slaveName> - string, axis object name

➔ **AxsPosAbs** - Absolute positioning of axis

Python:

```
<cmdID> = motion.axs_cmd_pos_abs( axs=<axsName>,
pos=<targetPos> [,vel=<velocity>] [,acc=<acceleration>]
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]
[,jrk_dec=<jerkDeceleration>] [,buffered=<buffered>] )
```

Generates a posAbs command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#).

- <axsName> - string, axis object name
- <pos> - double, target position of a command
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ **AxsPosRel** - Position axis relatively

Python:

```
<cmdID> = motion.axs_cmd_pos_rel( axs=<axsName>,
pos=<targetPos> [,vel=<velocity>] [,acc=<acceleration>]
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]
[,jrk_dec=<jerkDeceleration>] [,buffered=<buffered>] )
```

Generates a posRel command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <axsName> - string, axis object name
- <pos> - double, target position of a command
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)

- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ AxsPosAdd - Position axis additively

Python:

```
<cmdID> = motion.axs_cmd_pos_add( axs=<axsName>,  
pos=<targetPos> [,vel=<velocity>] [,acc=<acceleration>]  
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]  
[,jrk_dec=<jerkDeceleration>] [,buffered=<buffered>] )
```

Generates a posAdd command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <axsName> - string, axis object name
- <pos> - double, target position of a command
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ AxsVelocity - Axis velocity

Python:

```
<cmdID> = motion.axs_cmd_velocity( axs=<axsName>,  
target_vel=<targetVel> [,vel=<velocity>] [,acc=<acceleration>]  
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]  
[,jrk_dec=<jerkDeceleration>] [,drive_vel_mode=<driveVelMode>]  
[,buffered=<buffered>] )
```

Generates a velocity command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <axsName> - string, axis object name
- <targetVel> - double, target velocity of the command
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)

- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <driveVelMode> - bool, specifies if the command is to switch the drive to the velocity mode (otherwise, the position mode is retained), (presetting is FALSE)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ **AxsJogCont** - AxsJogCont - Jog axis continuously

Python:

```
<cmdID> = motion.axs_cmd_jog_cont( axs=<axsName>,
dir=<direction>, [,vel=<velocity>] [,acc=<acceleration>]
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]
[,jrk_dec=<jerkDeceleration>] )
```

Generates a continuous jogging command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#)

- <axsName> - string, axis object name
- <dir> - string, + or - jogging direction
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

➔ **AxsJogIncr** - Jog axis incrementally

Python:

```
<cmdID> = motion.axs_cmd_jog_incr( axs=<axsName>,
dir=<direction>, incr=<increment> [,vel=<velocity>]
[,acc=<acceleration>] [,dec=<deceleration>]
[,jrk_acc=<jerkAcceleration>] [,jrk_dec=<jerkDeceleration>] )
```

Generates an incremental jogging command for an single axis (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#)).

- <axsName> - string, axis object name
- <dir> - string, + or - jogging direction
- <incr> - double, increment to be moved
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)

➔ **AxsInterrupt** - Interrupt axis command

Python:

```
<cmdID> = motion.axs_cmd_interrupt ( obj=<axsName> )
```

Generates an interrupt command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#)

- <axsName> - string, axis name

➔ **AxsContinue** - Continue axis command

Python:

```
<cmdID> = motion.axs_cmd_continue ( obj=<axsName> )
```

Removes an active Interrupt command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

The command is active immediately, getCmdState() with the specified command ID always returns a DONE.

- <axsName> - string, axis name

➔ **AxsAbort** - Abort axis command

Python:

```
<cmdID> = motion.axs_cmd_abort( axs=<axsName>  
[,dec=<deceleration>] [,jrk_dec=<jerkDeceleration>] )
```

Generates an Abort command for a single axis (object has to be "attached"), see documentation Python Runtime App chapter [Python functions](#).

- <axsName> - string, axis object name
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)

➔ **AxsPower** - Axis power

Python:

```
<cmdID> = motion.axs_cmd_power( axs=<axsName>
[,switch_on=<switchOn>])
```

Generates a power command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter➔[Python functions](#).

- <axsName> - string, axis object name
- <switchOn> - bool, is set to TRUE to switch on power (TRUE is default)

➔ **AxsFixedStop** - Move axis to fixed stop

Python:

Is not yet supported.

➔ **AxsResetFixedStop** - Reset axis fixed stop

Python:

```
<cmdID> = motion.resetfixedstop('axsName', buffered = true)
```

- <axsName> - string, axis object name
- buffered - boolean, command buffered type

➔ **AxsReset** - Reset axis

Python:

```
<cmdID> = motion.axs_cmd_reset( axs=<axsName> )
```

Generates a reset command for an single axis (object has to be "attached"), see documentation Python Runtime App, chapter➔[Python functions](#).

- <axsName> - string, axis name

➔ **AxsSetAbsPos** - Set axis to absolute position

Python:

```
<cmdID> = MotionLib.axs_set_pos_abs( axs=<axsName>,
pos=<NewPosition>, buffered=True )
```

Generates a posAbs command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter➔[Python functions](#).

- <axsName> - string, axis object name
- <NewPosition> - double, new axis position

➔ **AxsHoming** - Referencing command

Python:

```
<cmdID> = motion.axs_cmd_homing(axes=<axsName>, newRefPos)
```

Generates an axis reference command for a single axis (object has to be "attached"), see documentation Python Runtime App chapter [Python functions](#).

- <axsName> - string, axis name
- <newRefPos> - double, new axis position

➔ [AxsCtrlBasedGantryHoming](#) - Control-based referencing of gantry axes

Python:

Is not yet supported.

