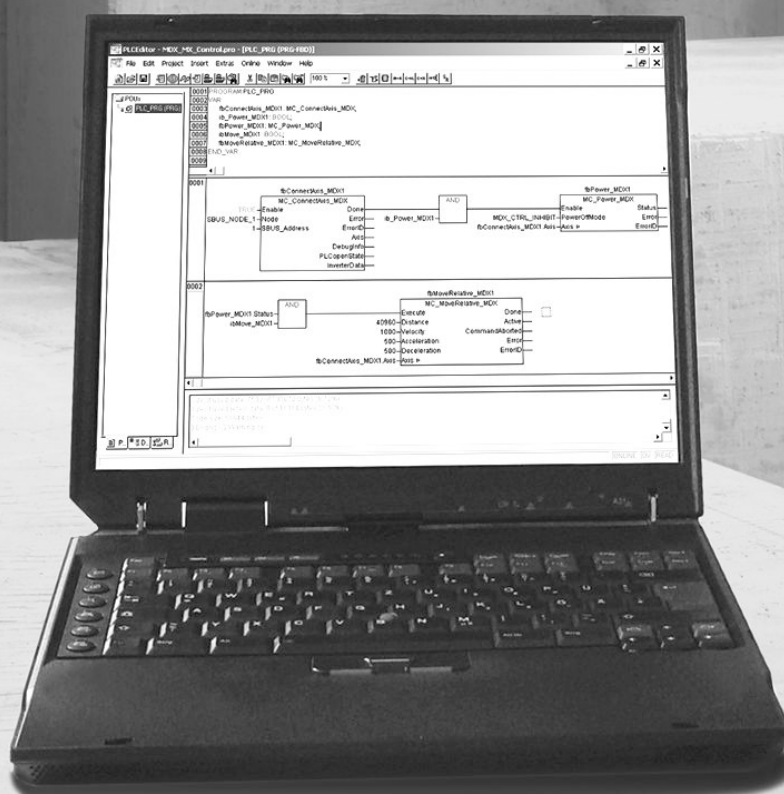


# Manual



# MultiMotion / MultiMotion Light Technology Module Kinematics



## Contents

<b>1</b>	<b>General information .....</b>	<b>7</b>
1.1	About this documentation .....	7
1.2	Design of the safety notes .....	7
1.2.1	Meaning of signal words .....	7
1.2.2	Structure of section-related safety notes .....	7
1.2.3	Structure of embedded safety notes .....	8
1.3	Right to claim under warranty .....	8
1.4	Exclusion of liability .....	8
1.5	Applicable documentation .....	9
1.6	Product names and trademarks .....	9
1.7	Copyright .....	9
<b>2</b>	<b>Safety notes .....</b>	<b>10</b>
2.1	General .....	10
2.2	Target group .....	10
2.3	Designated use .....	11
2.4	Bus systems .....	11
<b>3</b>	<b>System description .....</b>	<b>12</b>
3.1	Content of this manual .....	12
3.1.1	Writing conventions.....	13
3.2	Requirements .....	13
3.2.1	Hardware .....	13
3.2.2	Software.....	13
3.2.3	Required technology level.....	13
3.3	Functions .....	15
3.4	Areas of application .....	17
<b>4</b>	<b>Basics of robotics .....</b>	<b>18</b>
4.1	Continuous-path control .....	18
4.2	Types of kinematic transformation .....	18
4.3	Examples of further transformations .....	18
4.4	Cartesian position .....	19
<b>5</b>	<b>Selecting Kinematics or HandlingKinematics .....</b>	<b>21</b>
5.1	Level 1: HandlingKinematics application module for CCU .....	21
5.1.1	Characteristics .....	21
5.1.2	Advantages.....	21
5.2	Level 2: HandlingKinematics technology module for MOVI-PLC® .....	22
5.2.1	Characteristics .....	22
5.2.2	Additional advantages.....	22
5.3	Level 3: Kinematics technology module for MOVI-PLC® .....	23
5.3.1	Characteristics .....	23
5.3.2	Additional advantages.....	23
<b>6</b>	<b>Startup and configuration.....</b>	<b>25</b>
6.1	Procedure .....	25
6.2	Requirements .....	26

6.3	Step 1: Create a MotionStudio project .....	26
6.4	Step 2: Create a MOVI-PLC® project .....	27
6.5	Step 3: Integrate the technology module .....	28
6.6	Step 4: Start up the single axes .....	30
6.7	Step 5: Configure the kinematic model .....	30
6.7.1	Adding a new technology module .....	30
6.7.2	Selecting "Kinematics" or "HandlingKinematics" .....	32
6.7.3	Selecting device type and kinematic model .....	33
6.7.4	Configuring axes .....	34
6.7.5	Adapting model axes to real axes .....	35
6.7.6	Setting kinematic parameters .....	36
6.7.7	Setting kinematic software limit switches .....	36
6.7.8	Cartesian settings .....	37
6.7.9	Continuous path settings .....	38
6.7.10	Other settings .....	39
6.7.11	Default assignment of input variables .....	39
6.7.12	Download .....	40
<b>7</b>	<b>Diagnostics .....</b>	<b>41</b>
7.1	Kinematic monitor .....	41
7.1.1	User interface .....	41
7.1.2	Monitor and control mode .....	43
7.1.3	Setting for kinematic mode .....	45
7.1.4	Referencing the axes .....	46
7.1.5	Axis-wise jog mode .....	46
7.1.6	Cartesian jog mode .....	47
7.1.7	Axis-wise TARGET mode .....	48
7.1.8	Cartesian TARGET mode .....	49
7.1.9	ContinuousPath mode .....	49
7.1.10	Selecting the transformation .....	50
7.1.11	Various settings .....	52
7.2	3D simulation .....	53
7.2.1	Purpose .....	53
7.2.2	Requirements .....	53
7.2.3	Starting 3D simulation .....	54
7.2.4	Adjusting the 3D simulation .....	55
7.3	MessageHandler .....	55
<b>8</b>	<b>Kinematic models .....</b>	<b>57</b>
8.1	CARTESIAN GANTRY .....	57
8.2	ROLLER GANTRY .....	60
8.3	SCARA .....	62
8.4	DELTA .....	64
8.5	TRIPOD .....	65
8.6	MIXED .....	66
8.7	USER kinematic model .....	66
<b>9</b>	<b>Interpolating operating modes .....</b>	<b>67</b>



9.1	Overview .....	67
9.2	Jog mode .....	68
9.2.1	Specifying direction and kinematic quantities .....	68
9.3	TARGET modes .....	68
9.3.1	Principle .....	68
9.3.2	Example: Pick-and-place .....	69
9.3.3	Advantages .....	70
9.3.4	KIN_TARGET_AXIS .....	71
9.3.5	KIN_TARGET_CART .....	72
9.3.6	TARGET blending .....	73
9.4	Continuous path operating modes .....	74
9.4.1	Principle .....	74
9.4.2	KIN_LIN_XY / _YZ / _ZX .....	75
9.4.3	KIN_LIN_3D .....	75
9.4.4	KIN_CIRC_XY / _YZ / _ZX .....	75
9.4.5	Continuous path blending .....	76
9.5	Comparison of continuous path blending and TARGET blending .....	78
9.6	Combining continuous path modes and TARGET modes .....	79
<b>10</b>	<b>MOVI-PLC® program .....</b>	<b>81</b>
10.1	Task configuration .....	81
10.2	User program .....	82
10.3	Sample programs .....	83
10.3.1	TARGET programming .....	85
10.3.2	Continuous path programming .....	87
<b>11</b>	<b>AxisGroupKin.Inst[..] variable interface .....</b>	<b>89</b>
11.1	In.General .....	90
11.2	In.Homing .....	92
11.3	In.Transform .....	92
11.4	In.MasterPosition .....	93
11.5	In.Jog .....	94
11.6	In.Target .....	95
11.7	In.Cp .....	97
11.7.1	In.Cp.Settings .....	98
11.7.2	In.Cp.BackToPath .....	99
11.7.3	In.Cp.Segment .....	99
11.7.4	In.Cp.Path .....	103
11.8	In.Simu3D .....	103
11.9	In.Diag .....	105
11.10	Out.General .....	106
11.11	Out.Homing .....	111
11.12	Out.Cp .....	111
11.13	Out.Simu3D .....	113
11.14	Out.Diag .....	113
<b>12</b>	<b>Functions and function blocks of the MPLCKinematics library .....</b>	<b>114</b>
12.1	MC_KinInProximity function .....	114

12.2	MC_KinCoordSysMeasurement function block .....	115
12.3	MC_KinEncoderDataProcessing function block .....	117
12.4	MC_KinWcsPcs1Assignment function block .....	119
12.5	MC_KinWcsPcs2Assignment function block .....	121
12.6	MC_KinProfGen function block .....	121
<b>13</b>	<b>Applications .....</b>	<b>124</b>
13.1	Referencing a roller gantry .....	124
13.2	Open parallel kinematics .....	124
13.3	Changing the coordinate system .....	124
13.4	Calibrating a coordinate system .....	127
13.5	Program structure for tracking applications with PCS1/2 .....	129
13.6	Program structure for tracking applications with more than 2 PCS .....	130
13.7	Using a tool transformation .....	132
13.8	Combination of cam disk and kinematics function .....	132
13.9	CP interpolation as slave of an external master profile .....	133
13.10	Continuous change of the kinematics parameters .....	133
13.11	Kinematics with more than 6 drives .....	134
13.12	Integrating user kinematics .....	135
13.13	Dealing with ambiguities of the ABC orientation values .....	135
13.13.1	One-to-one assignment .....	135
13.13.2	No one-to-one assignment.....	136
13.13.3	Applying saved orientation values .....	136
13.14	Storing resolver position values .....	137
<b>14</b>	<b>Troubleshooting .....</b>	<b>138</b>
14.1	3D simulation .....	138
14.1.1	Unable to establish connection .....	138
14.1.2	No model is displayed .....	139
14.2	Load precontrol .....	140
14.2.1	Sagging axes .....	140
14.3	Setting inverter parameters .....	141
<b>15</b>	<b>Error codes .....</b>	<b>144</b>
15.1	General errors .....	144
15.2	Configuration errors .....	145
15.3	General parameter errors .....	147
15.4	Target parameter errors .....	148
15.5	ContinuousPath parameter errors .....	149
15.6	Profile generator errors .....	152
15.7	3D simulation errors .....	153
15.8	AxisGroupControl kinematic errors .....	154
<b>16</b>	<b>Glossary and list of abbreviations.....</b>	<b>156</b>
16.1	Glossary .....	156
16.2	List of abbreviations .....	162
	<b>Index .....</b>	<b>163</b>

## 1 General information

### 1.1 About this documentation

The documentation is part of the product and contains important information. The documentation is for everyone who works with this product.

The documentation must be accessible and legible. Make sure that persons responsible for the system and its operation as well as persons who work independently with the software and the connected units of SEW-EURODRIVE have read through the manual carefully and understood it. If you are unclear about any of the information in this documentation or if you require further information, please contact SEW-EURODRIVE.

### 1.2 Design of the safety notes

#### 1.2.1 Meaning of signal words

The following table shows the grading and meaning of the signal words for safety notes.

Signal word	Meaning	Consequences if disregarded
<b>▲ DANGER</b>	Imminent hazard	Severe or fatal injuries
<b>▲ WARNING</b>	Possible dangerous situation	Severe or fatal injuries
<b>▲ CAUTION</b>	Possible dangerous situation	Minor injuries
<b>NOTICE</b>	Possible damage to property	Damage to the drive system or its environment
<b>INFORMATION</b>	Useful information or tip: Simplifies handling of the drive system.	

#### 1.2.2 Structure of section-related safety notes

Section-related safety notes do not apply to a specific action but to several actions pertaining to one subject. The hazard symbols used either indicate a general hazard or a specific hazard.

This is the formal structure of a safety note for a specific section:



#### SIGNAL WORD

Type and source of hazard.






Possible consequence(s) if disregarded.

- Measure(s) to prevent hazard.

#### Meaning of the hazard symbols

The hazard symbols in the safety notes have the following meaning:

Hazard symbol	Meaning
	General hazard

Hazard symbol	Meaning
	Warning of dangerous electrical voltage
	Warning of hot surfaces
	Warning of risk of crushing
	Warning of suspended load
	Warning of automatic restart

### 1.2.3 Structure of embedded safety notes

Embedded safety notes are directly integrated into the instructions just before the description of the dangerous action.

This is the formal structure of an embedded safety note:

- **▲ SIGNAL WORD** Type and source of hazard.  
Possible consequence(s) if disregarded.  
– Measure(s) to prevent hazard.

## 1.3 Right to claim under warranty

A requirement of fault-free operation and fulfillment of any rights to claim under limited warranty is that you adhere to the information in the documentation at hand. Therefore, read the documentation before you start working with the software and the connected units from SEW-EURODRIVE.

Make sure that the documentation is available to persons responsible for the machinery and its operation as well as to persons who work independently on the units. Also ensure that the documentation is legible.

## 1.4 Exclusion of liability

Please observe this documentation as well as the documentation for the software used and the SEW-EURODRIVE devices connected. This documentation must be observed to ensure that the devices operate safely and that the specified product properties and performance characteristics are achieved.

SEW-EURODRIVE assumes no liability for injury to persons or damage to equipment or property resulting from non-observance of the documentation. In such cases, SEW-EURODRIVE assumes no liability for defects.

## **1.5 Applicable documentation**

Observe the following applicable documentation:

- "Multi-Axis Servo Inverter MOVIAXIS® MX" operating instructions
- "MOVIDRIVE® MDX Drive Inverter" operating instructions
- "MOVITRAC® MC07 Frequency Inverter" operating instructions
- "DH.21B (standard) / DH.41B (advanced) Controller" manual
- "UHX71B (power) Controller" manual
- "MultiMotion Program Module, Universal, Parameterizable Software Platform for MOVI-PLC®" manual
- "MOVI-PLC® Programming in the PLC Editor" system manual
- "HandlingKinematics Technology Module for MultiMotion/MultiMotion Light" manual

Always use the latest version of the documentation and software.

The SEW-EURODRIVE homepage ([www.sew-eurodrive.com](http://www.sew-eurodrive.com)) provides a wide selection of documentation in various languages available for download.

If required, you can order printed and bound copies of the documentation from SEW-EURODRIVE.

## **1.6 Product names and trademarks**

The brands and product names in this documentation are trademarks or registered trademarks of their respective titleholders.

## **1.7 Copyright**

© 2014 SEW-EURODRIVE. All rights reserved.

Unauthorized reproduction, modification, distribution or any other use of the whole or any part of this documentation is strictly prohibited.

## 2 Safety notes

### 2.1 General

The following basic safety notes are intended to prevent injury to persons and damage to property. The user must ensure that the basic safety notes are read and observed.

Ensure that persons responsible for the machinery and its operation as well as persons who work independently have read through the documentation carefully and understood it. If you are unclear about any of the information in this documentation or if you require further information, please contact SEW-EURODRIVE.

The following safety notes refer to the use of the software. Also observe the supplementary safety notes in this documentation and in the documentation for the connected units from SEW-EURODRIVE.

This document does not replace the detailed documentation for the connected units. This documentation assumes that the user has access to and is familiar with the documentation for all connected units from SEW-EURODRIVE.

Never install or operate damaged products. Report any damage to the shipping company immediately.

Depending on the degree of protection, units may have live, uninsulated, and sometimes moving or rotating parts, as well as hot surfaces during operation.

Removing required covers without authorization, improper use or incorrect installation and operation may result in severe injury to persons, or damage to machinery. Consult the documentation for further information.

### 2.2 Target group

Work with the software in this solution may only be performed by adequately qualified personnel. Qualified personnel in this context are persons who have the following qualifications:

- Appropriate training in their relevant field.
- Knowledge of this documentation and other applicable documentation.
- SEW-EURODRIVE recommends additional product training for products that are operated using this software.

All mechanical work on connected units is to be performed exclusively by adequately qualified personnel. Qualified personnel in the context of this documentation are persons familiar with the design, mechanical installation, troubleshooting and servicing of the product, who possess the following qualifications:

- Training in mechanical engineering, e.g. as a mechanic or mechatronics technician (final examinations must have been passed).
- Knowledge of this documentation and other applicable documentation.

All electrical work on connected units is to be performed exclusively by adequately qualified electricians. Qualified electricians in the context of this documentation are persons familiar with electrical installation, startup, troubleshooting and servicing of the product, who possess the following qualifications:

- Training in electrical engineering, e.g. as an electrician or mechatronics technician (final examinations must have been passed).
- Knowledge of this documentation and other applicable documentation.



- Knowledge of the relevant safety regulations and laws.
- Knowledge of all other standards, directives and laws named in this documentation.

The above-mentioned persons must have the express authorization of the company to operate, program, configure, label and ground units, systems and circuits in accordance with the standards of safety technology.

All work in the areas of transportation, storage, operation and waste disposal must be carried out by persons who are trained appropriately.

## 2.3 Designated use

The MultiMotion program module with the "Kinematics" technology module is a universal, parameterizable software platform for the MOVI-PLC® advanced and power controllers from SEW-EURODRIVE.

The program module comprises the following components:

- **"AxisControl\_MultiMotion.pro" project template**

The user creates a MOVI-PLC® project from the project template. He or she then imports the "Kinematics" technology module and the matching task configuration. Users are now able to access the kinematics functions in the MOVI-PLC® program.

- **MultiMotion Editor**

This tool is used for setting the parameters of the functions in the MOVI-PLC® program. It is also used for diagnostics purposes and for testing the parameterized functions.

The MultiMotion program module with the "Kinematics" technology module is not a complete application solution but rather a software template to which users have to add their own program parts. Without these additional program parts, MultiMotion is not suited for controlling automatic processes in machines.

### **Test in control mode**

When testing the parameterized functions with the MultiMotion Editor, you can directly access the interface in control mode, which means you can directly access the drive functions as well. The interface is no longer controlled by the program parts added by the user. This means that limits and inhibits in the user program might no longer be effective. Suitable precautionary measures must therefore be taken when using control mode. Using control mode is the sole responsibility of the user.