➔ [AxsCyclicSetPoint](#), ➔ [SetCyclicSetPoint](#) - Cyclic command value

Python:

```
<cmdID> = motion.axs_cyclic_set_point (axes=<axsName>)
```

Generates a cyclic command value command for a single axis (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <axsName> - string, axis name

➔ [AxsProbe](#), ➔ [AxsProbeAbort](#), ➔ [GetAxisProbeValues](#) - Axis probe

Python:

```
<cmdID> = MotionLib.axs_cmd_probe(axes=<axsName>,
trigge_src=<trigge_src>,
    probe_index=< probe_index >,
    measur_sig_sel=<measur_sig_sel >,
    measur_type=< measur_type >,
    eval_edge_type=< measur_type >,
    eval_diff_type=< eval_diff_type >,
    exp_window_edge_enable=<>,
    exp_window_start=< exp_window_start >,
    exp_window_end=< exp_window_end >,
    enable_mark_failure=< enable_mark_failure >,
    max_num_of_mark_failuer=<
max_num_of_mark_failuer >,
    time_comp_enable=< time_comp_enable >,
    time_comp_pos_edge=< time_comp_pos_edge >,
    time_comp_neg_edge=< time_comp_neg_edge >,
    lvl_monitor_active=< lvl_monitor_active >,
    is_auto_activated=< is_auto_activated >)
```

- <axsName> - string, axis name
- <probe_index> - string, the index of the probe, currently available "probe1", "probe2"
- <trigge_src> - string, default is empty
- <measur_sig_sel> - string, type of measurement signal ("encoder1", "encoder2", "finetime")
- <measur_type> - string, type of measurement ("continuous", "singleShot")

- <eval_edge_type> - string, edge type ("disabled", "posEdge", "negEdge", "posNeg")
- <eval_diff_type> - string, edge type ("disabled", "posEdge", "negEdge", "posNeg")
- <exp_window_edge_enable> - bool, the expectation window is enabled
- <exp_window_start> - float, start of the expectation window when the positive edge is evaluated
- <exp_window_end> - float, end of the expectation window when the positive edge is evaluated
- <enable_mark_failure> - bool, enable check for lost marker
- <max_num_of_mark_failuer> - int, threshold value for the registered number of marking errors
- <time_comp_enable> - bool, enable compensation of tester dead time
- <time_comp_pos_edge> - float, dead time compensation for the positive edge, if evaluated
- <time_comp_neg_edge> - float, dead time compensation for the negative edge, if evaluated
- <lvl_monitor_active> - bool, switch-on level monitoring active, works only with drive based probe1
- <is_auto_activated> - bool, button is automatically activated after configuration

Python:

```
<cmdID> =
MotionLib.axs_cmd_probe_abort(axs=<axsName>,trigge_src_name=<trigge_src_name>,probe_index=<probe_index>)
```

- <axsName> - string, axis object name
- <trigge_src_name> - string, default is empty
- <probe_index> - string, the index of the currently available probe "probe1", "probe2"

Python:

```
<Result> = MotionLib.get_axs_probe_values(axs=<axsName>,
probe_index=<probe_index> )
```

- <axsName> - string, axis name
- <probe_index> - string, the index of the probe, currently available "probe1", "probe2"
- <Return> - Returns a directory with these entries:
 - measuredValueNegEdge - negative edge value
 - measuredValueNegEdgeCount - number of negative edges
 - measuredValuePosEdge - positive edge value
 - measuredValuePosEdgeCount - number of positive edges
 - missingMarksEdgeCount - number of missing marks on the edge
 - positionDifference - Position difference
 - positionDifferenceCount - number of position differences
 - Status - Status
 - Valid

➔ **AxsSetIpoPosFromActPos** - Set interpolator position to actual value of the axis

Python:

```
<cmdID> =  
motion.axs_cmd_set_ipo_pos_from_act_pos( axs=<axsName>,  
buffered=<buffered> )
```

Generate a command to set IPO position from the act position command for a single axis (object must be "attached"), see documentation Python Runtime App chapter [Python functions](#).

- <axsName> - string, axis object name
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

5.2 Kinematic commands

→ KinEnable - Enable kinematics

Python:

```
<cmdID> = motion.kin_cmd_enable( kin=<kinName> )
```

Generates an enable kinematic command (object has to be "attached"), see documentation Python Runtime App chapter [Python Functions](#).

- <kinName> - string, kinematic name

→ KinDisable - Disable kinematics

Python:

```
<cmdID> = motion.kin_cmd_disable( kin=<kinName> )
```

Generates a disable kinematic command (object has to be "attached"), see documentation Python Runtime App chapter [Python Functions](#).

- <kinName> - string, kinematic name

→ KinMoveLinAbs - Move kinematics absolutely

Python

```
<cmdID> = motion.kin_cmd_move_lin_abs( kin=<kinName>,  
pos=<targetPos> [,coord_sys=<coordSys>] [,vel=<velocity>]  
[,acc=<acceleration>] [,dec=<deceleration>]  
[,jrkacc=<jerkAcceleration>] [,jrk_dec=<jerkDeceleration>]  
[,buffered=<buffered>] )
```

Generates a MoveLinAbs command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <kinName> - string, kinematic name
- <pos> - tuple of double or list of double (16 values max.), target position of the command
- <coordSys> - string ("PCS", "WCS", "MCS" or "ACS")
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)

- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

→ KinMoveLinRel - Move kinematics relatively

Python:

```
<cmdID> = motion.kin_cmd_move_lin_rel( kin=<kinName>,
pos=<targetPos> [,coord_sys=<coordSys>] [,vel=<velocity>]
[,acc=<acceleration>] [,dec=<deceleration>]
[,jrk_acc=<jerkAcceleration>] [,jrkdec=<jerkDeceleration>]
[,buffered=<buffered>] )
```

Generates a MoveLinAbs command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter → [Python functions](#).

- <kinName> - string, kinematic name
- <pos> - tuple of double or list of double (16 values max.), target position of the command
- <coordSys> - string ("PCS", "WCS", "MCS" or "ACS")
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

→ KinMoveDirectAbs - Move kinematics directly absolute

Python:

```
<cmdID> = motion.kin_cmd_move_direct_abs( kin=<kinName>,
pos=<targetPos> [,coord_sys=<targetCoordinateSystem>]
[,buffered=<buffered>])
```

Generates an MoveDirectAbs command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter → [Python functions](#).

- <kinName> - string, kinematic name
- <pos> - tuple of double or list of double (16 values max.), target position of the command
- <targetCoordinateSystem> - Coordinate system in which <pos> is defined (coordinate system is "PCS", if not programmed)
- <buffered> - boolean, is this command supposed to be a buffered command?

➔ **KinMoveDirectRel** - Move kinematics directly relative

Python:

```
<cmdID> = motion.kin_cmd_move_direct_rel( kin=<kinName>,  
pos=<targetPos> [,coord_sys=<targetCoordinateSystem>]  
[,buffered=<buffered>])
```

Generates an MoveDirectAbs command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#).

- <kinName> - string, kinematic name
- <pos> - tuple of double or list of double (16 values max.), target position of the command
- <targetCoordinateSystem> - Coordinate system in which <pos> is defined (coordinate system is "PCS", if not programmed)
- <buffered> - boolean, is this command supposed to be a buffered command?

➔ **KinMoveDirectAsyncAbs** - Move kinematics directly asynchronously absolutely

Python:

```
<cmdID> = motion.kin_cmd_move_direct_async_abs( kin=<kinName>,  
pos=<targetPos> [,coord_sys=<targetCoordinateSystem>]  
[,vel_factor=<velocityFactor>]  
[,acc_factor=<accelerationFactor>]  
[,dec_factor=<decelerationFactor>]  
[,jrk_acc_factor=<jerkAccelerationFactor>]  
[,jrk_dec_factor=<jerkDecelerationFactor>]  
[,buffered=<buffered>])
```

Generates a MoveDirectAsyncAbs command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter ➔ [Python functions](#).

- <kinName> - string, kinematic name
- <targetPos> - tuple of double or list of double (max. 16 values), target position of command
- <targetCoordinateSystem> - Coordinate system in which <pos> is defined (coordinate system is "PCS", if not programmed)
- <velocityFactor> - double, velocity limit factor of the command (set velocity limit factor for this kinematics to "1", for "0" or not programmed → no limit)
- <accelerationFactor> - double, acceleration limit factor of the command (the last programmed acceleration limit factor is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <decelerationFactor> - double, deceleration limit factor of the command (the last programmed deceleration limit factor is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)

- <jerkAccelerationFactor> - double, factor for the jerk limit of the acceleration of the command (the last programmed factor is used for the jerk limit of the acceleration (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <jerkDecelerationFactor> - double, factor for jerk limit of the deceleration of the command (the last programmed factor for the jerk limit of the deceleration is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

→ **KinMoveDirectAsyncRel** - Move kinematics directly asynchronously relatively

Python:

```
<cmdID> = motion.kin_cmd_move_direct_async_rel( kin=<kinName>,
pos=<targetPos> [, coord_sys=<targetCoordinateSystem>]
[, vel_factor=<velocityFactor>]
[, acc_factor=<accelerationFactor>]
[, dec_factor=<decelerationFactor>]
[, jrk_acc_factor=<jerkAccelerationFactor>]
[, jrk_dec_factor=<jerkDecelerationFactor>]
[, buffered=<buffered>])
```

Generates a MoveDirectAsyncRel command for a kinematics (object has to be "attached"), see documentation Python Runtime App chapter → [Python functions](#).

- <kinName> - string, kinematic name
- <pos> - tuple of double or list of double (16 values max.), target position of the command
- <targetCoordinateSystem> - Coordinate system in which <pos> is defined (coordinate system is "PCS", if not programmed)
- <velocityFactor> - double, velocity limit factor of the command (set velocity limit factor for this kinematics to "1", for "0" or not programmed → no limit)
- <accelerationFactor> - double, acceleration limit factor of the command (the last programmed acceleration limit factor is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <decelerationFactor> - double, deceleration limit factor of the command (the last programmed deceleration limit factor is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <jerkAccelerationFactor> - double, factor for the jerk limit of the acceleration of the command (the last programmed factor is used for the jerk limit of the acceleration (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <jerkDecelerationFactor> - double, factor for jerk limit of the deceleration of the command (the last programmed factor for the jerk limit of the deceleration is used (if this factor is unequal to "0"), if this parameter is not programmed or if the value "0" is specified)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

→ **KinJogCont** - Kinematics continuous jogging

Python:

```
<cmdID> = motion.kin_cmd_jog_cont( kin=<kinName>,  
dir=<direction>, [,vel=<velocity>] [,acc=<acceleration>]  
[,dec=<deceleration>] [,jrk_acc=<jerkAcceleration>]  
[,jrk_dec=<jerkDeceleration>] )
```

Generates a continuous typing command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <kinName> - string, kinematic object name
- <dir> - tuple of double or list of double (16 values max.), jogging direction
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)

➔ KinJogIncr - Incremental kinematic jogging**Python:**

```
<cmdID> = motion.kin_cmd_jog  
incr( kin=<kinName>, dir=<direction>, incr=<increment>  
[,vel=<velocity>] [,acc=<acceleration>] [,dec=<deceleration>]  
[,jrk_acc=<jerkAcceleration>] [,jrk_dec=<jerkDeceleration>] )
```

Generates an incremental jogging command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <kinName> - string, kinematic object name
- <dir> - tuple of double or list of double (16 values max.), jogging direction
- <incr> - double, increment to be moved
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)

↪ KinInterrupt, KinInterrupt2 - Interrupt kinematic command**Python:**

```
<cmdID> = motion.kin_cmd_interrupt( obj=<kinName>, str=<type> )
```

Generates an Interrupt command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [↪ Python functions](#).