## 2.4 Bus systems

A bus system makes it possible to adapt frequency inverters and/or motor starters to the particulars of the machinery within wide limits. This results in the risk that a change of parameters that cannot be detected externally can result in unexpected, though not uncontrolled, system behavior.

### 3 System description

#### 3.1 Content of this manual

The manual describes the functions, startup and operation of the kinematic control system for the “Kinematics for MultiMotion/MultiMotion Light” technology module.

The following is an overview of what to expect in the individual chapters:

Basics of robotics	This chapter describes the basic aspects of controlling kinematic models. These are then later dealt with in detail.
Selection of Kinematics or HandlingKinematics	Two different technology modules are available for controlling kinematic models. This chapter provides information on module selection.
Startup and configuration	This chapter explains the first steps for creating a project, the configuration, and startup.  This chapter also explains basic operation using the kinematic monitor.
Kinematic models	Provides an overview of the configurable kinematic models. (For detailed information, refer to the configuration wizard in the MultiMotion Editor.)
Interpolating operating modes	This chapter explains the operating modes available for path interpolation, and shows the suitability for various areas of application.
MOVI-PLC® program	This chapter explains the structure of the MOVI-PLC® program and shows examples of program parts used for controlling automatic processes in machines.
<i>AxisGroupKin.Inst[..]</i> variable interface	This chapter provides a detailed explanation of the global variable interface for controlling the kinematic functions described.
Function blocks of the MPLCKinematics library	This chapter describes the function blocks available for use in the user program.
Applications	This chapter describes the use of special kinematics functions.  It references to variables and function blocks from the previous chapters.
Troubleshooting	This chapter shows possible problems and information on how to solve them.
Error codes	This chapter lists the error codes.
Glossary and list of abbreviations	Lists the technical terms and abbreviations used in this manual.

### 3.1.1 Writing conventions

- The manual contains many objects (paths, variables, libraries, etc.). These are written in different formats for reasons of clarity. The following table shows the writing conventions used:

Object	Notation	Example
Path	Courier font	\TechModules\Kinematics
Input signal (variable) Output signal	Italics	<i>AxisGroup-Kin.Inst[.].In.General.Ignore.SWLS.Cart</i>
Operating mode	Bold, all caps	<b>KIN_JOG_CART</b>
Program, program section	Bold, italic	<b><i>PRG_TaskMain</i></b>

- Some of the technical expressions used have commonly used (English) abbreviations. In the manual, the designation is given with the English abbreviation in brackets, for example piece coordinate system (PCS). For details, refer to chapter "Glossary and list of abbreviations" (→ 156).

## 3.2 Requirements

### 3.2.1 Hardware

The following hardware requirements apply to the use of the "Kinematics for MultiMotion/MultiMotion Light" technology module:

- MOVI-PLC® advanced or MOVI-PLC® power
- SD/CFast memory card with a sufficiently high technology level.

The technology level depends on the required functions (see chapter "Required technology level" (→ 13)).

### 3.2.2 Software

The following software requirements apply:

- MultiMotion or MultiMotion Light program module V150r100 or later
- Kinematics technology module V150r100 or later

Both software components are already included in MOVITOOLS® MotionStudio 6.0 or later.

The software is available for download from the SEW-EURODRIVE homepage ([www.sew-eurodrive.com](http://www.sew-eurodrive.com)).

### 3.2.3 Required technology level

The technology level is determined from the technology points required for the functionality used. You can calculate it using the following information. Alternatively, you can parameterize the required function in the MultiMotion configuration wizard that shows the required technology points.

The following technology points are included in the calculation:

- Technology points for the operating mode (if applicable with additional operating mode)
- Additional technology points for the functions

The following table shows how many technology points are used up for the respective operating mode (per kinematics instance):

Operating mode	Technology points (per kinematics instance)
AM_HOMING KIN_JOG_AXIS KIN_JOG_CART KIN_TARGET_AXIS KIN_TARGET_CART	2
AM_HOMING KIN_JOG_AXIS KIN_JOG_CART KIN_TARGET_AXIS KIN_TARGET_CART KIN_LIN_XY/YZ/ZX KIN_CIRC_XY/YZ/ZX	3
AM_HOMING KIN_JOG_AXIS KIN_JOG_CART KIN_TARGET_AXIS KIN_TARGET_CART KIN_LIN_XY/YZ/ZX KIN_CIRC_XY/YZ/ZX KIN_LIN_3D	4

The following table shows how many technology points are used up for the respective function (per kinematics instance):

Function	Additional technology points (per kinematics instance)
Kinematic model is not a CARTESIAN_GANTRY	+1
Use of the world coordinate system or piece coordinate system (WCS, PCS 1/2), for example for tracking applications or palletizing with varying pallet positions.	+1

The following table shows how many technology points are used up for the respective function (per MOVI-PLC®):

Function	Additional technology points (per MOVI-PLC®)
3D simulation: <ul style="list-style-type: none"> <li>Usually only needed for MOVI-PLC® on which motion and sequential programs are created.</li> </ul> A memory card with a stand-alone license for developers is available for this purpose (for example with 18 technology points).	+10 <sup>1)</sup>
MultiMotion "Full": <ul style="list-style-type: none"> <li>The "Full" feature is NOT required for the "Kinematics" technology module.</li> <li>MultiMotion "Full" is only required if additional MultiMotion axes are used for which MultiMotion Light is not sufficient.</li> </ul>	+2

1) 30 minute test mode without additional technology points

Note the following regarding the use of technology points:

- The required number of technology points must be present on MOVI-PLC®, else the functionality cannot be used.
- The required technology points are used up when MOVI-PLC® is started regardless of whether the functionality is actually used in the MOVI-PLC® project or not.

### Calculation example

The table shows the calculation of the technology points for a palletizing application, implemented using a SCARA robot with LIN\_D3 path interpolation:

Functionality		Technology points
Operating mode	LIN_3D	4
Functions	SCARA (kinematic model is not a CARTESIAN_GANTRY)	1
	Use of a piece coordinate system (PCS)	1
Total		6

## 3.3 Functions

The extensive functions of the universally parameterizable MultiMotion software platform are described in the manual "MultiMotion: Universal, Parameterizable Software Platform for MOVI-PLC®" and can be used in their entirety.

The "Kinematics" technology module uses 8 MultiMotion axes per configured kinematic instance and communicates with these axes via the global MultiMotion variable interface called "AxisInterface".

The technology module provides the following functionality:

- Jog mode of the single axes (**KIN\_JOG\_AXIS**)

- Jog mode of the Cartesian TCP coordinates (**KIN\_JOG\_CART**)
- Positioning of the single axes, asynchronous or synchronous (**KIN\_TARGET\_AXIS**)
- Positioning of the Cartesian TCP coordinates, asynchronous or synchronous (**KIN\_TARGET\_CART**)
- Motion along geometrically defined straight line segments and circle segments with automatic blending (continuous path, such as **KIN\_LIN\_...**, **KIN\_CIRC**)
- Control in different, moving coordinate systems (kinematics, world, workpieces)
- Tool transformation (*Transform.Tool*)
- Master/slave relations (such as traversing a centrally calculated curve using a kinematic model)
- Ability to implement kinematic models for special applications (customized kinematic models)
- Control of up to 8 degrees of freedom (DOF). But it is also possible to integrate additional axes in the kinematic models, such as externally controlled telescope axis or axis that moves along the entire kinematic model, and much more.



### **3.4 Areas of application**

The “Kinematics” technology module is suitable for controlling any conceivable open and closed kinematic chain in different areas of application because of the various operating modes and interfaces.

- Handling
- Palletizing / depalletizing
- Tracking
- Decoration (of pasta, for example)
- Assembly
- Processing (milling, gluing, painting, for example)
- Cinematography
- and much more

The focus is on the handling of objects. The operating modes available for path interpolation are particularly suitable for handling tasks, such as pick and place, palletizing, or tracking.

## 4 Basics of robotics

### 4.1 Continuous-path control

In contrast to single-axis control, the motion of a tool is controlled in a 3D space in continuous-path control. The user specifies the target position, the shape of the path (for example a straight line or an arc of a circle), and the kinematic quantities (such as speed, acceleration). The kinematic controller automatically moves all axes in a suitable manner so that the tool is moved along the desired path to the target position.

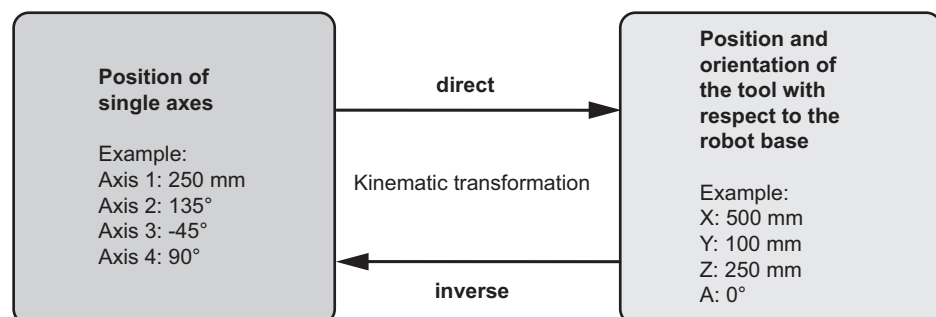
Besides continuous-path control, the kinematic controller also supports conventional control of the axes and can seamlessly switch between different operating modes.

### 4.2 Types of kinematic transformation

**Direct transformation:** The direct kinematic transformation calculates the position and orientation of the robot tool – with reference to the robot base – from the positions of the single axes. This transformation is implemented, for example, in the activation of the kinematic controller, in order to determine the Cartesian XYZ coordinates from the motor increments.

**Inverse kinematic transformation:** The inverse kinematic transformation calculates the positions of the axes with which the robot tool reaches a determined Cartesian spatial position. This transformation is implemented, for example, during path interpolation in order to lead the single axes in an appropriate way so that the tool moves along the desired path.

The following figure shows direct and inverse kinematic transformation:



9290210443

### 4.3 Examples of further transformations

The Cartesian tool position of the robot can be specified both with reference to the robot base, as well as with reference to a freely selectable world coordinate system (WCS) or piece coordinate system (PCS).

A piece coordinate system (PCS) can be placed, for example, in the corner of a pallet. The positions of the boxes on the pallet can then be expressed with reference to this piece coordinate system (PCS), for example X = 200 mm, Y = 300 mm. The palletizing robot is possibly 2 m away from the pallet. The pallet may also be rotated relative to the robot base.

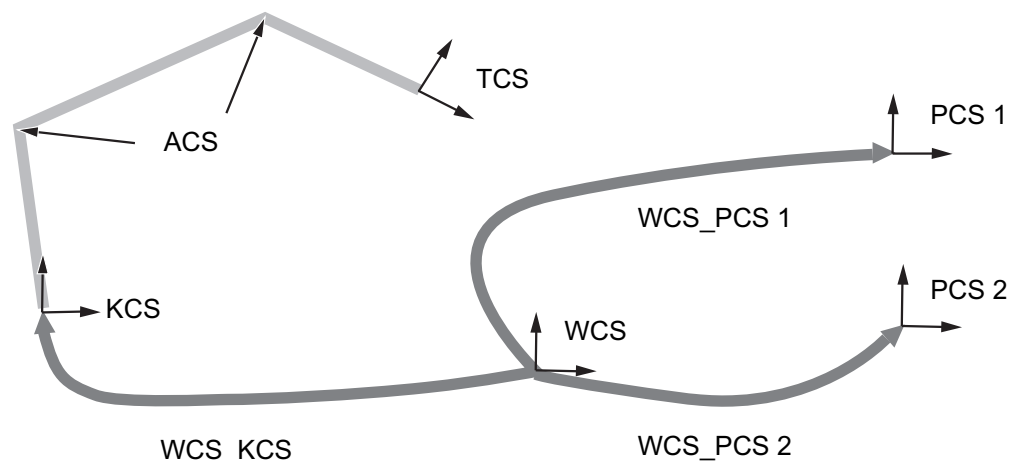
If, in this example, the transformation from the robot base to the piece coordinate system (PCS) is supplied to the kinematic control system, the robot can be easily moved to the box coordinates  $X = 200 \text{ mm}$ ,  $Y = 300 \text{ mm}$  in the piece coordinate system (PCS). This means the user need not perform any cumbersome calculation of the position of the boxes with reference to the robot base.

The required transformation involves just specifying the position and rotating the piece coordinate system (PCS) with respect to the robot base. If the pallet is, for example, slipped or turned a bit during each palletizing step, only these displacements must be detected and transferred to the kinematic control system. The coordinates of the palletizing pattern remain unchanged.

If a piece coordinate system (PCS) is in motion, for example because a cake to be decorated with icing is transferred on a conveyor belt to the robot, then decorations, such as circles, labels, etc. can be easily made on moving objects.

Moreover, the robot itself can be moved along one or more additional axes, for example to enlarge its work envelope. The variable transformation in this case from a world coordinate system, for example a reference point in a room, to the moving robot base can be easily supplied to the kinematic control system, so that the motion is compensated automatically by the robot axes.

Tool transformation is another frequently used option for extending the kinematic chain. The direct kinematic transformation ends at the robot flange. Should the Cartesian coordinates not be in reference to the flange, but instead, for example, to the center point between the jaws of a gripper, then the position of this tool center point with reference to the flange is supplied to the kinematic control system.



12549710731

#### 4.4 Cartesian position

The Cartesian pose (position and orientation) of the tool coordinate system (TCS) in a given Cartesian coordinate system, for example KCS is expressed using the following coordinates:

- Position coordinates:  $X$ ,  $Y$  and  $Z$
- Orientation coordinates:  $A$ ,  $B$  and  $C$

The position coordinates  $X$ ,  $Y$ ,  $Z$  indicate the position of the origin of the tool coordinate system (TCP) in the reference coordinate system, for example KCS.

The orientation coordinates A, B, C indicate how the tool coordinate system (TCS) is rotated with respect to the reference coordinate system. If all three values are equal to 0, then the tool coordinate system (TCS) and the reference coordinate system have the same orientation.

Otherwise, the orientation of the tool coordinate system (TCS) is derived from the orientation of the reference coordinate system as follows:

1. Rotation of the reference coordinate system with the value A about its Z axis
2. Rotation of the coordinate system rotated in the first step with the value B around its Y axis, that is around the new Y axis
3. Rotation of the coordinate system rotated in the second step with the value C around its X axis, that is around the new X axis

In the same manner, the following transformations are each described using a six-dimensional vector:

- Tool transformation:  
Position of the tool coordinate system (TCS) relative to the flange coordinate system (FCS)
- Transformation from the world coordinate system (WCS) to the kinematics coordinate system (KCS)
- Transformation from the world coordinate system (WCS) to a piece coordinate system (PCS)

## 5 Selecting Kinematics or HandlingKinematics

### 5.1 Level 1: HandlingKinematics application module for CCU

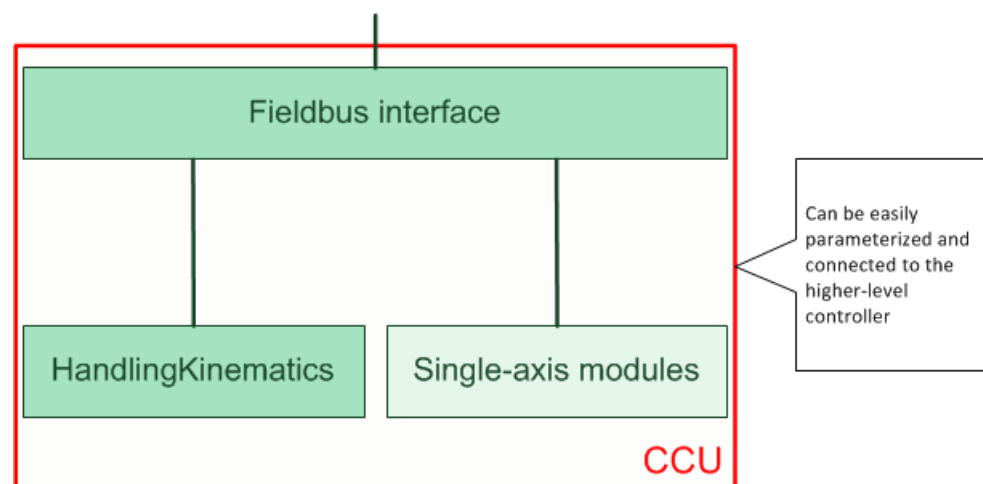
#### 5.1.1 Characteristics

- No programming is required.
- The CCU takes over the drive operating modes; the central controller coordinates the process sequence and defines the product data.
- Complete interface to the higher-level controller with up to 20 path points.
- Sequences and diagnostics can be simulated even without real machines.
- Choose from four motion types to select the perfect motion profile for your case.
- Choose a mechanism with up to four XYZ degrees of freedom and rotation around Z.
- Reproducible path fidelity with BACK-TO-PATH, even after interference.
- Suitable for static objects and combinable with up to 8 further application modules, such as for conveyor belts, lifting axes, grippers.
- Wait points can be defined for each path segment.
- Touch probe measuring function and sensor-based positioning.

#### 5.1.2 Advantages

- Performance guarantee due to encapsulated continuous-path control that has been tried and tested and has proven itself time and time again.
- Extremely quick and easy startup of the entire kinematic model using graphical software that is intuitive to use and features a clear diagnostics and monitoring function.
- The cycle time is significantly reduced due to synchronous continuous-path control with look-ahead and by-passing of interfering edges while maintaining contour accuracy.