- <kinName> - string, kinematic name
- <type> - string, type of braking limit
 - BRAKE_WITH_COMMANDDED_LIMITS (default, if type = none)
 - BRAKE_WITH_AXS_REDUCED_LIMITS
 - BRAKE_WITH_AXS_CFG_LIMITS

↪ KinContinue - Continue kinematic command**Python:**

```
<cmdID> = motion.kin_cmd_continue( obj=<kinName> )
```

Removes an Interrupt command of a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [↪ Python functions](#).

The command is active immediately, getCmdState() with the specified command ID always returns a DONE.

- <kinName> - string, kinematic name

↪ KinAbort, KinAbort2 - Cancel kinematic command**Python:**

```
<cmdID> = motion.kin_cmd_abort( kin=<kinName> , str=<type> )
```

Generates an Abort command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [↪ Python functions](#).

- <kinName> - string, kinematic object name
- <type> - string, type of braking limit
 - BRAKE_WITH_COMMANDDED_LIMITS (default, if type = none)
 - BRAKE_WITH_AXS_REDUCED_LIMITS
 - BRAKE_WITH_AXS_CFG_LIMITS

↪ KinReset - Reset kinematics**Python:**

```
<cmdID> = motion.kin_cmd_reset( kin=<kinName> )
```

Generates a reset command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [↪ Python functions](#).

- <kinName> - string, kinematic name

↪ KinContour - Define contour area**Python:**

```
<cmdID> = motion.kin_cmd_contour( kin=<kinName>
[,is_start=<isStart>] [,prep_cmds=<prepCmds>])
```

Generates a kinematic command to specify the start and end of a contour (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <kinName> - string, kinematic name
- <isStart> - Boolean, is this the start of the contour? (Default: true)
- <prepCmds> - unsigned int, how many commands should be completely prepared? Must be >= 0. Default value is 0.

5.3 Kinematic command options

→ **KinBlend, KinBlendP** - Kinematics option overblending/permanent

Python:

```
motion.kin_cmd_opt_blend( kin=<kinName> , d1=<D1>, d2=<D2> )
```

Generates a blending command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

- <kinName> - string, kinematic object name
- <D1> - blending, length of the first motion command
- <D2> - blending, length of the second motion command

```
motion.kin_cmd_opt_blend_p( kin=<kinName> [,d1=<D1>]  
[,d2=<D2>] )
```

Generates a permanent blending command for a kinematics (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

If d1 and d2 are not programmed, the permanent command option is disabled.

- <kinName> - string, kinematic object name
- <D1> - blending, length of the first motion command
- <D2> - blending, length of the second motion command

→ **KinMCSP** - Command option for kinematics machine coordinate system

Python:

```
motion.kin_cmd_opt_mcs_p( kin=<kinName>, set=<setName>  
[,switch=<ON/OFF>] )
```

Creates a permanent kinematics command option for an MCS (enabling an axis transformation, object has to be "attached"), see documentation Python Runtime App chapter [Python functions](#).

If the optional switch parameter is not programmed, the command enables the option.

- <kinName> - string, kinematic object name
- <SetName> - name of the MCS set that has to be enabled/disabled
- <switch> - string that defines whether the block should be switched "ON" or "Off"

→ **KinPCSP, KinPCSToolP** - Command option for kinematic product coordinate system

Python

```
motion.kin_cmd_opt_pcs_p( kin=<kinName> [, set=<setName>] )
```

Creates a permanent kinematic command option for a PCS that refers to the base (object has to be "attached"), see documentation Python Runtime App chapter [Python Functions](#).

If the optional set parameter is not programmed, the PCS command option (referring to the basis) is disabled.

- <kinName> - string, kinematic object name
- <setName> - string, name of the PCSTool set to be enabled or PCSTool group to be enabled

```
motion.kin_cmd_opt_pcs_tool_p( kin=<kinName> [,
set=<setName>] )
```

Generates a permanent kinematic command option for a PCSTool that refers to the tool (object has to be "attached"), see documentation Python Runtime App chapter [Python Functions](#).

If the optional set parameter is not programmed, the PCSTool command option (referring to the tool).

- <kinName> - string, kinematic object name
- <setName> - string, name of the PCSTool set to be enabled or PCSTool group to be enabled

→ **KinAxsDynLimP** - Lower dynamic limits of kinematic axis by command

Python:

```
motion.kin_cmd_opt_axs_dyn_lim_p( kin=<kinName>, axs=<axsName>
[,vel=<velocity>] [,acc=<acceleration>] [,dec=<deceleration>]
[,jrk_acc=<jerkAcceleration>] [,jrk_dec=<jerkDeceleration>])
```

Enables/disables the permanent command option to reduce the kinematic axis dynamics. The kinematics has to be assigned to the script instance (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#).

If no optional parameters are programmed for the dynamics, the command option is disabled.