The following figure shows the basic structure of the application module:



12572940299

## 5.2 Level 2: HandlingKinematics technology module for MOVI-PLC®

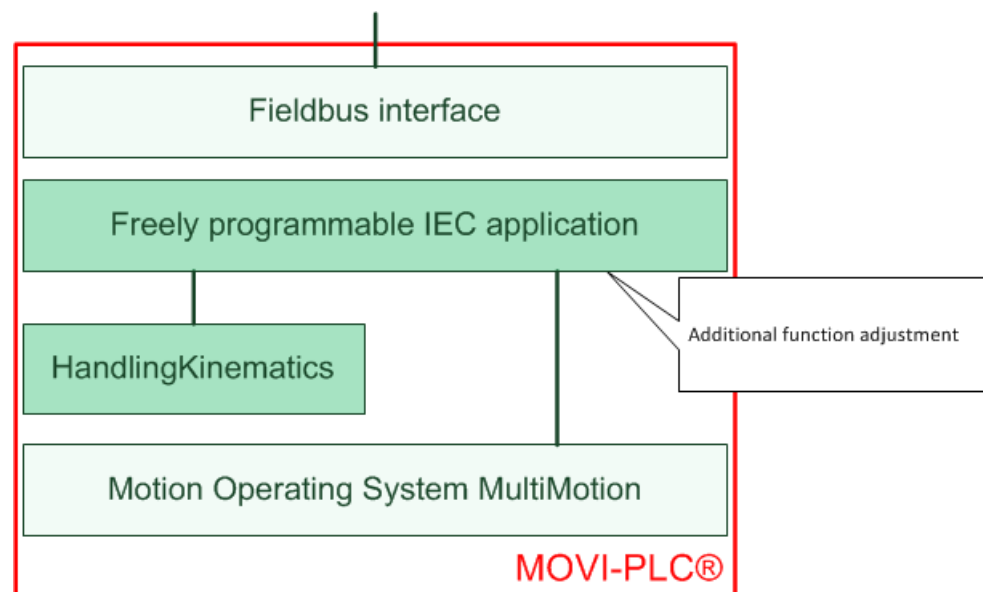
### 5.2.1 Characteristics

- Includes all functions of the HandlingKinematics CCU application module.
- Interpret and scale the resolution of the path points according to your needs.
- Practically (almost) all robots can be operated: Select a mechanism with up to 6 degrees of freedom XYZABC.
- Amend and modify the fieldbus interface according to your needs: Control the technology module directly or via signals in your MOVI-PLC® program.

### 5.2.2 Additional advantages

- Efficiency due to modular machine design. Shift all sensors and actuators that are relevant for motion to our MOVI-PLC®. Light barriers, proximity switches, vision systems, pneumatic axes are coordinated directly by us.
- Reduce data exchange at runtime to only what is necessary: Place the recipe data for path planning on the MOVI-PLC® in advance.
- Use our MOVI-PLC® power controller to monitor the interaction of multiple kinematic modules and a total of 64 drives in a practical manner.
- One software package for everything: Using MultiMotion, operate the master machine, such as a packaging machine, and the kinematic module on the same controller.

The following figure shows the basic structure of the technology module:



12572944395



### 5.3 Level 3: Kinematics technology module for MOVI-PLC®

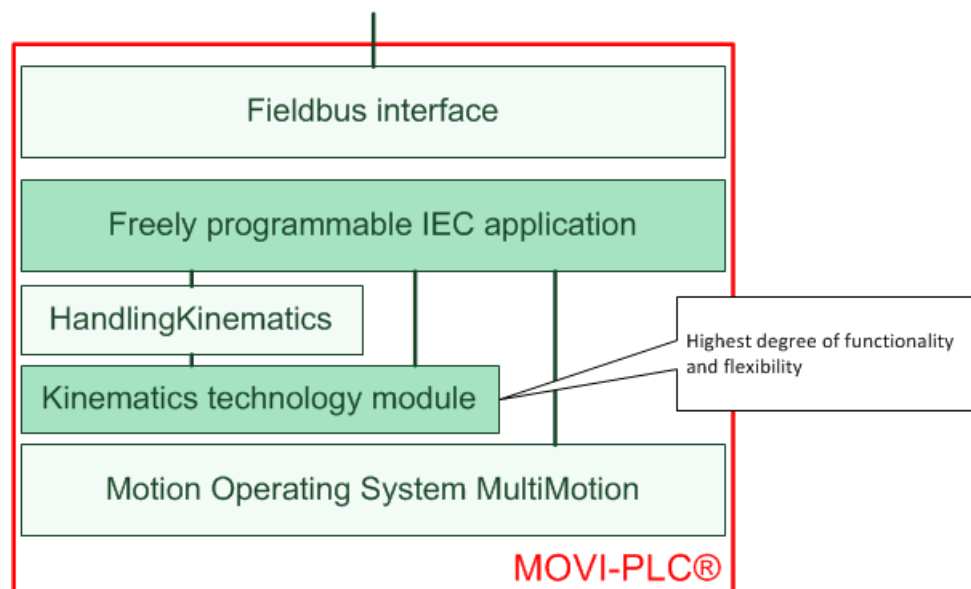
#### 5.3.1 Characteristics

- Fine-tuned access to all motion parameters in each path segment.
- Extensive options for entering circle segments.
- All coordinate systems can be used for control (also axis/world/piece coordinate systems) and it is possible to toggle between them during motion.
- Multiple kinematics instances can be synchronized on the same workpiece.
- HandlingKinematics and Kinematics technology modules can be operated on one controller.
- Master/slave relations can be implemented, such as motion of the kinematic model along the CAM profile or path progression as a function of the MultiMotion axis.
- Basic G-code import for motion control along the CAD contour.

#### 5.3.2 Additional advantages

- Solve complex tasks quickly and easily: If, for example, you want to use various coordinate systems to move the entire palletizing template depending on the front corner and the rotation of the pallet that is currently being used, then our MOVI-PLC® completes this in one program line in real time.
- You need not wait until the objects are stationary but can instead pick them while the system is running at full speed (TRACKING), e.g. piece position as a function of the signal of an external encoder.
- One software package for everything: Using MultiMotion, operate the master machine, such as a packaging machine, and the kinematic module on the same controller.

The following figure shows the basic structure of the technology module:



12572948491

**Note:**

If you want to operate the instances of both technology modules on one controller, you have to import the function blocks from the "HandlingKinematics" directory (instead of "Kinematics") when creating a MOVI-PLC® project in Step 3: Integrate the technology module (→ 28). The reason is that a "Kinematics" technology module with 1 to 1 assignment (HandlingKinematics instance n uses Kinematics instance n) exists at a subordinate level for each HandlingKinematics technology module.

**Example:**

If instance 2 is to be run as the Kinematics technology module, you must select "Kinematics" when configuring instance 2 (see Step 2: Create a MOVI-PLC® project (→ 27)). In this case, the HandlingKinematics functions of the second instance remain unused. However, if instance 1 is configured as "HandlingKinematics" (see Step 2: Create a MOVI-PLC® project (→ 27)), then you use the HandlingKinematics functions and variables from instance 1. In this case, the lower-level Kinematics technology module from instance 1 is exclusively reserved for the higher-level HandlingKinematics technology module and is served by this.

## 6 Startup and configuration

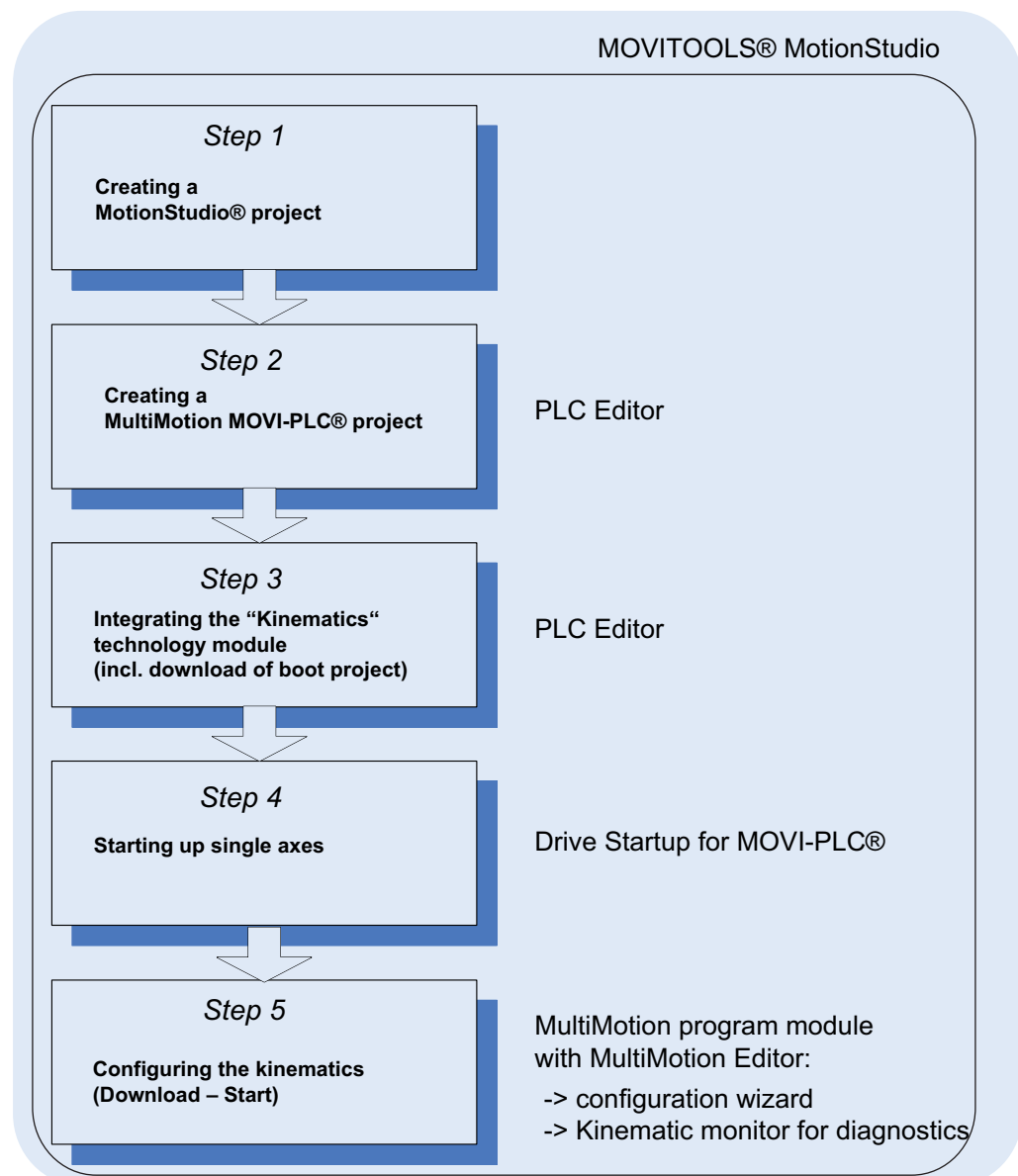
### 6.1 Procedure

You need the MOVITOOLS® MotionStudio engineering software for startup and configuration.

The following tools are included in the delivery:

- PLC Editor
- Drive startup for MOVI-PLC®
- MultiMotion / MultiMotion Light with the following tools:
  - MultiMotion Editor
  - Configuration wizard

You need all 3 tools for startup and configuration. The following figure shows the entire procedure:



9380451083

**Regarding step 4:**

- In the network view of MOVITOOLS® MotionStudio, select the **drive** that you want to start up. Next, start it up by choosing "Drive Startup" from the context menu.
- This step is not necessary if you only want to program offline for the moment, that is using simulated drives.

**Regarding step 5:**

- In the network view of MOVITOOLS® MotionStudio, select the **controller** and start the "MultiMotion"/"MultiMotion Light" technology editor.

Use the following tools to carry out diagnostics on the configured kinematic model:

- Kinematic monitor in the "MultiMotion"/"MultiMotion Light" technology editor
  - Manual mode/control mode can be activated.
- 3D simulation (starts automatically with the kinematic monitor)
  - For checking the motion sequences.
  - Adjustment of the configuration to match the real robot because the configuration is only correct when they match.
- MessageHandler (is opened from the context menu of the controller)
  - Provides detailed plain text information on errors and warnings.

A detailed description of all steps is available below.

After you have successfully completed the configuration, create your motion and sequential program (see chapter "MOVI-PLC® program").

## 6.2 Requirements

- Check the inverter installation, the encoder connection, and the MOVI-PLC® installation based on the installation notes in the operating instructions and in the field-bus manuals.
- Ensure that you have a sufficient number of technology points for the kinematic model to be configured. For details, refer to chapter "Required technology level".
- Special settings that go beyond the basic functionality require the MotionStudio® authorization level to be changed to "Advanced" or "Advanced configuration".

To learn how to change the MotionStudio authorization level, refer to chapter "Changing the MotionStudio authorization level" (→ 39).

## 6.3 Step 1: Create a MotionStudio project

More information on MOVITOOLS® MotionStudio is available in the documentation for this software.

Proceed as follows:

1. Connect the engineering PC/laptop with MOVI-PLC® via Ethernet.  
Make sure the IP address of the engineering interface and of the connected MOVI-PLC® are in the same subnetwork. The default IP address of MOVI-PLC® is 192.168.10.4. This means the IP address of the engineering interface should be set to 192.168.10.xxx.
2. Supply the MOVI-PLC® with voltage.
3. Start MOVITOOLS® MotionStudio.

4. Create a new project in MOVITOOLS® MotionStudio (choose [Project] > [New] from the menu).

The project file is saved in the specified target directory.

5. Configure the engineering interface you want to use for accessing the MOVI-PLC® (Choose [Network] > [Communication Connections] from the menu).
6. To find the MOVI-PLC®, scan the network in "Online" mode ([Scan] icon).



The controller on the interface is detected and automatically displayed in the network view of MOVITOOLS® MotionStudio.

7. Configure the MOVI-PLC®, for example by selecting the device and dragging it to the project view of MOVITOOLS® MotionStudio.

A window opens into which you have to enter a device name. Next, the device parameters are uploaded.

The configured device is displayed in the project view. A blue circle indicates that the device has been configured. During configuration, a subfolder with the device name is created in the "Devices" subdirectory of the MotionStudio project folder.

We recommend that you do not include the firmware here because saving then takes a relatively long time. You can make the relevant setting by choosing [Settings] > [Options] > [Other] from the menu.

8. Create a MOVI-PLC® project as described below.

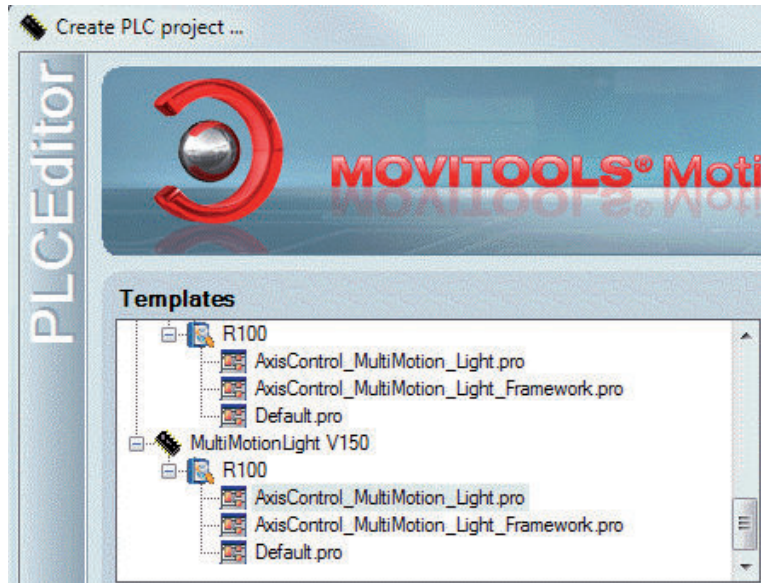
## 6.4 Step 2: Create a MOVI-PLC® project

Create a MOVI-PLC® project with the aid of the PLC Editor as follows:

1. Start the project wizard. To do so, right-click the node of the MOVI-PLC® (choose [Programming] > [New PLC Editor project] from the context menu).

2. In the project wizard, choose the following template:

AxisControl\_MultiMotion\_Light



9308425483

Note regarding the other projects that can be selected:

- For kinematic control, there are no additional advantages to using MultiMotion Full.
  - If you want to use a sample application framework, create a project with the suffix `_Framework`.
3. Assign a project name. Do not change the default directory because this is where the MultiMotion Editor searches for the symbol files.

The PLC Editor opens. The project template is the basis of the control program to which the user can add own program modules.

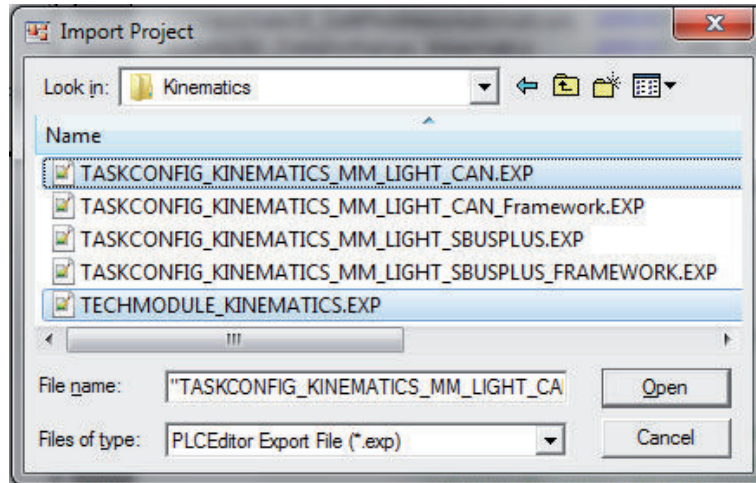
4. Integrate the "Kinematics" technology module as described below.