- <kinName> - string, kinematic name
- <axsName> - string, kinematic axis name
- <velocity> - double, velocity limit value of the command (the last programmed velocity limit value is used if this parameter is not programmed)
- <acceleration> - double, acceleration limit value of the command (the last programmed acceleration limit value is used if this parameter is not programmed)
- <deceleration> - double, deceleration limit value of the command (the last programmed deceleration limit value is used if this parameter is not programmed)
- <jerkAcceleration> - double, jerk limit value for the acceleration of the command (the last programmed jerk limit value for the acceleration is used if this parameter is not programmed)
- <jerkDeceleration> - double, jerk limit value for the deceleration of the command (the last programmed jerk limit value for the deceleration is used if this parameter is not programmed)
- <buffered> - bool, use TRUE if the command is to be buffered (TRUE by default)

→ **KinContMotionP** - Continuous motion across multiple commands

Python:

```
motion.kin_cmd_opt_cont_motion_p( kin=<kinName> [,switch=<ON/OFF>] )
```

Enables/disables the permanent command option for a continuous motion across several commands. The kinematics has to be assigned to the script instance (attached, see documentation Python Runtime App, chapter [Python functions](#)).

If the optional parameter "switch" is not programmed, the command option is enabled.

- <kinName> - string, kinematic name
- <switch> - string, defines whether the command option is to be enabled ("ON") or disabled ("OFF")

→ KinPolyTransP - Command option for kinematics - Rounding the path with polynomial

Python:

```
motion.kin_cmd_opt_poly_trans_p( kin=<kinName> [,d1=<D1>][,d2=<D2>][,eps=<Eps>])
```

Enables/disables the permanent command option for contour rounding by using polynomial paths to be inserted. The kinematics has to be assigned to the script instance (object has to be "attached"), see documentation Python Runtime App, chapter [Python functions](#)).

If none of the optional parameters is programmed for the path, the command option is disabled.

- <kinName> - string, kinematic object name
- <D1> - double; max. distance before the corner for rounding
- <D2> - double; max. distance behind the corner for rounding
- <Eps> - double; max. distance of the generated polynomial path from the corner

To enable the command option, program either D1 and D2 with a value greater than 0 or Eps.

→ KinSafeAreaP - Work areas/safety zones

Python:

```
motion.kin_cmd_opt_safe_area_p( kin=<kinName>,  
safe_area=<safeArea> [,switch=<ON/OFF>] )
```

Enables/disables the monitoring of a working area or safe zone as permanent command option. The kinematics must be assigned to the script instance (attached, see documentation Python Runtime App, chapter [Python functions](#)).

If the optional parameter "switch" is not programmed, the monitoring of the transferred working area or safe zone is enabled.

- <kinName> - string, kinematic name
- <safeArea> - string, name of the working area or safe zone whose monitoring is to be enabled or disabled
- <switch> - string, defines whether monitoring is to be enabled ("ON") or disabled ("OFF")

[↪ KinFeedGroupP](#) - Specify feed group**Python:**

```
motion.kin_cmd_opt_feed_group_p( kin=<kinName>,
feedGroup=<feedgroup> )
```

Example: `kin_cmd_opt_feed_group_p("Kinematics", "FG_XYZ_O")`

5.4 General commands

[↪ SetErrorLevel](#) - Set error level**Python:**

```
<cmdID> = motion.set_err_level( <objName>, <errLvl> )
```

Sets an error in the motion object (object does not have to be attached).

This command is executed immediately. When querying the command status with the returned command ID (`getCmdState()`), DONE is always returned.

To set the error class to NONE again, generate a reset command.

- <objName> - string, Motion object name
- <errLvl> - string, error class to be set ('LIGHT', 'SEVERE', 'CRITICAL')

[↪ SetOverride](#) - Override**Python**

```
<cmdID> = motion.set_override( obj=<objName>, val=<Value> )
```

Provides the current override value of the Motion object

- <objName> - string, Motion object name
- <Value> - the current override value of the Motion object

[↪ SetAccOverride, GetAccOverride](#) - Acceleration override**Python:**

```
<cmdID> = motion.set_acc_override( obj=<objName>, val=<Value> )
```

Sets the acceleration override for the specified axis/kinematics. The axis/kinematics has to be assigned to the script instance (attached, object has to be "attached"), see documentation Python Runtime App, chapter [↪ Python functions](#)).

The command is active immediately, `getCmdState()` with the received command ID always returns a DONE.

The value of the override should be in the interval [0.01; 1]. A value of 0.01 indicates that an acceleration of 1 % of the commanded acceleration must not be exceeded. A value of 1 indicates 100% override so that the commanded acceleration can be used.

- <objName> - string, axis/kinematic name
- <Value> - number, new value for the acceleration override

Python:

```
<Value> = motion.get_acc_override( obj=<objName> )
```

Returns the current acceleration override of the axis or the kinematics.

- <objName> - string, axis/kinematic name
- <Value> - the current acceleration override value of the axis or the kinematics

➔ WaitPrepare - Stop preparation

Python:

```
<cmdID> = MotionLib.wait_prepare ( obj=<objName>,  
buffered=True )
```

Creates a waitPrepare command for a motion object (object has to be "attached"), see documentation Python Runtime App chapter ➔ [Python functions](#).

- <objName> - string, Motion object name

➔ JoinCmd, SetSignalCmd, SetSignalCmdOpt, ResetSignalCmd, WaitForSignalCmd, SetSignal, ResetSignal, GetSignal - Synchronizing motion objects

Python:

```
<cmdId> = motion.join(<objName>, group=<groupObjNames>,  
sync_start_next_cmd=<syncStartNextCmd>)
```

Python:

Commands:

```
<cmdId> = motion.axs_cmd_set_signal(<objName>,  
signal_id=<signalId>)
```

```
<cmdId> = motion.kin_cmd_set_signal(<objName>,  
signal_id=<signalId>)
```

```
<cmdId> = motion.axs_cmd_reset_signal(<objName>,  
signal_id=<signalId>)
```

```
<cmdId> = motion.kin_cmd_reset_signal(<objName>,  
signal_id=<signalId>)
```

```
<cmdId> = motion.axs_cmd_wait_for_signal(<objName>,  
signal_id=<signalId>,
```

```
auto_reset=<autoReset>)
```

```
<cmdId> = motion.kin_cmd_wait_for_signal(<objName>,  
signal_id=<signalId>,
```

```
auto_reset=<autoReset>)
```

Command options:

```
<cmdId> = motion.axs_cmd_opt_set_signal(<objName>,  
perm_type=<permType>,  
signal_id=<signalId>)
```

```
<cmdId> = motion.kin_cmd_opt_set_signal(<objName>,  
perm_type=<permType>,  
signal_id=<signalId>)
```

Where

<permType> - Permanent type of the command option
"ONCE" - Execute this command option only once

System interface functions

Data Layer:

```
set_signal(signal_id):
    datalayer.write(f'motion/somo/signals/{signal_id}', True)

reset_signal(signal_id):
    datalayer.write(f'motion/somo/signals/{signal_id}', False)

get_signal(signal_Id):
    return datalayer.read(f'motion/somo/signals/{signal_id}')
```

5.5 Additional information

↪ **GetOverride** - Read out override

Python:

```
<Value> = motion.get_override( obj=<objName> )
```

Provides the current override value of the Motion object

- <objName> - string, Motion object name
- <Value> - the current override value of the Motion object

↪ **GetCmdState** - Command states

Python:

```
motion.get_cmd_state(obj: str, id: int) -> str:
    """Read the state of a given command (command id has to be
    given) as a string.
    You get the command id as a return value, when you create
    a command.
    Note, that the result is undefined, when you call the
    function with a invalid command id.

    Arguments:
        obj -- The name of the motion object (axis or kinematics).
        id  -- The command id of the command, that state is
        requested.

    Returns:
        str -- The state of the command as string
    """
```

↪ **AxsGetBrakingDistance, AxsGetBrakingDistanceEx** - Determine current braking distance

Python:

```
<brakingDistance> =
motion.get_axs_braking_distance( <axsName> )

def get_axs_braking_distance_ex(axs: str, selected_types:
Tuple[str] = ('SOFT_STOP', 'ESTOP')) -> Dict:
```

Determines the braking distance of a single axis.

- <axsName> - string, axis object name
- <brakingDistance> - double, current braking distance

Read the braking distance of the specified axis with the selected brake type.

Arguments:

- axs -- axis name
- selected_types -- the selected brake types ("SOFT_STOP", "ESTOP") that will be taken into account in the calculation

Return:

- Dict -- data type: dictionary with the fields "braking_distance" (calculated braking distance, float), "distance_type" (type of calculated braking distance, string ("SOFT_STOP" or "ESTOP")).

➔ **AxsGetIpoAddData** - Additional axis data

Python:

```
<result> = motion.get_axs_ipo_add_data( <axsName> )
```

Return a directory with these entries:

- dist_from_start - distance from start position
- dist_to_target - distance to target position
- time_from_start - lapsed time from the start position
- time_to_target - estimated time to target position (assuming an override)
- <axsName> - string, axis name

➔ **KinGetIpoAddData** - Additional kinematic data

Python:

```
<result> = motion.get_kin_ipo_add_data( <kinName> )
```

Return a directory with these entries:

- dist_from_start - distance from start position (on the distance)
- dist_to_target - distance to target position (on the distance)
- time_from_start - lapsed time from the start position
- time_to_target - estimated time to target position by taking the current override into consideration
- <kinName> - string, kinematic object name