## 6.5 Step 3: Integrate the technology module

Do the following to import the "Kinematics" technology module into the MultiMotion/ MultiMotion Light MOVI-PLC® project:

1. Choose [Project] > [Import] from the menu:





9308509707

2. Import the following two export files from the directory "\\Your PLC Project Folder"...\TechModules\Kinematics:
  - TASKCONFIG\_KINEMATICS\_MM\_XXX\_XXX.EXP  
 xxx depends on the selected bus system or device (CAN, SBUSPLUS, FDCadv) and also depends on the created MOVI-PLC® project (MultiMotion or MultiMotion Light – with or without sample application framework).
  - TECHMODULE\_KINEMATICS.EXP

## INFORMATION



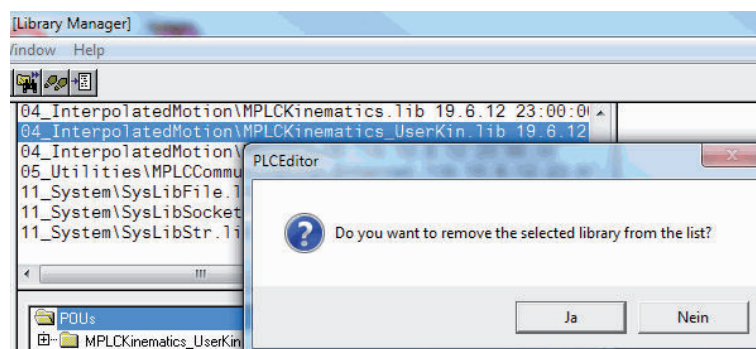
For the import, make absolutely sure to navigate to the PLC project directory that you created in "Step 2: Create a MOVI-PLC® project (→ 27)". The window that appears does not automatically navigate to this directory. If you import the wrong files, you will either not be able to compile the PLC project without errors or it will not work correctly.

3. Confirm replacement of **Task\_Main** and **Task\_Priority** during the import.

If you are using a customized USER kinematic model (such as Articulated), you must delete the following library in the library manager:

"04\_InterpolatedMotion\MPLCKinematics\_UserKin.lib"

and add the corresponding UserKin library from SEW-EURODRIVE.



9308559115

4. Compile the project and load it to the MOVI-PLC® (choose [Online] > [Log in] from the menu and [Project] > [Compile]).

In the first step, you can start the project template without any changes.

5. Generate a boot project (choose [Online] > [Generate boot project] from the menu).

## INFORMATION



Note that the task configuration is replaced during import.

If you do not create a new MOVI-PLC® project but want to integrate the "Kinematics" technology module into an existing MOVI-PLC® project with a previously modified task configuration, then you should extend the task configuration manually. To do this, you can import the "Kinematics" technology module, for example, into a new project and copy the relevant task entries from this project or add them manually as described in the "Task configuration" chapter.

### 6.6 Step 4: Start up the single axes

Proceed as follows to start up the single axes:

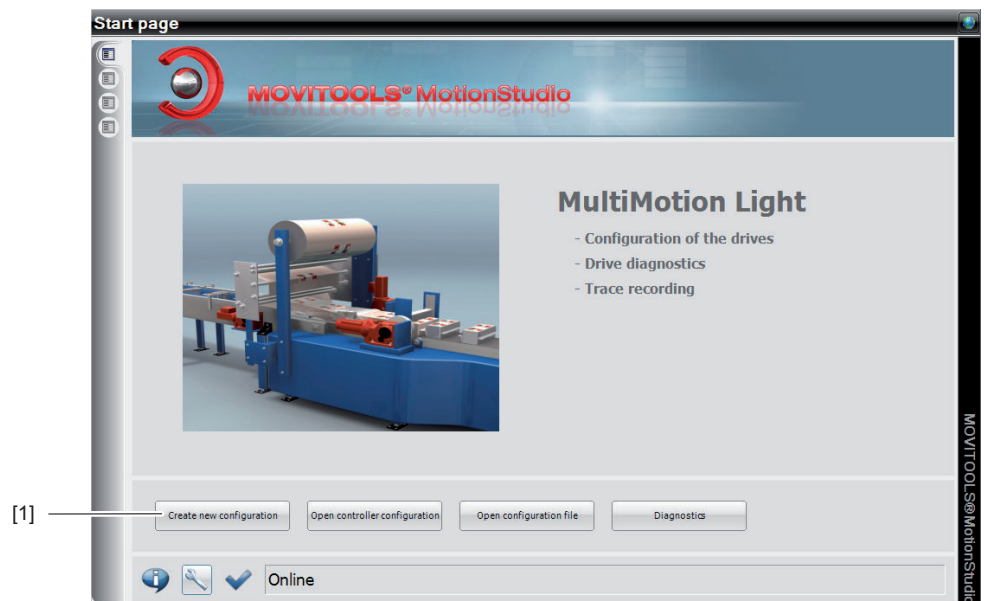
1. Start the "MOVITOOLS® MotionStudio" engineering software.
2. Run the "Drive Startup for MOVI-PLC" startup wizard for all drives.

### 6.7 Step 5: Configure the kinematic model

#### 6.7.1 Adding a new technology module

Proceed as follows to add a new "Kinematics" technology module:

1. On the initial screen of the MultiMotion editor, click the [Create new configuration] button [1].



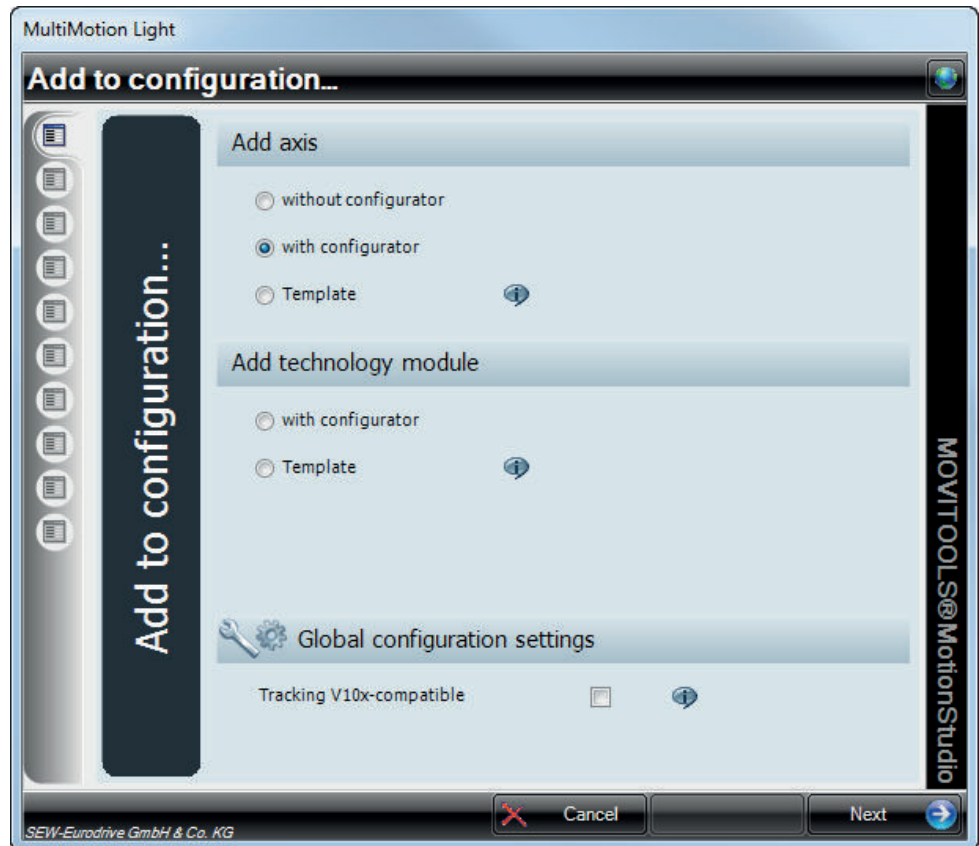
9409047563

A window for selecting the technology module opens.



9463746955

2. Choose "Kinematics" with "Configurator" as the technology module and click [Next] to continue.



9463750667

The configuration wizard opens.

3. Follow the instructions of the wizard. For details, refer to the following sections.

When you have successfully completed your entries, the wizard closes and adds the configured "Kinematics" technology module. You can then open and edit the technology module any time using the MultiMotion Editor.

### 6.7.2 Selecting "Kinematics" or "HandlingKinematics"

The Kinematics and HandlingKinematics technology modules are compared in chapter "Selecting Kinematics or HandlingKinematics (→ 21)".

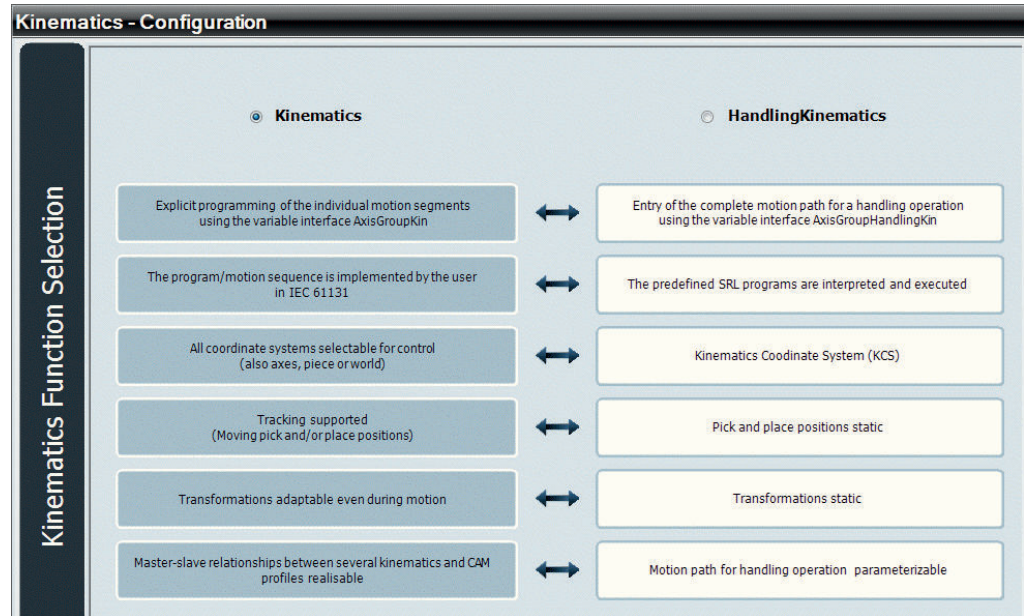
The special features of the "HandlingKinematics" technology module are documented in the "HandlingKinematics technology module for MultiMotion/MultiMotion Light" manual.

As the HandlingKinematics technology module is based on the Kinematics technology module, basic kinematic descriptions and properties are valid for both technology modules.

In addition, this manual describes the extensive functionality of the Kinematics technology module.

Proceed as follows to select the technology module:

- To configure the Kinematics technology module, click the "Kinematics" button.

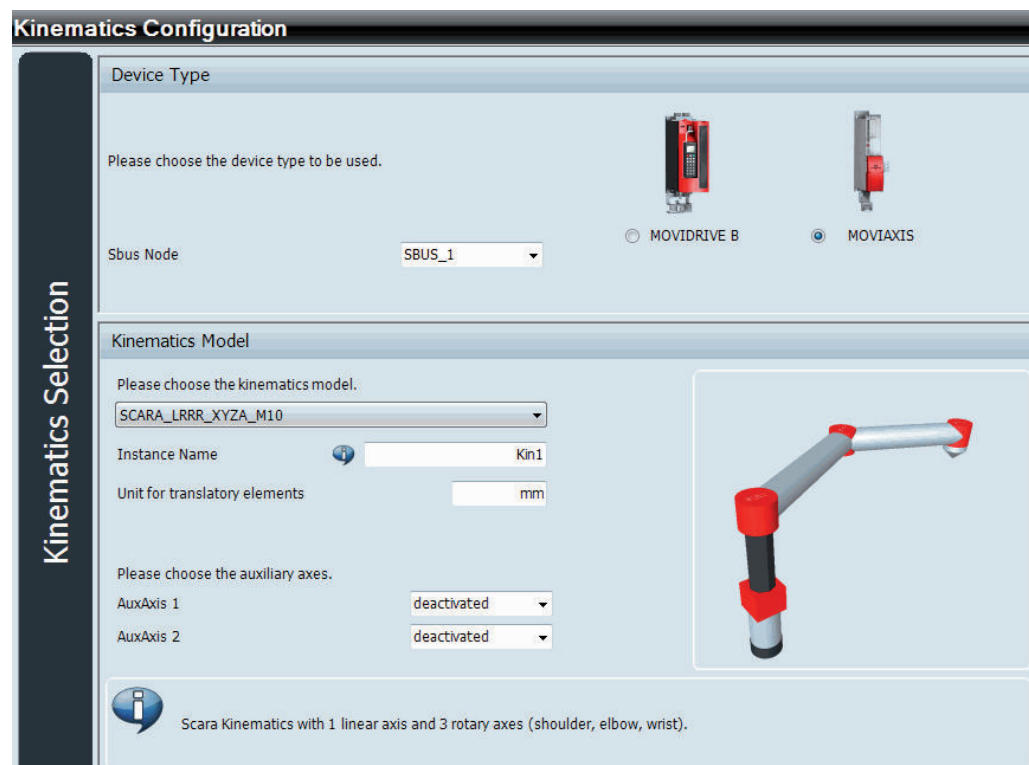


12497284747

### 6.7.3 Selecting device type and kinematic model

Proceed as follows to select the device type and kinematic model:

1. Select the device type for the inverter, as well as the bus type.



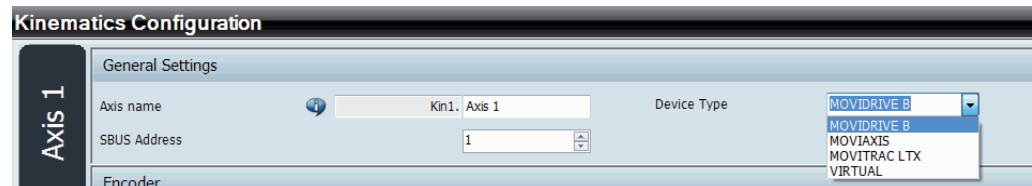
9308358283

2. Select the required kinematic model. For details, refer to chapter "Kinematic models" (→ 57).

When selecting the kinematic model, ensure that the memory card has a sufficient number of technology points. For details, refer to chapter "Required technology level" (→ 13).

### Setting up individual device types

If a device type is different to the other inverters used in the kinematics instance, you can set individual types in the "Axis settings" in the "Advanced" MotionStudio authorization level.

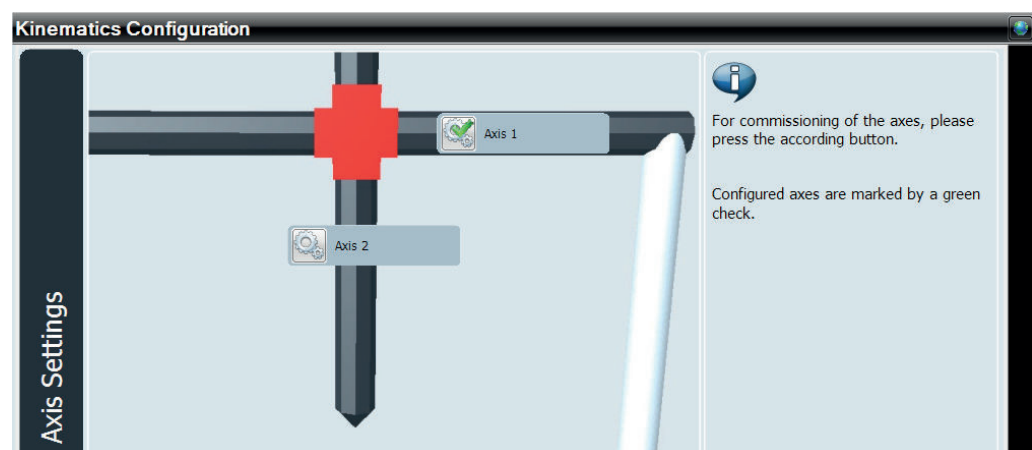


9308351115

## 6.7.4 Configuring axes

To configure the axes, continue the configuration wizard:

1. Click the button of the axis you want to configure.



9400622475

2. Make the following axis settings one after the other:
  - General settings (axis name and SBUS address)
  - Encoder (source of actual position)
  - Scaling (gear ratio, etc.)
  - Homing parameters
  - System limits (rapid stop ramp, maximum motor speed, lag error window, software limit switches, etc.)
  - Ramps for axes in jog mode (**KIN\_JOG\_AXIS**)
  - Using the digital inputs and hardware limit switches

Once all the settings for the axis have been made, a check mark appears on the button to confirm this.
3. Repeat the process for all additional axes.



The following sections provide information on adapting the kinematic model to the real axis settings.

#### Important information about setting the software limit switches:

### INFORMATION



The software limit switches are set with reference to the zero point and the direction of movement of the respective **real** axis.

The zero point is the motor position at 0 increments, and the direction of movement is positive for increasing motor increments.

### INFORMATION



The software limit switches of the axes are checked in each operating mode.

They may also be ignored by setting the *AxisGroupKin.Inst[...].In.General.Ignore.SWLS.Axis* variables so that motion outside the software limit switches of the axes is possible.

#### 6.7.5 Adapting model axes to real axes

In order for the kinematic control to work correctly, you must adapt the selected kinematic model to the real axes.

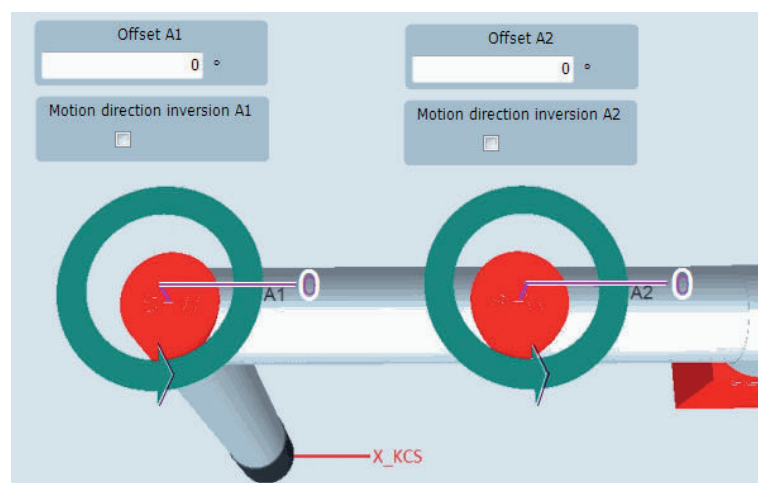
For example, trigonometric calculations are carried out for Delta or SCARA kinematic models. These calculations work only properly if the applied angle between the arm parts turned towards each other have the correct reference point and the correct direction.

Accordingly, you must set the direction of rotation reversal for an axis when the positive rotation of the model axis differs from the positive direction of movement of the real axis (increase in motor increments).

In addition, you must enter an offset for an axis when the zero position of the model differs from the axis zero. This offset is specified in the previously configured user travel units. The offset specifies where the zero position of the model is in reference to the real axis zero along the real direction of movement.

#### Display of the zero position of the model and positive direction of movement

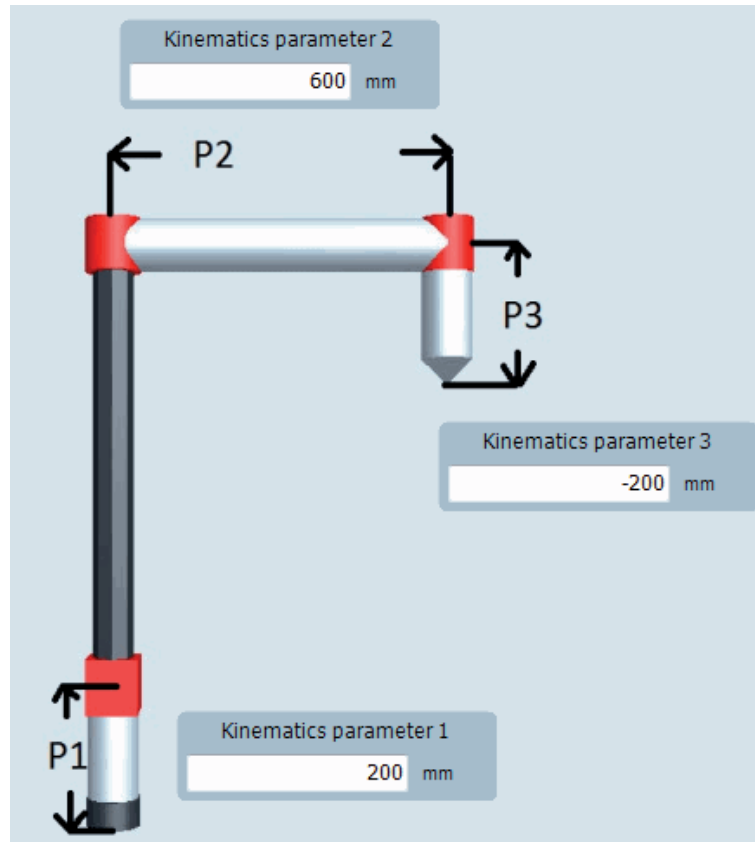
The zero position as well as the positive direction of movement of the model are displayed with the aid of arrows in corresponding figures in the configuration wizard.



9308407691

### 6.7.6 Setting kinematic parameters

Example: Setting the distances between the joints of the kinematic chain



9308378635

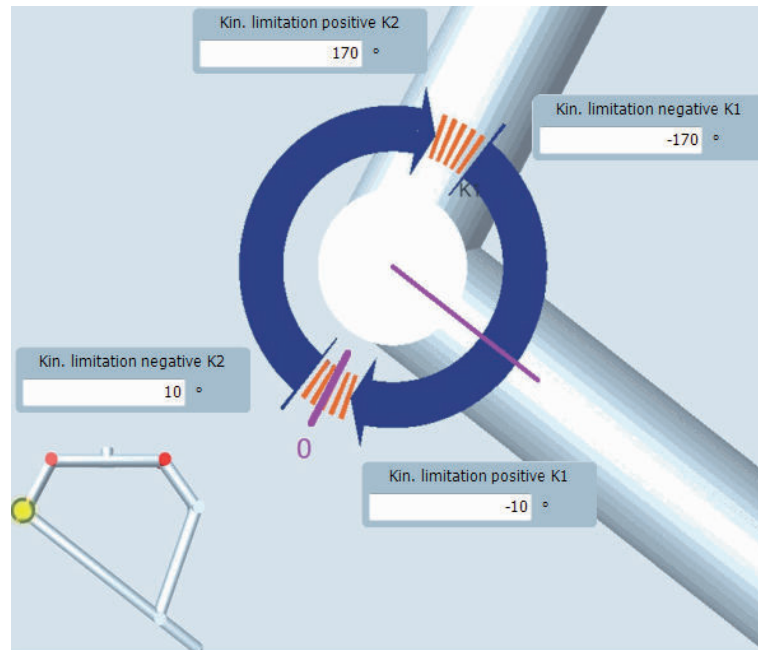
### 6.7.7 Setting kinematic software limit switches

Besides the software limit switches of the axes and the Cartesian software limit switches, some kinematic models also include a third type of software limit switch, that is the kinematic software limit switches.

**Setting valid/prohibited angle areas**



You can use the kinematic software limit switches to define, for example, valid or prohibited angle areas of the joints to which a drive is not directly assigned:



9308386571

## INFORMATION



The kinematic software limit switches are checked in each operating mode.

They may also be ignored by setting the *AxisGroup- Kin.Inst[.].In.General.Ignore.SWLS.Kin* variables so that motion outside the kinematic software limit switches is possible.

### 6.7.8 Cartesian settings

This is where you make the following settings (among others) for the Cartesian degrees of freedom (X, Y, Z, A, B, C):

- Cartesian software limit switches
- Ramps for Cartesian jog mode (**KIN\_JOG\_CART**)
- Rapid stop ramps

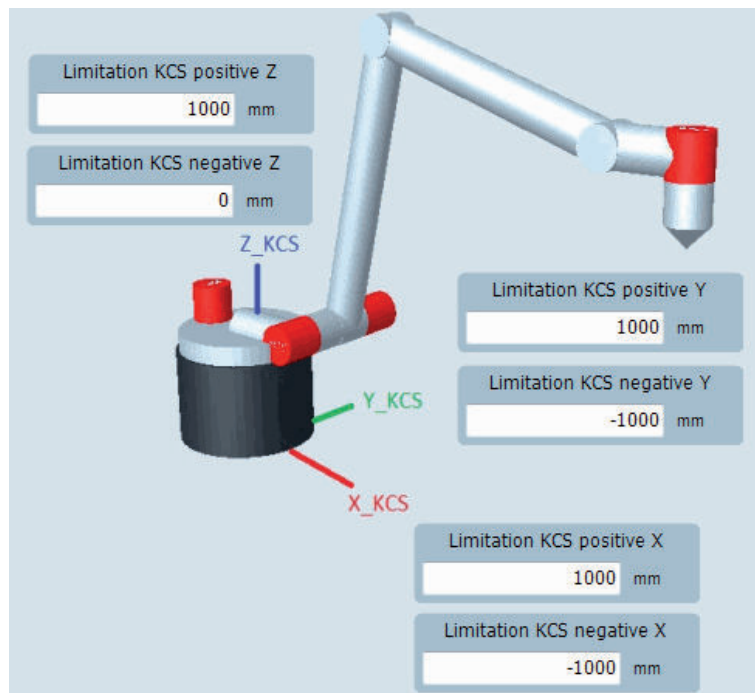
## INFORMATION



The Cartesian software limit switches are checked in each operating mode.

They may also be ignored by setting the *AxisGroupKin.Inst[.].In.General.Ignore.SWLS.Cart* variables so that motion outside the Cartesian software limit switches is possible.

The Cartesian rapid stop ramps are used for Cartesian decelerating of the motion in the currently set coordinate system if the kinematic control system is in the **KIN\_JOG\_CART** or **KIN\_TARGET\_CART** mode when a kinematics error occurs or when setting *xisGroupKin.Inst[.].In.General.Kin.RapidStop*.

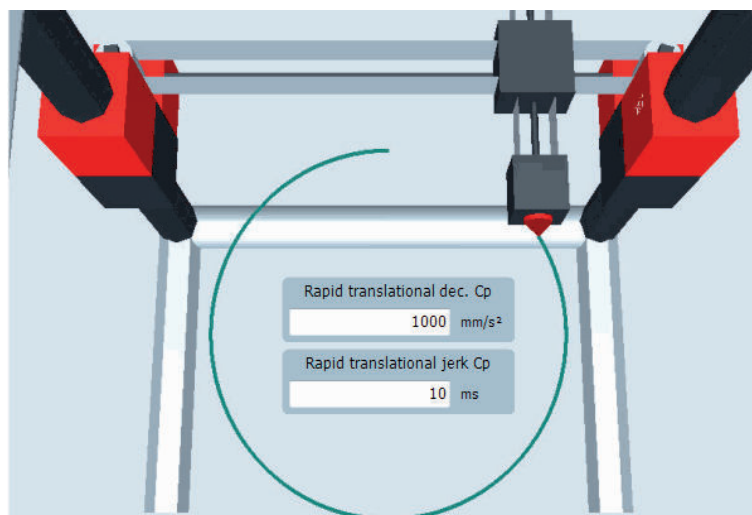
**Example: Setting Cartesian software limit switches**

9308331659

**6.7.9 Continuous path settings**

This is where you set the rapid stop ramps for continuous path translation.

The CP rapid stop ramps are used for decelerating the motion along the CP path in the currently set coordinate system if the kinematic control system is in a continuous path mode when a kinematics error occurs or when setting *AxisGroup-Kin.Inst[.].In.General.Kin.RapidStop*.

**Example: Setting rapid stop ramps**

9308337931

6.7.10 Other settings

This is where you set the IP address of the engineering PC on which the 3D simulation runs.

6.7.11 Default assignment of input variables

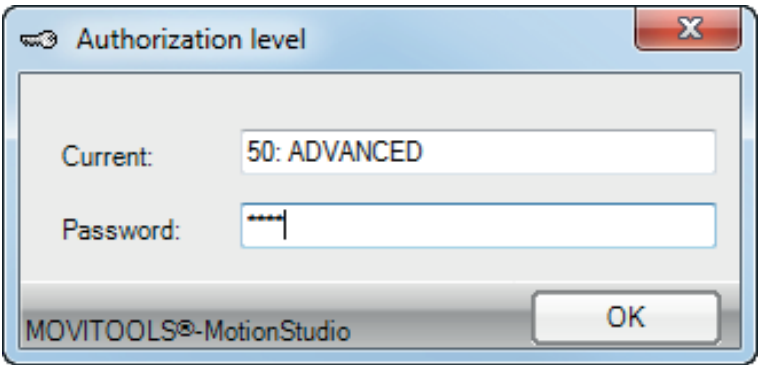
The default values of selected input variables are assigned when the MOVI-PLC® is started. To change the values, you can use either the advanced configuration or you can switch to the "Advanced" MotionStudio authorization level.

Changing the MotionStudio authorization level

Proceed as follows to change the MotionStudio authorization level:

- 1. In MOVITOOLS® MotionStudio, choose [Settings] > [Authorization levels] from the menu.

A window opens for entering the password for the required authorization level:



9455251211

- 2. Enter "5249" in the bottom line.  
The authorization level changes to "Advanced" in the line above it.
- 3. Restart the MultiMotion/MultiMotion Light technology editor.

Changing the default assignment

When running the wizard to configure the kinematic model, an additional window is available in which you have detailed access to all configuration settings in list form.

Do not make **any changes** to the top area of the list without consulting SEW-EURODRIVE.

The parameters for the default assignment of the selected input variables are at the end of the list. If required, you can adapt this default assignment according to your requirements.

(Expert)	In.Cp.BackToPath.MaxInitialDistance	10	mm
	In.Cp.Segment.Blending.LimitationPercentage	50	%
	In.Cp.Segment.Blending.Velocity.Percentage	100	%
	In.Cp.Segment.Blending.Velocity.Profile	CENTRIFUGAL	

9308344331

The assignments are made when starting the MOVI-PLC® as soon as the *AxisGroup-Kin.ConfigDataAvailable* variable is set to TRUE.

(Observe the information in the following "Download" section.)

The following table shows the input variables with their default values:

Input variable	Default	Unit
In.Transform.Tool[1..6]	0	[User unit]
In.Transform.WCS_KCS[1..6]	0	[User unit]
In.Cp.BackToPath.MaxInitialDistance	10	[User unit]
In.Cp.Segment.Blending.LimitationPercentage	50	%
In.Cp.Segment.Blending.Velocity.Percentage	100	%
In.Cp.Segment.Blending.Velocity.Profile	HANDLING	-
In.Cp.Path.FidelityPercentage	95	%
In.Cp.Segment.Circ.EllipticDistortion-Factor	1	-
In.Cp.Path.VelocityPercentage	100	%
In.Target.RapidDynamicsAtOvershooting	FALSE	-
In.Target.CoordSys	KCS	-
In.General.Kin.OverridePercentage	10	%
In.General.Ignore.Inverter[1..8]	FALSE	-
In.Jog.CoordSys	KCS	-
In.Jog.VelocityPercentage	10	%
In.Simu3D.Enable	TRUE	-

The meanings of the input variables are to be found in chapter “*AxisGroupKin.Inst[.]* variable interface”.

#### 6.7.12 Download

The MOVI-PLC® is restarted when downloading the configuration. During the restart, the MOVI-PLC® adopts the configuration data from the XML files located on the memory card.

### INFORMATION



Ensure that your user program does NOT start before *AxisGroupKin.ConfigDataAvailable* is set to TRUE.

Otherwise, the configuration data and the default assignments for the input variables will not be available. Any assignments from the user program (such as override) will be overwritten.

## 7 Diagnostics

### 7.1 Kinematic monitor

#### 7.1.1 User interface

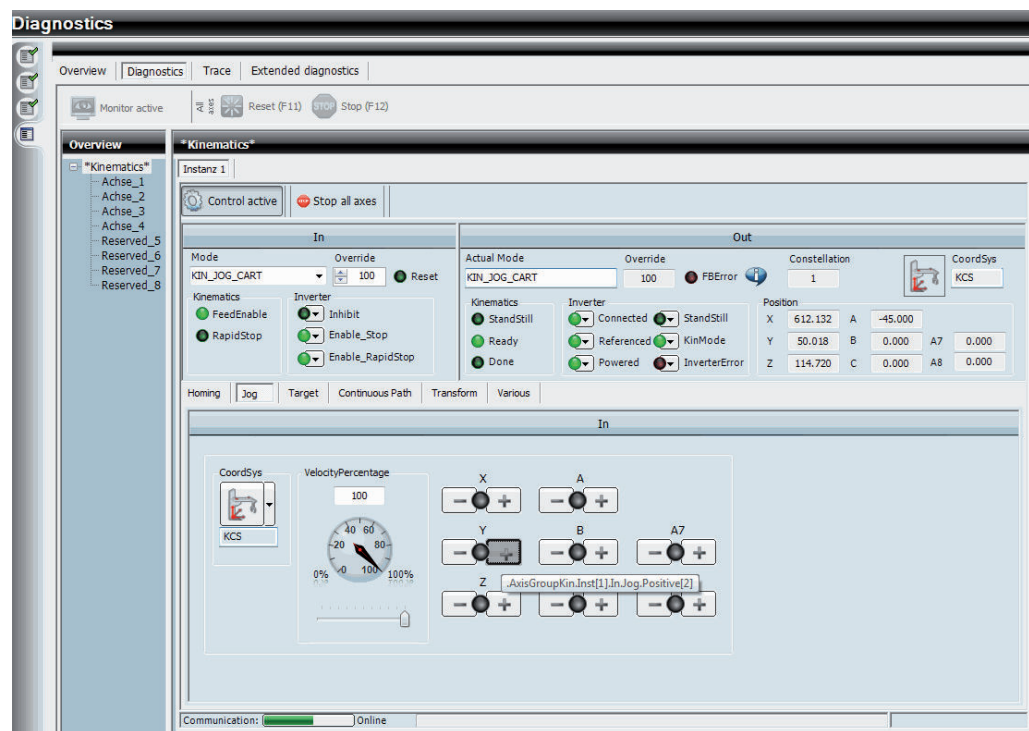
##### Calling up the kinematic monitor

Proceed as follows to call up the kinematic monitor:

1. Go to the initial screen of the MultiMotion or MultiMotion Light Editor.
2. Click [Diagnostics].

The configuration of the technology module is transferred from the memory card of the controller to the diagnostics interface. This may take a few seconds.

The kinematic monitor opens:



9308751499

3. The following diagnostics sections are available as tabs:

Diagnostics section	Function	Comment
Overview	Detailed diagnostics of the various modules	The information comes directly from the controller and is fieldbus-independent.
Diagnostics	Shows the input and output signals of the <i>AxisGroupKin.Inst[n]</i> variable interface. For details, refer to chapter "AxisGroupKin.Inst[.] variable interface". (→ 89)	You can choose either "Monitor mode" or "Control mode".
Trace	Record of different process signals (velocity, position of the axis etc.).	You can record up to 4 channels at once.

Diagnostics section	Function	Comment
Extended diagnostics	Extended diagnostics is used for expert diagnostics.	Use extended diagnostics only after consultation with SEW.

### Undocking tabs

You can “undock” the windows of tabs and display them separately. Proceed as follows:

1. Keep the control key pressed down and double-click the "In"/"Out" area of the relevant tab.

The window is "undocked" and displayed separately.

2. Close the window to dock it again.

### Displaying tooltips for variables

The names of all variables that you can set in the kinematic monitor are available as a tooltip. Using the name, you find detailed information in chapter "AxisGroupKin.Inst[..] variable interface" (→ 89).

In addition, the display of the corresponding variables is an aid to programming.

### 7.1.2 Monitor and control mode

The “kinematics” technology module has its own monitor and control mode for diagnostics:

- **Monitor mode:** Monitor mode displays the diagnostics data of the selected technology module. The monitor has only read access to the respective interfaces.

[Monitor active] is indicated on the top left-hand button when you are in monitor mode. Clicking this button lets you change to control mode.

**NOTICE** The system must be in a safe condition when changing to control mode. You have to confirm a relevant message.

- **Control mode:** In control mode, the monitor has write and read access to the interfaces of the technology module. It is possible to set or reset control signals directly on the interface and to set variable values, such as position or velocity.