6 Related documentation

6.1 Overview

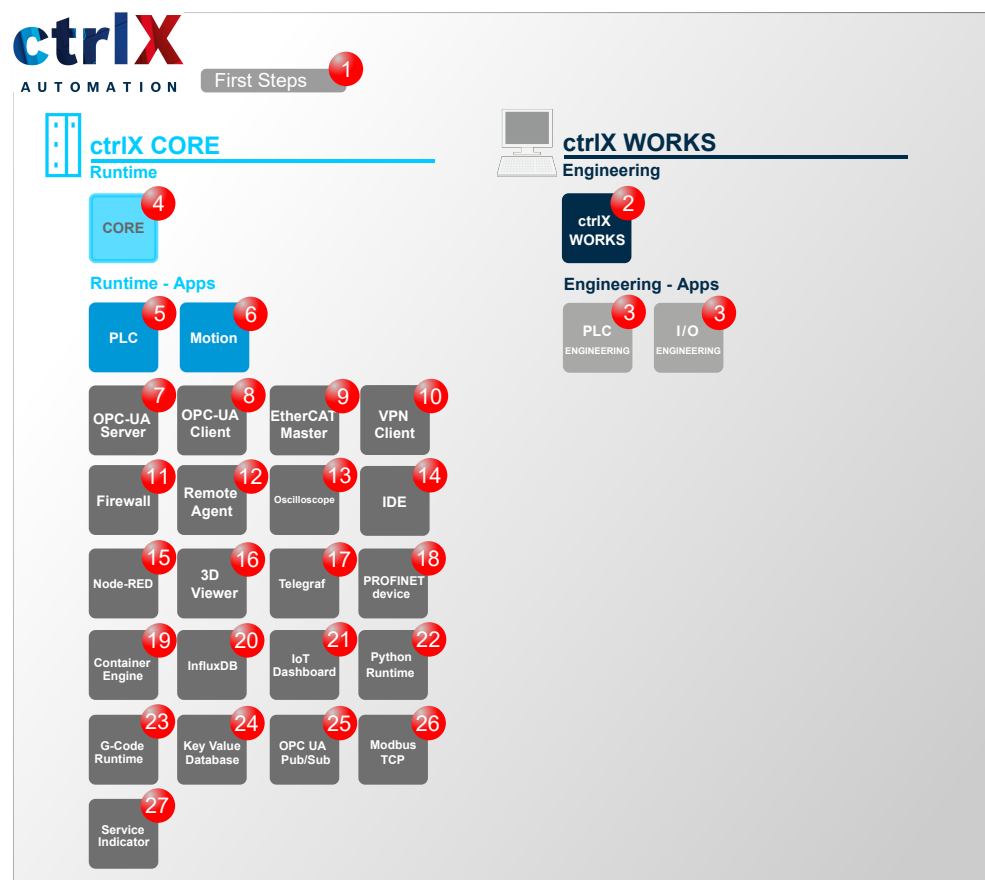


Fig. 1: Overview on further documentations

6.2 ctrlX AUTOMATION

No.	Documentation
1	ctrlX WORKS First Steps 01VRS Quick Start Guide ↪ Web documentation link Ordering information: <ul style="list-style-type: none">• DOK-XWORKS-F*STEP**V01-QURS-EN-P• R911403760

6.3 ctrlX WORKS

No.	Documentation
2	ctrlX WORKS Basic System 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XWORKS-*****V01-APRS-EN-P • R911403761
3	ctrlX PLC Engineering - PLC Programming System 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XPLC**-ENG*****V01-APRS-EN-P • R911403764
3	ctrlX PLC Engineering - PLC Libraries 01VRS Reference ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XPLC**-LIBRARY*V01-RERS-EN-P • R911403766

6.4 ctrlX CORE

No.	Documentation
4	ctrlX CORE - Runtime 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-BASE****V01-APRS-EN-P • R911403768
	ctrlX CORE - Nodes of the Data Layer 01VRS Reference ↗ Web documentation link Bestellinformationen: <ul style="list-style-type: none"> • DOK-XCORE*-BASE*DL*V01-RERS-EN-P • R911420072
	ctrlX CORE - Diagnostics 01VRS Reference ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-DIAG****V01-RERS-EN-P • R911403770

6.5 ctrlX CORE Apps

No.	Documentation
5	PLC App - PLC Runtime Environment for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-PLC*****V01-APRS-EN-P • R911403787
6	Motion App - Motion Runtime Environment for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-MOTION**V01-APRS-EN-P • R911403791
7	OPC UA Server App - OPC UA Server for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-OPCSERV*V01-APRS-EN-P • R911403778
8	OPC UA Client App - OPC UA Client for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-OPCCLIENV01-APRS-EN-P • R911403781
9	EtherCAT Master App - EtherCAT Master for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-ETHERCATV01-APRS-EN-P • R911403773
10	VPN Client App - Remote Support Software for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-VPN*****V01-APRS-EN-P • R911403775
11	Firewall App - Security Functions for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-FIREWALLV01-APRS-EN-P • R911403783

No.	Documentation
12	Remote Agent App - ctrlX Device Portal Connection for ctrlX Devices 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-REMOTE**V01-APRS-EN-P • R911403785
13	Oscilloscope App - Oscilloscope Function for ctrlX Devices 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-OSCI****V01-APRS-EN-P • R911409806
14	IDE App - Integrated Development Environment 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-IDE****V01-APRS-EN-P • R911410625
15	Node RED App - Graphic Programming for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-NODERED*V01-APRS-EN-P • R911403789
16	3D Viewer App - Browser-based 3D Kinematic Simulation for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-3D*VIEW*V01-APRS-EN-P • R911416124
17	Telegraf App - Server Agent for Collecting Data in the Data Layer 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-TELEGRAFV01-APRS-EN-P • R911416836
18	PROFINET Device App - PROFINET Device for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-PROFINETV01-APRS-EN-P • R911417857

No.	Documentation
19	Container Engine App - Use of Docker® Images on ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-DOCKER**V01-APRS-EN-P • R911417855
20	InfluxDB App - Influx Database Connection for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-INFLUX**V01-APRS-EN-P • R911418738
21	IoT Dashboard App - Data Visualization in Dynamic, Interactive Dashboards 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-GDB*****V01-APRS-EN-P • R911420426
22	Python Runtime App - Python Runtime Environment for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-PYR*****V01-APRS-EN-P • R911420430
23	G-Code Runtime App - G-Code Interpreter for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-GCO*****V01-APRS-EN-P • R911420428
24	Key Value Database App - Managing Data in the Data Layer 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-KVD*****V01-APRS-EN-P • R911420423
25	OPC UA Pub/Sub App - OPC UA Pub/Sub for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none"> • DOK-XCORE*-PUBSUB**V01-APRS-EN-P • R911418736

No.	Documentation
26	Modbus TCP App - Modbus TCP Communication for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none">• DOK-XCORE*-MOD*TCP*V01-APRS-EN-P• R911417926
27	Service Indicator App - Service Indicator for ctrlX CORE 01VRS Application Manual ↗ Web documentation link Ordering information: <ul style="list-style-type: none">• DOK-XCORE*-SIN*****V01-APRS-EN-P• R911418740

7 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts provide you with advice and assistance. You can contact us **24/7**.