[Control active] is indicated on the top left-hand button when you are in control mode.

Clicking this button lets you change to monitor mode again. Before you can exit the monitor area to change to the configuration area, for example, you have to disable control mode again.

When control mode is enabled, the *AxisGroupKin.HMI.HMIControl* variable in the global variable structure in the “Kinematics” technology module is set (TRUE).

In this case you must ensure that your user program is not run and that the *AxisGroupKin* variable is not written to. The corresponding case distinction is included in the sample programs imported with the technology module, see **PRG\_User\_Kinematics\_Examples** in chapter "Sample programs" (→ 83).

### Switching to control mode

Proceed as follows to switch to control mode:

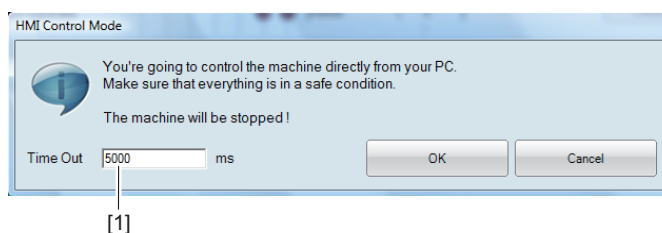
1. **▲DANGER** Unexpected movement of the machine while switching from monitor mode to control mode or the other way around.

Severe or fatal injuries.

- Make sure that an automatic restart or stop of the machine represents no danger to people or equipment.
- Make sure that the machine is in a safe state.

2. Click [Monitor active] to switch to control mode.

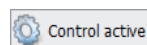
The program prompts you to set a time for the timeout [1]:



9470720779

3. Enter a suitable timeout value and click [OK] to confirm.

The button now indicates the control mode:



9470737035



### 7.1.3 Setting for kinematic mode

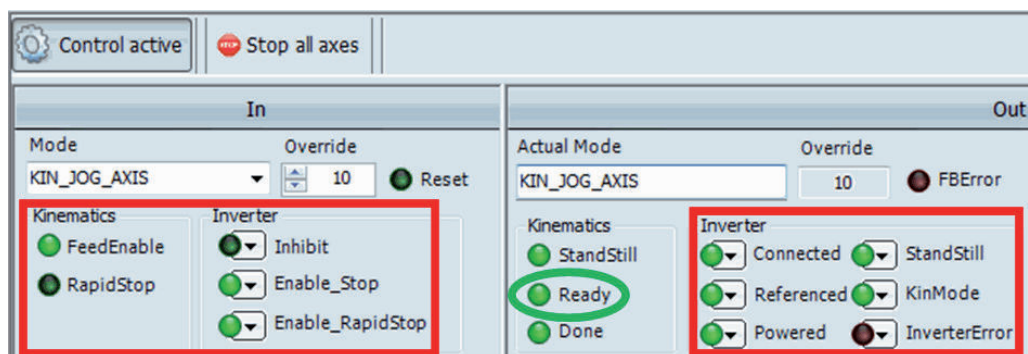
1. In order to being able to execute a kinematic mode, such as **KIN\_JOG\_AXIS**, set the following input signals (variables) in the "In" area:

"In" area	Setting value
Inhibit	FALSE
Enable_Stop	TRUE
Enable_RapidStop	TRUE
FeedEnable	TRUE
RapidStop	FALSE

2. When all input variables are set correctly, ensure that the output signals display the following values:

"Out" area	Display value	Remark
Connected	TRUE	-
Referenced	TRUE	-
Powered	TRUE	-
InverterError	FALSE	-
Inverter.StandStill	TRUE	When switching to a kinematic mode
	FALSE	For motion after having switched to a kinematic mode
Inverter.KinMode	TRUE	After having switched to a kinematic mode

The following figure shows the "In" and "Out" areas when all requirements for a kinematic mode have been fulfilled:



9308883467

## INFORMATION

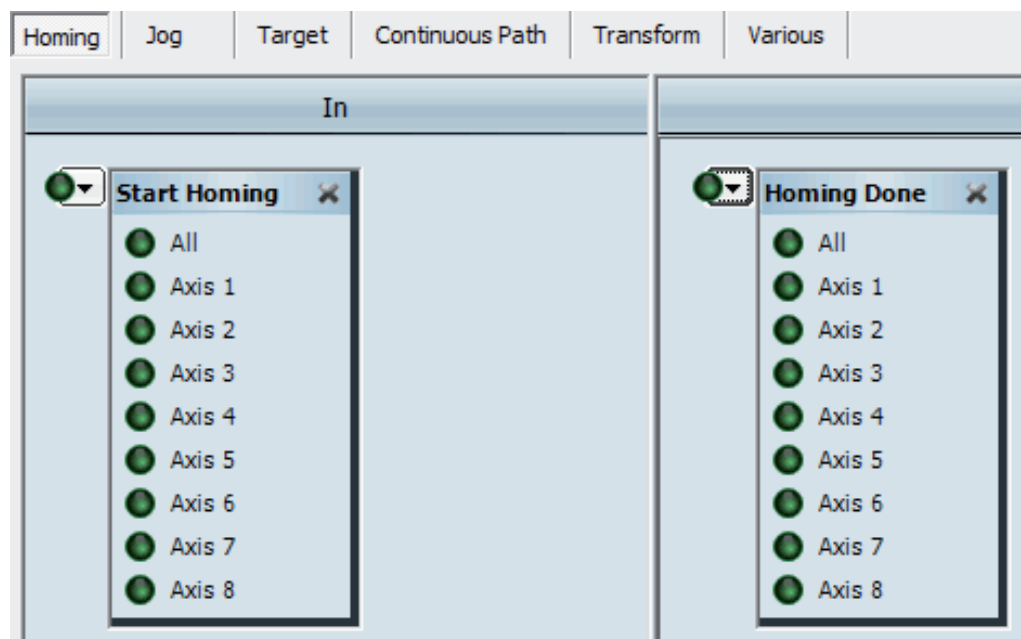


In the event of a fault, you receive detailed information in plain text in the Message tab of the kinematic monitor or in a clearly arranged display in the "MessageHandler (→ 55)" diagnostics tool.

### 7.1.4 Referencing the axes

1. Set the **KIN\_AM\_HOMING** mode.
2. Go to the "Homing" tab ("In" area)
3. Reference either all axes or certain axes by clicking the respective LED icon.

Referencing takes place in the previously configured operating mode (for example reference cam left).



9308642315

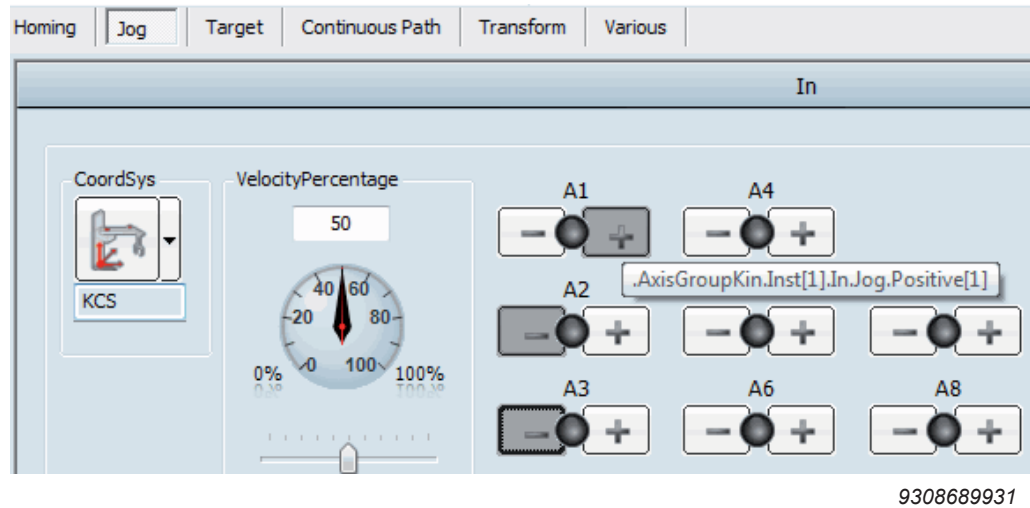
The result of the action is displayed in the "Out" area.

### 7.1.5 Axis-wise jog mode

1. Set the **KIN\_JOG\_AXIS** mode.
2. Go to the "Jog" tab ("In" area).
3. Activate jog movement for one or more axes.

To do so, click the following buttons:

- **[+]**: Positive jog movement
- **[-]**: Negative jog movement



4. If required, adjust the jog speed.

### Effect on the coordinate system

Irrespective of which coordinate system is selected, the axes are always moved in jog mode in the **KIN\_JOG\_AXIS** mode, even if a Cartesian coordinate system is set.

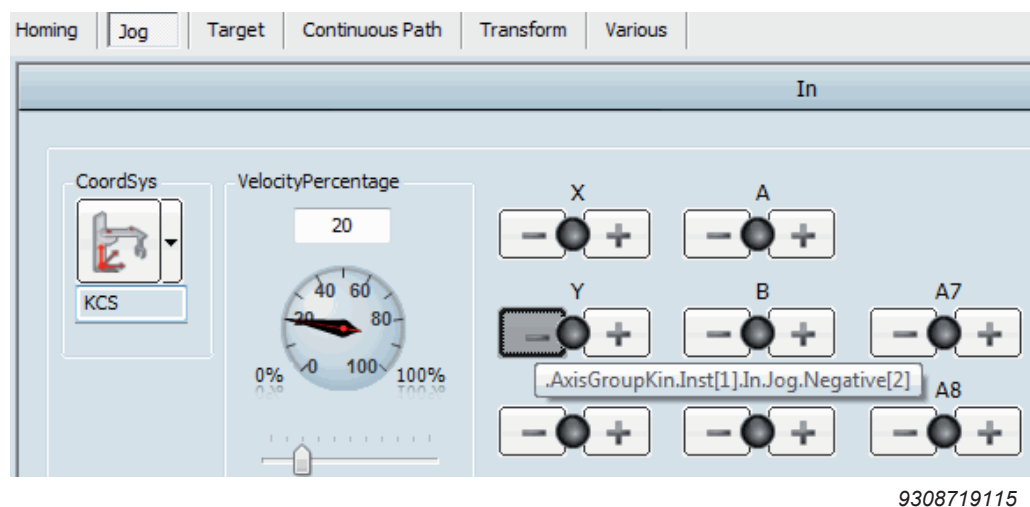
However, selecting the coordinate system sets the current coordinate system of the kinematic control system; the display of the current position in the "Out" area also refers to this coordinate system.

#### 7.1.6 Cartesian jog mode

1. Set the KIN\_JOG\_CART mode.
2. Go to the "Jog" tab ("In" area).
3. Activate a jog movement for each Cartesian degree of freedom of the tool coordinate system (TCS).

To do so, click the following buttons:

- **[+]**: Positive jog movement
- **[-]**: Negative jog movement



4. If required, adjust the jog speed.

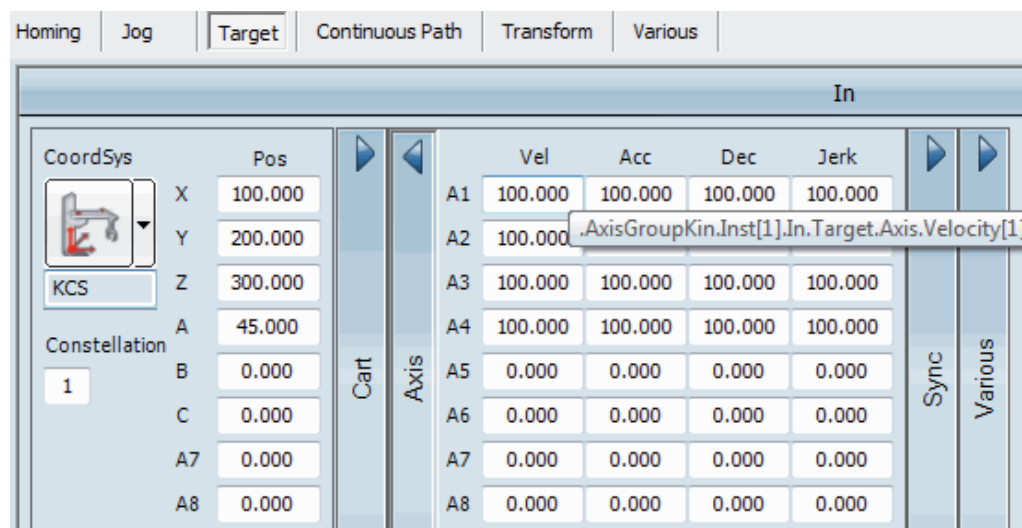
### Effect according to the coordinate system

The jog movements take place relative to the set coordinate system. When you select the axis coordinate system (ACS), the positions are displayed in the "Out" area.

However, when you select the axis coordinate system (ACS), the jog movements take place in reference to the kinematics coordinate system (KCS), as if KCS were selected.

#### 7.1.7 Axis-wise TARGET mode

1. Set the **KIN\_TARGET\_AXIS** mode.
2. Go to the "Target" tab ("In" area).
3. Set the following motion parameters for the interpolation of the axes:
  - **Velocity (Vel)** in [user units/s]
  - **Acceleration (Acc)** in [user units/s<sup>2</sup>]
  - **Deceleration (Dec)** in [user units/s<sup>2</sup>]
  - **Jerk time (Jerk)** in [ms]



9309131019

### Effect according to the coordinate system

Irrespective of which coordinate system is selected, the robot moves in axis interpolation, even if a Cartesian coordinate system is set.

When you set the axis coordinate system (ACS), enter specific target axis positions as the "Pos" target position.

When setting a Cartesian coordinate system, enter the Cartesian target coordinates of the tool coordinate system (TCS) relative to the set coordinate system.

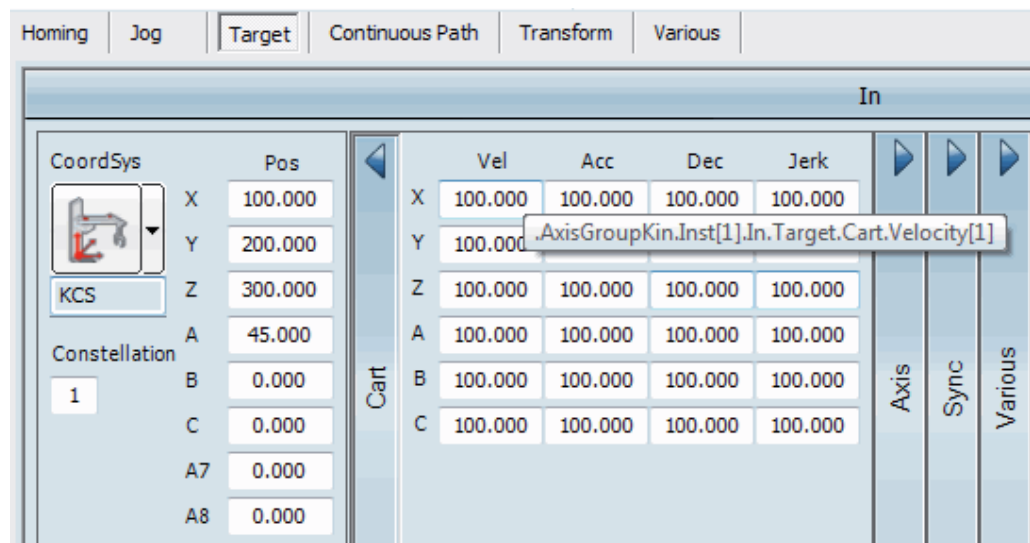
The conversion to the corresponding axis positions takes place automatically through the kinematic controller.

In this case, you can also set the "Constellation" (for example 1, 2 for elbow counter-clockwise, clockwise for a SCARA kinematic model) in which the Cartesian target position is to be moved to. If you set "Constellation" to 0, this causes the current "Constellation" to be retained.

The current "Constellation" is displayed at the top right of the "Out" area.

### 7.1.8 Cartesian TARGET mode

1. Set the **KIN\_TARGET\_CART** mode.
2. Go to the "Target" tab ("In" area).
3. Set the following motion parameters for the interpolation of the Cartesian degrees of freedom (DOF):
  - **Velocity (Vel)** in [user units/s]
  - **Acceleration (Acc)** in [user units/s<sup>2</sup>]
  - **Deceleration (Dec)** in [user units/s<sup>2</sup>]
  - **Jerk time (Jerk)** in [ms]



9309404555

#### Effect according to the coordinate system

Irrespective of which coordinate system is selected, the tool coordinate system (TCS) moves by means of Cartesian interpolation, i.e. along the X coordinate, Y coordinate, etc.

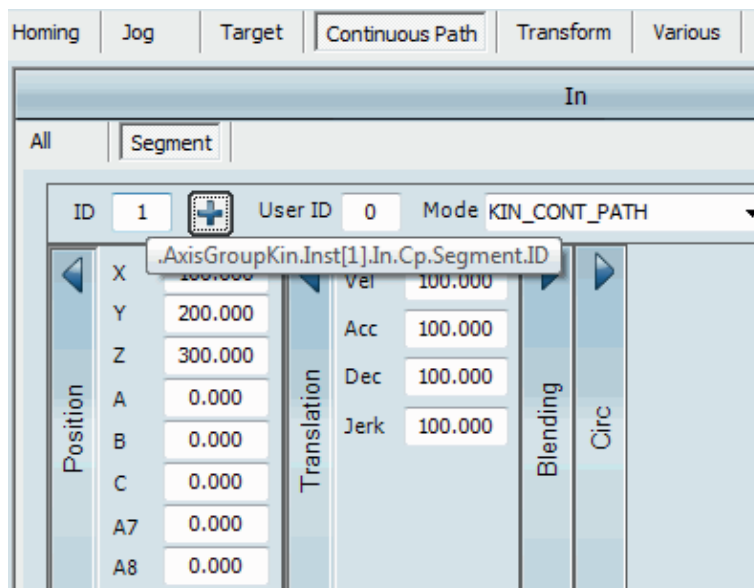
The Cartesian target coordinates refer to the configured coordinate system.

When you select the axis coordinate system ACS, enter the target axis positions. These are automatically converted to Cartesian target coordinates by the kinematic control system and moved to in the kinematics coordinate system (KCS), as if KCS were set.

The axis positions are displayed in the Out range when configuring ACS.

### 7.1.9 ContinuousPath mode

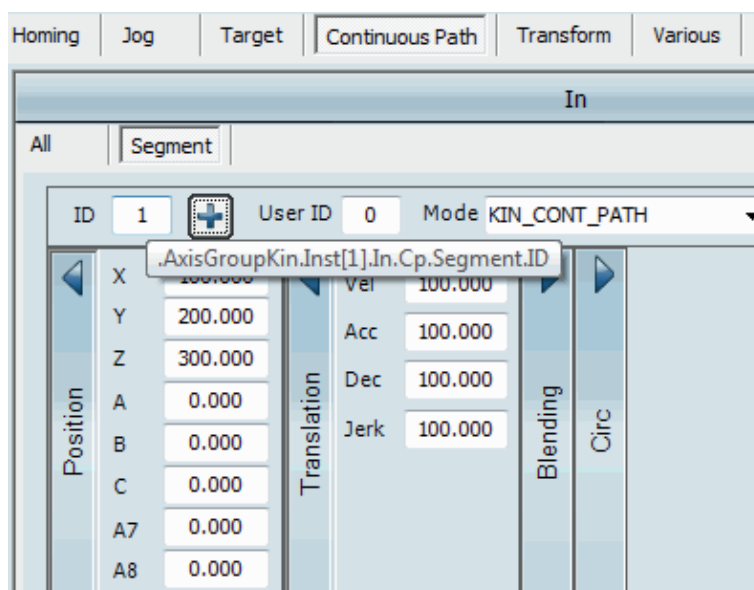
1. Set the **KIN\_LIN\_3D** mode.
2. Go to the "Target" tab ("In" area).
3. Set the following parameters for the translational motion along the path:
  - **Velocity (Vel)** in [user units/s]
  - **Acceleration (Acc)** in [user units/s<sup>2</sup>]
  - **Deceleration (Dec)** in [user units/s<sup>2</sup>]
  - **Jerk time (Jerk)** in [ms]



9308735115

- Enter the target coordinates (X, Y, etc.) in the "Pos" group.
- Click the [+] button.

This increases the segment ID by "1" and the current segment is taken over by the kinematic control system.



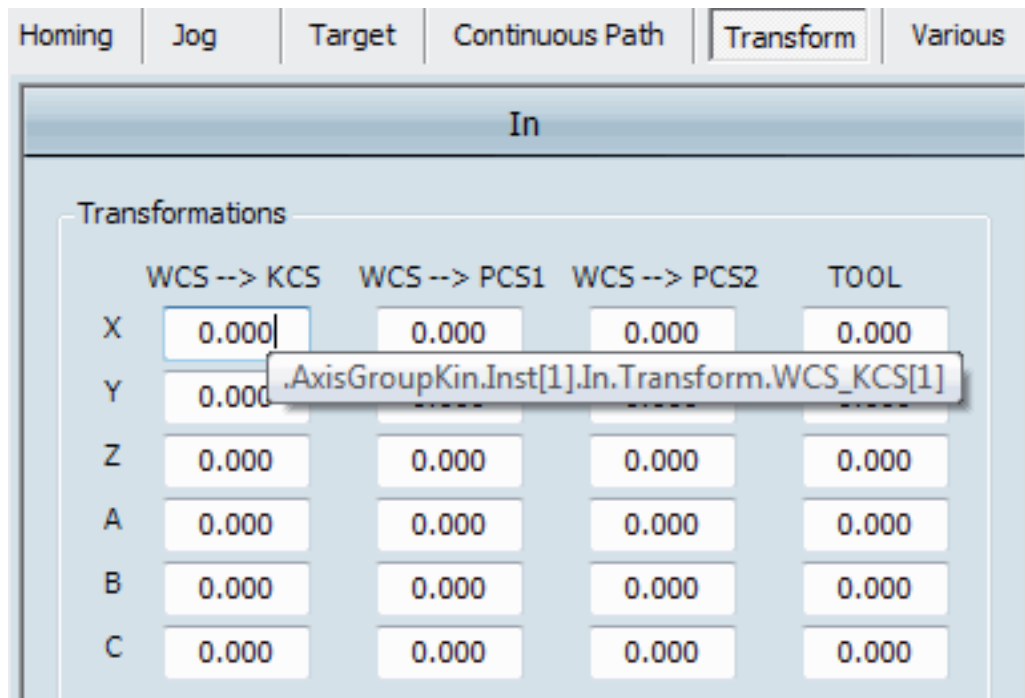
9308735115

For parameterizing circular interpolation, refer to chapters "KIN\_CIRC\_XY / \_YZ / ZX" (→ 75) and "In.Cp.Segment" (→ 99).

## 7.1.10 Selecting the transformation

1. Go to the "Transform" tab ("In" area).
2. Select the transformation:
  - **WCS --> KCS**
  - **WCS --> PCS1**

- **WCS --> PCS2**
- **TOOL (corresponds to FCS --> TCS)**



9309665675

### Inconsistent change

Note that inconsistently changing a transformation can lead to a lag error or to very prolonged motions. This is the case, for example, when the transformation is used for Cartesian interpolation simultaneously with the inconsistent change or a few cycles of TaskPriority later.

For details, refer to the "Changing the coordinate system" (→ 124) chapter. The transformations used for controlling the kinematic model may only be changed inconsistently if one of the following conditions is met:

- The inconsistent change to the transformation takes place while the kinematic control system is in axis interpolation. The kinematic control system must also remain in axis interpolation in the following 5 cycles of TaskPriority.

In axis interpolation, the kinematic controller is in the **KIN\_JOG\_AXIS** or **KIN\_TARGET\_AXIS** modes, or in **KIN\_STOP** mode, if **KIN\_JOG\_AXIS** or **KIN\_TARGET\_AXIS** was previously set.

- If the kinematic controller is not ready (*Out.General.Kin.Ready* = FALSE).

### Inhibiting transformations

For the one-to-one assignment of axis values to Cartesian coordinates, the permitted transformations are partly restricted depending on the configured kinematic model. In such cases, refer to the information in plain text that is provided in the messages that appear.

For more information, refer to chapter "Dealing with ambiguities of the ABC orientation values (→ 135)".

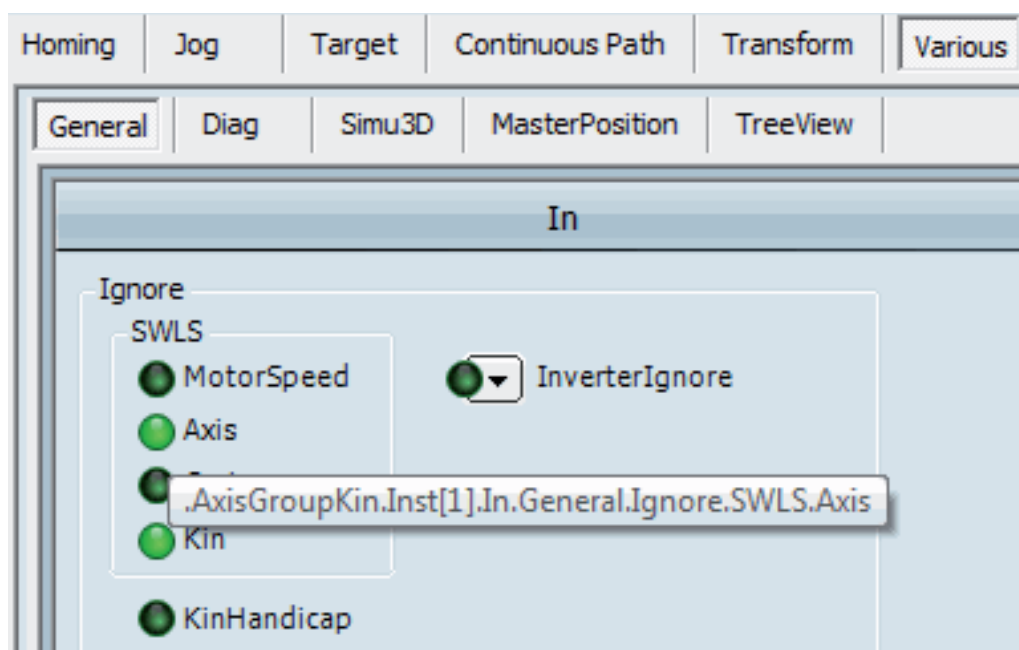
### 7.1.11 Various settings

1. Go to the "Various" tab ("In" area).

This tab contains the following lower-level tabs:

- [General]
- [Diag]
- [Simu3D]
- [MasterPosition]
- [TreeView]

2. Select, for example, the "General" tab. In the "Ignore" group, set that the settings for the software limit switches are to be ignored.



9309747979

3. For details on the "Simu3D" tab, refer to the "3D simulation" (→ 53) chapter.



## 7.2 3D simulation

### 7.2.1 Purpose

The 3D simulation integrated in MOVITOOLS® MotionStudio allows automatic simulation of all configured kinematics. The 3D models automatically adapt to the given configuration.

3D simulation is very useful for the following tasks:

- Configuring the kinematic model and adaptation to the real robot
- Learning how to use the kinematic control
- Checking motion sequences

3D simulation can be carried out both offline and online. Offline means that no real axes are connected/moved. Online means that the simulation is carried out in parallel with the real robot motion.

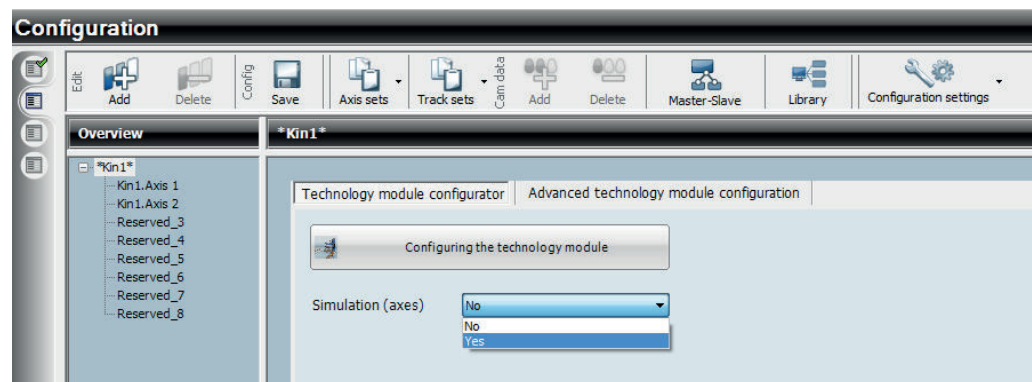
Online 3D simulation is, for example, very useful for visualizing the tool motion of the robot. Appraising the tool path in terms of path type and blending behavior is generally quite difficult for the human eye.

### 7.2.2 Requirements

Note that the "3D simulation" function requires additional technology points. For details, refer to the "Required technology points" (→ 13) chapter

1. Enter the IP address of the engineering PC in the configuration wizard.
2. If you do not have any real axes available, activate the function for simulating the axes in the MultiMotion/MultiMotion Light Editor:

To do so, go to the "Technology module configurator" tab and choose "Yes" for Simulation (axes):



9308228107

3. Establish an Ethernet connection between MOVI-PLC® and engineering PC.

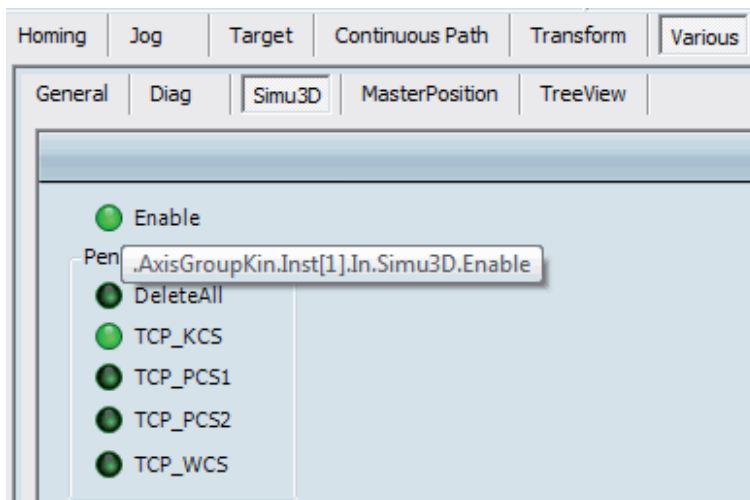
**Note:** 3D simulation is not possible via USB or fieldbus.

Communication with 3D simulation takes place solely via the Ethernet engineering interface (X37 when using MOVI-PLC® advanced, LAN 3 when using MOVI-PLC® power).

In order for you to be able to carry out the 3D simulation, the *.AxisGroup-Kin.Inst[.].In.Simu3D.Enable* variable must be set to TRUE. The default assignment is TRUE.

You can activate the variable in the user program or in the kinematic monitor:

4. Go to "Various"/"Simu3D" tab.



9308210955

5. By clicking the LED icons, you can enable pens at the TCP that draw lines in reference to various coordinate systems (such as TCP\_KCS). You can also delete these lines again (DeleteAll).

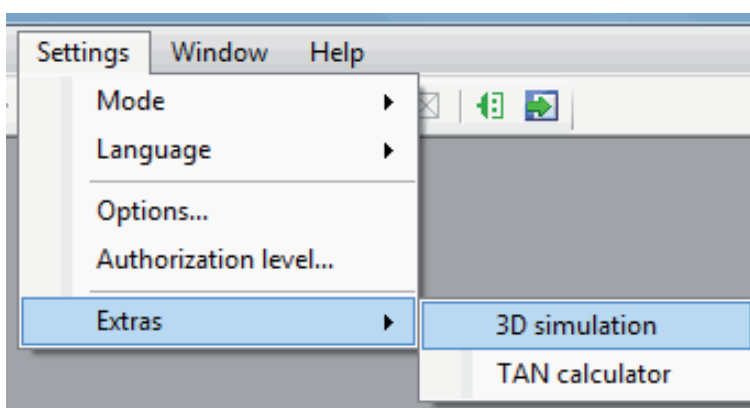
### 7.2.3 Starting 3D simulation

3D simulation starts automatically when you are working with the kinematic monitor and when *In.Simu3D.Enable* is set to TRUE.

You can also specifically start 3D simulation in one of the following ways:

- In the kinematic monitor:  
On the "Various"/"Simu3D" tab, click the [Start Simu3D Tool] button.
- With MOVITOOLS® MotionStudio:

Start the 3D simulation of MOVITOOLS® MotionStudio by choosing [Settings] > [Extras] > [3D Simulation] from the menu.

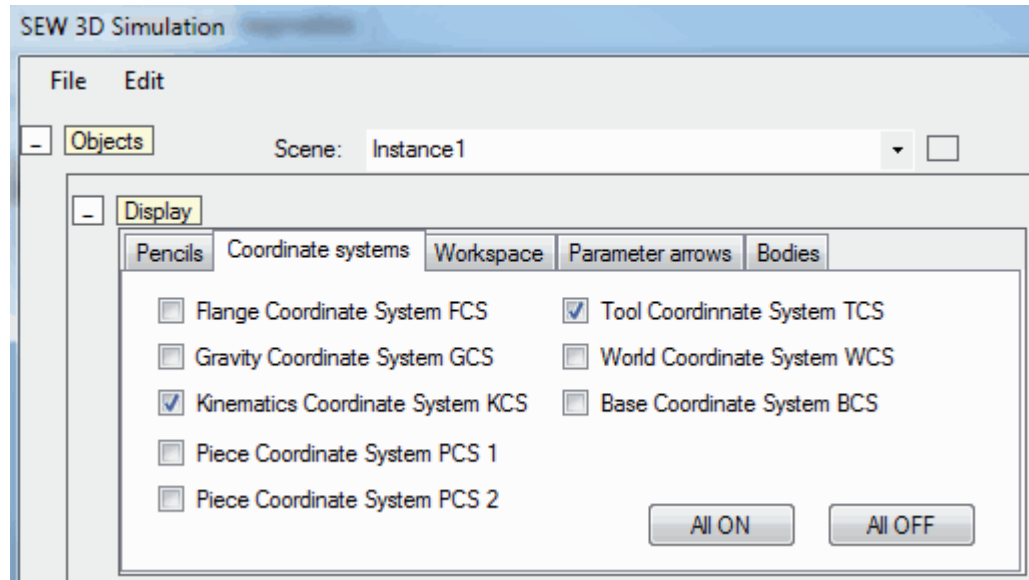


9308214667

### 7.2.4 Adjusting the 3D simulation

You can customize the look and feel of the 3D simulation in the control window that is automatically started with the 3D simulation.

1. Start the 3D simulation as described in "Starting 3D simulation" (→ 54).



9308207755

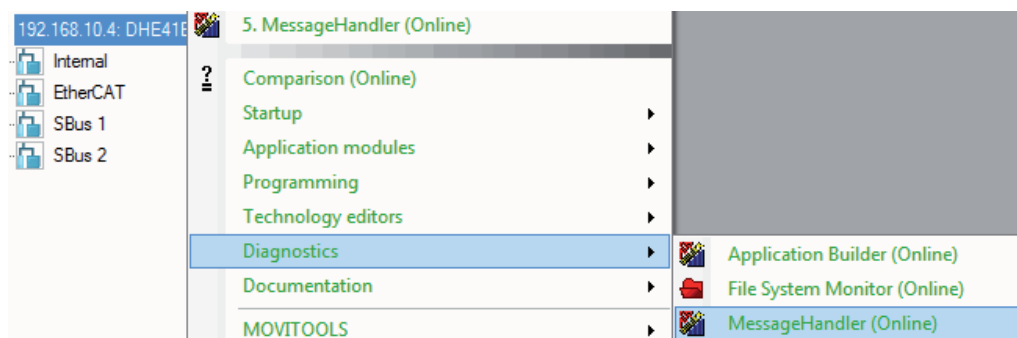
2. Adjust the 3D simulation with the aid of the following tabs:
  - **"Pencils"**: This is where you can set the color of the lines, for example. However, you cannot show/hide the pencils here.
  - **"Coordinate systems"**: This is where you can show various coordinate systems to be able to clearly check currently valid transformations.
  - **"Workspace"**: This is where you can show the arrows for displaying the zero points and directions of the valid travel ranges for the axes, the Cartesian dimensions, and the kinematic limitations.
  - **"Parameter arrows"**: This is where you can display useful information by means of arrows (for example to identify kinematic parameters such as arm lengths and offsets or arrows for displaying the software limit switches of axes).
  - **"Bodies"**: This is where you can show/hide selected bodies.