Service Germany

Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone: **+49 9352 40 5060**

Fax: **+49 9352 18 4941**

Email: ➔ service.svc@boschrexroth.de

Internet: ➔ <http://www.boschrexroth.com>

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide

Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information

To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

8 Index

1, 2, 3 ...

(nodes) = datalayer.browse (<path>). 15

(nodes) = datalayer.read (<path>). 16

A

Achskommandos

AxsJogCont - AxsJogCont - Jog axis
continuously. 22

Additional information. 38

AxsGetBrakingDistance,
AxsGetBrakingDistanceEx - Determine
current braking distance. 38

AxsGetIpoAddData - Additional axis data. . . 39

GetCmdState - Command states. 38

GetOverride - Read out override. 38

KinGetIpoAddData - Additional kinematic
data. 39

Additional Python libraries. 11

Axis commands. 19

AxsAbort - Abort axis command. 23

AxsAddToGantry - Add axis to gantry. 19

AxsAddToKin - Add axis to kinematics. . . . 19

AxsContinue - Continue axis command. . . . 23

AxsCtrlBasedGantryHoming - Control-based
referencing of gantry axes. 25

AxsFixedStop - Move axis to fixed stop. . . 24

AxsHoming - Referencing command. 24

AxsInterrupt - Interrupt axis command. . . . 23

AxsJogIncr - Jog axis incrementally. 22

AxsPosAbs - Absolute positioning of axis. . . 20

AxsPosAdd - Position axis additively. 21

AxsPosRel - Position axis relatively. 20

AxsPosRemoveFromKin - Remove axis from
kinematics. 19

AxsPower - Axis power. 24

AxsProbe, AxsProbeAbort,
GetAxisProbeValues - Axis probe. 25

AxsRemoveFromGantry - Remove axis from
gantry. 19

AxsReset - Reset axis. 24

AxsSetAbsPos - Set axis to absolute position
. 24

AxsSetIpoPosFromActPos - Set interpolator
position to actual value of the axis. 26

AxsVelocity - Axis velocity. 21

B

Basics. 11

C

ctrlX AUTOMATION

Related documentation. 41

D

datalayer.browse. 15

datalayer.create. 16

datalayer.create_json. 17

datalayer.create_json(<path>, <value>). . . . 17

datalayer.create(<path>, <value>). 16

datalayer.read. 16

datalayer.read_json. 16

datalayer.read_json(<path>). 16

datalayer.remove. 16

datalayer.remove(<path>). 16

datalayer.write. 16

datalayer.write_json. 16

datalayer.write_json(<path>, <value>). 16

datalayer.write(<path>, <value>). 16

G

General commands. 36

JoinCmd, SetSignalCmd, SetSignalCmdOpt,
ResetSignalCmd, WaitForSignalCmd,
SetSignal, ResetSignal, GetSignal -
Synchronizing motion objects. 37

SetAccOverride, GetAccOverride -

Acceleration override. 36

SetErrorLevel - Set error level. 36

SetOverride - Override. 36

WaitPrepare - Stop preparation. 37

H

Helpdesk. 47

Hotline. 47

I

Integrated libraries. 11

Integrated library - datalayer. 15

Integrated Library - motion. 13

Intended use

Areas of application. 7

Areas of use. 7

Introduction. 7

K

Kinematic command options. 33

KinBlend, KinBlendP - kinematic option
overblending/permanent. 33

KinContMotionP - Continuous motion across
multiple commands. 34

KinFeedGroupP - Specify feed group. 36

KinMCSP - Command option for kinematic
machine coordinate system. 33

KinSafeAreaP - Work areas/safety zones. . . 35

Kinematic commands. 27

KinContinue - Continue kinematic command
. 32

KinContour - Define contour area. 32

KinDisable - Disable kinematics. 27

KinEnable - Enable kinematics. 27

KinInterrupt, KinInterrupt2 - Interrupt
kinematic command. 32

Kinematics command options

KinAxsDynLimP - Lower dynamic limits of the kinematic axis by command.	34
KinPCSP, KinPCSToolP - Command option for kinematics product coordinate system. . . .	33
KinPolyTransP - Command option for kinematics - Rounding the path with polynomial.	35

Kinematics commands

KinAbort, KinAbort2 - Cancel kinematic command.	32
KinJogCont - Kinematics continuous jogging	30
KinJogIncr - Incremental kinematic jogging	31
KinMoveDirectAbs - Move kinematics directly absolute.	28
KinMoveDirectAsyncAbs - Move kinematics directly asynchronously absolutely.	29
KinMoveDirectAsyncRel - Move kinematics directly asynchronously relatively.	30
KinMoveDirectRel - Move Kinematics directly relative.	29
KinMoveLinAbs - Move kinematics absolutely	27
KinMoveLinRel - Move kinematics relatively	28
KinReset - Reset kinematics.	32

N

Name arguments.	11
-------------------------	----

P

Python functions.	13
Python functions for Data Layer Access. . . .	15
Python Misc.	13
Python runtime environment.	11
Python states.	13

S

Safety instructions.	9
Script control via the script manager.	11
Script execution.	13
Search paths for Python modules.	12
Service hotline.	47
Support.	47

T

Troubleshooting.	11
--------------------------	----

U

Unintended use.	8
Consequences, disclaimer.	7

Bosch Rexroth AG
Bgm.-Dr.-Nebel-Str. 2
97816 Lohr a.Main
Germany
Tel. +49 9352 18 0
Fax +49 9352 18 8400
www.boschrexroth.com/electrics



R911420430