## 7.3 MessageHandler

The MessageHandler is a diagnostics tool that displays information on errors in plain text and in a clear manner. This information supplements the information that you receive in relation to the error number (error ID) and makes troubleshooting easier.

Proceed as follows to start the "MessageHandler":

1. Start the "MOVITOOLS® MotionStudio" engineering software.
2. In the network view, select the controller icon.



9308613003

3. Right-click to open the context menu and choose [Diagnostics] > [MessageHandler].

The "MessageHandler" opens.

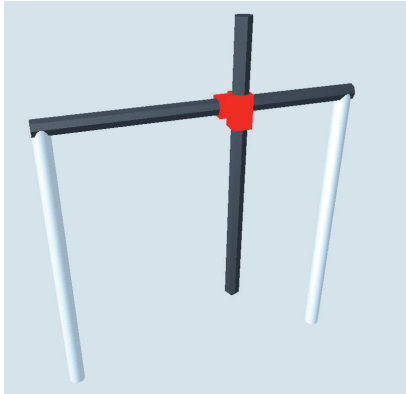
Current error messages are located in the top area. Messages that have already been archived are displayed below that.

## 8 Kinematic models

This chapter provides an overview of the kinematic models that you can select in the configuration wizard.

### 8.1 CARTESIAN GANTRY

**CARTESIAN\_GANTRY  
\_LL\_XY\_M10**



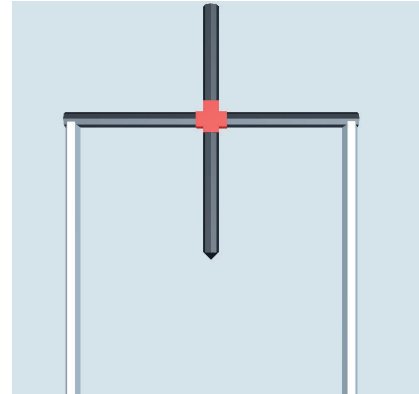
9308233355

**CARTESIAN\_GANTRY  
\_LL\_ZX\_M10**



9308238987

**CARTESIAN\_GANTRY  
\_LL\_ZX\_M15**



9308245003

Cartesian gantry with 2 linear axes:

- Axis 1 corresponds to the X direction
- Axis 2 corresponds to the Y direction

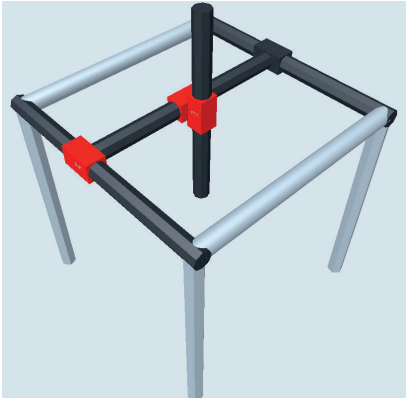
Cartesian gantry with 2 linear axes:

- Axis 1 corresponds to the Z direction
- Axis 2 corresponds to the X direction

The angle between the Z and X direction can also be adjusted to values other than 90° with the aid of a parameter.

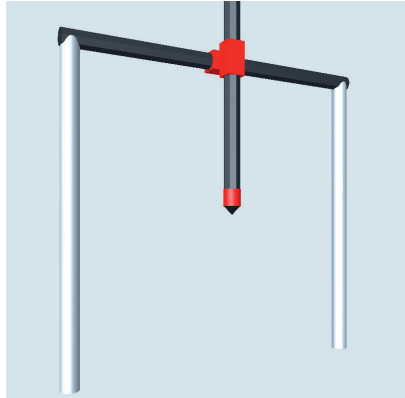
Cartesian gantry with 2 linear axes:

- Axis 1 corresponds to the X direction
- Axis 2 corresponds to the Z direction

**CARTESIAN\_GANTRY  
\_LLL\_XYZ\_M10**

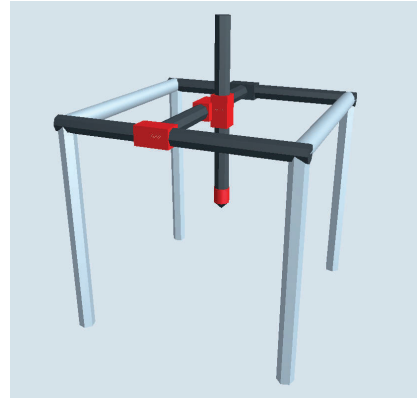
9308281483

Cartesian gantry with 3 linear axes.

**CARTESIAN\_GANTRY  
\_LLR\_XYB\_M10**

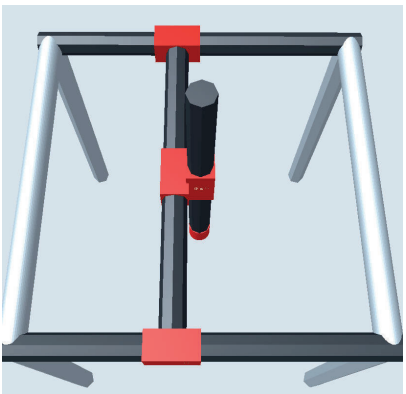
9308312715

Cartesian gantry with 2 linear axes (X, Y) and 1 rotative axis (rotation around Y).

**CARTESIAN\_GANTRY  
\_LLLR\_XYZA\_M10**

9308272907

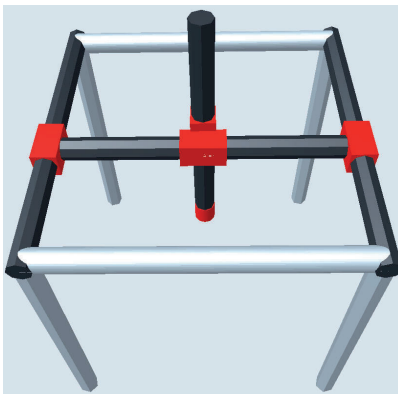
Cartesian gantry with 3 linear axes and one rotative axis (rotation around Z).

**CARTESIAN\_GANTRY  
\_LLLLR\_XYZA\_M10**

9308251403

Cartesian gantry with 4 linear axes and 1 rotative axis (rotation around Z).

The X direction is achieved with 2 axes (A1, A2).

**CARTESIAN\_GANTRY  
\_LLLLR\_XYZA\_M20**

9308258187

Cartesian gantry with 4 linear axes and 1 rotative axis (rotation around Z).

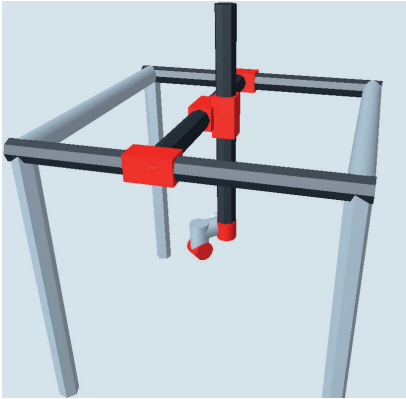
The Y direction is achieved with 2 axes (A2, A3).

**CARTESIAN\_GANTRY  
\_LLLR\_XYZAB\_M10**

9308303627

Cartesian gantry with 3 linear axes and 2 rotative axes.

#### CARTESIAN\_GANTRY\_LLLLRR\_XYZAB\_M10



9308265355

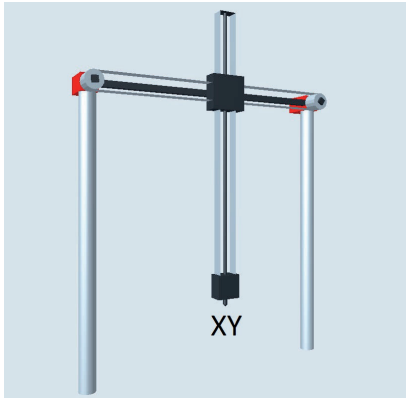
Cartesian gantry with 4 linear axes  
and 2 rotative axes.

The X direction is achieved with 2  
linear axes (A1, A2).

## 8.2 ROLLER GANTRY

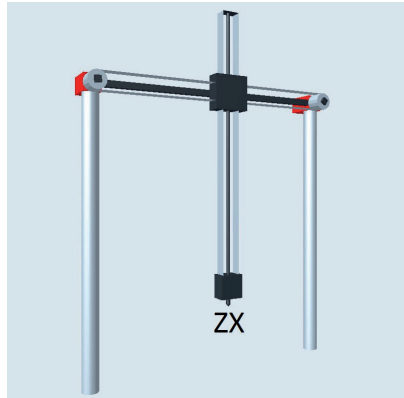
ROLLER\_GANTRY\_LL\_XY\_M10

ROLLER\_GANTRY\_LL\_ZX\_M10

ROLLER\_GANTRY  
\_LLL\_XYZ\_M10

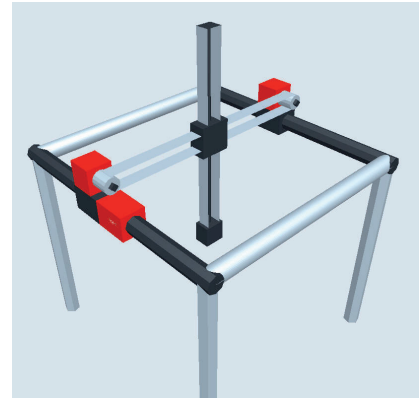
9310212747

Roller gantry with 2 axes:  
Motions take place in the XY  
plane.



9310234123

Roller gantry with 2 axes.  
Motions take place in the ZX  
plane.



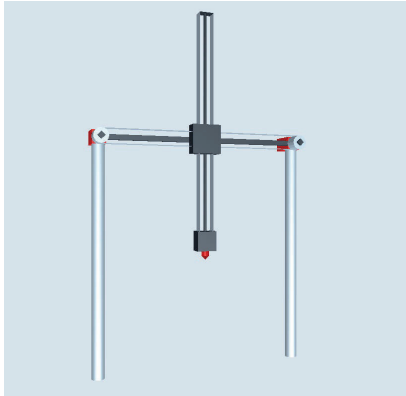
9310255883

Roller gantry with 3 axes:

- Axis 1 corresponds to the X direction
- Axes 2 and 3 drive the roller belts in the YZ plane



ROLLER\_GANTRY\_LLRL\_XYB\_M10

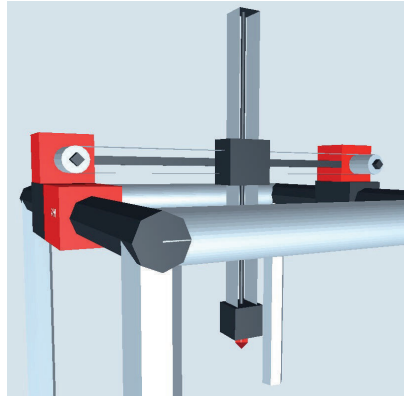


9310370443

Roller gantry with 3 axes:

- The axes 1 and 2 drive the roller belts in the XY plane.
- Axis 3 corresponds to the rotation B.

ROLLER\_GANTRY\_LLLR\_XYZA\_M10

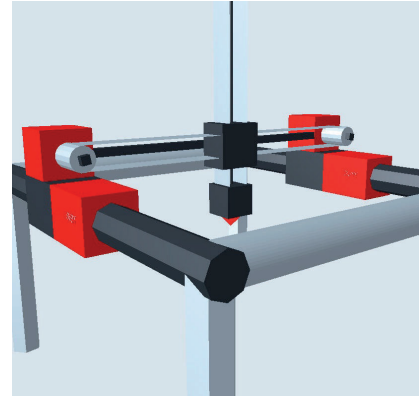


9310323467

Roller gantry with 4 axes:

- Axis 1 corresponds to the X direction.
- The axes 2 and 3 drive the roller belts in the YZ plane.
- Axis 4 corresponds to the rotation A.

ROLLER\_GANTRY\_LLLLR\_XYZA\_M10

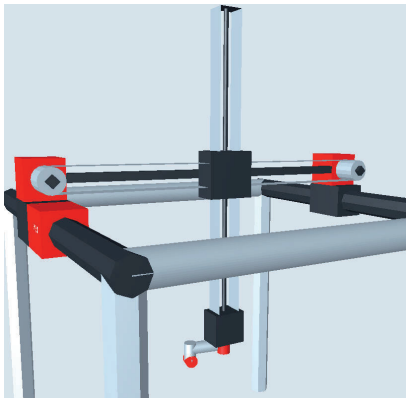


9310278027

Roller gantry with 5 axes:

- The X direction is achieved with 2 axes (A1, A2).
- The axes 3 and 4 drive the roller belts in the YZ plane.
- Axis 5 corresponds to the rotation A.

ROLLER\_GANTRY\_LLLRR\_XYZAB\_M10

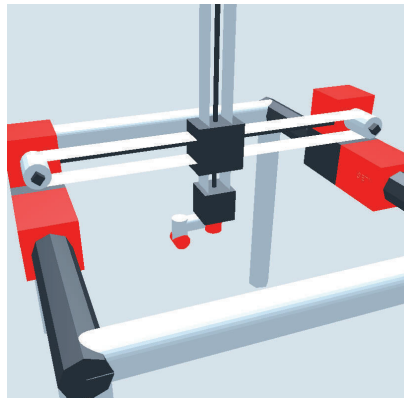


9310346763

Roller gantry with 5 axes:

- Axis 1 corresponds to the X direction.
- The axes 2 and 3 drive the roller belts in the YZ plane.
- Axis 4 corresponds to the rotation A.
- Axis 5 corresponds to the rotation B.

ROLLER\_GANTRY\_LLLLR\_XYZAB\_M10



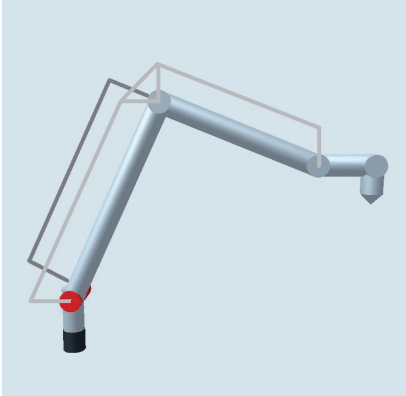
9310300555

Roller gantry with 6 axes:

- The X direction is achieved with 2 axes (A1, A2).
- The axes 3 and 4 drive the roller belts in the YZ plane.
- Axis 5 corresponds to the rotation A.
- Axis 6 corresponds to the rotation B.

## 8.3 SCARA

SCARA\_RR\_XY\_M20



9310418955

SCARA kinematic model with 2 rotational axes. The lower arm is moved by stationary drive 2 with the aid of a parallelogram or belts.

SCARA\_RRL\_XYZ\_M20



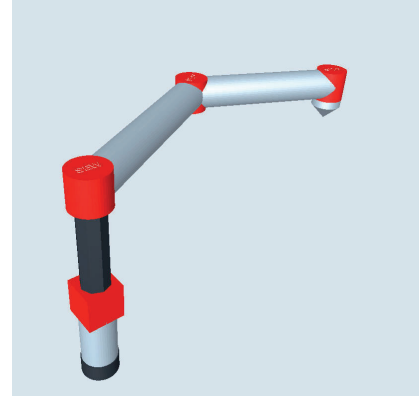
9310443787

SCARA kinematic model with 2 rotational axes (shoulder, elbow) and 1 linear axis. The flange coordinate system (FCS) can only move translationally i.e. cannot be rotated.

The flange orientation is maintained constant with the aid of a parallelogram or a belt, or the tool only comprises, for example, a pen (the orientation of which is not of significance).

The linear axis 3 corresponds to the Z direction.

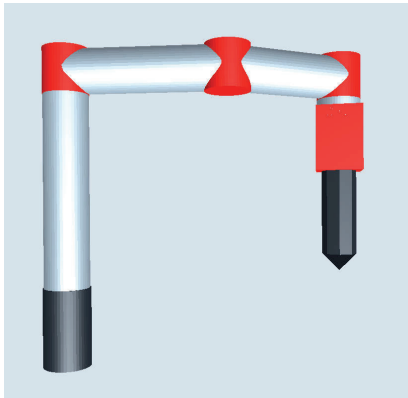
SCARA\_LRRR\_XYZA\_M10



9310394507

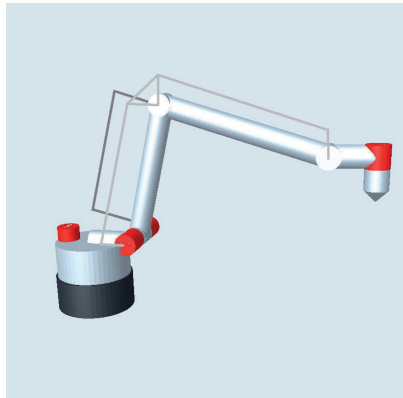
SCARA kinematic model with 1 linear axis and 3 rotational axes (shoulder, elbow, wrist).

SCARA\_RRRL\_XYZA\_M10



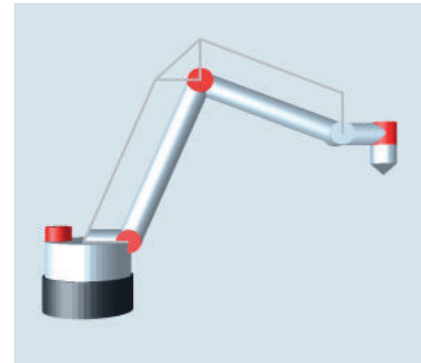
9310469003

SCARA\_RRRR\_XYZA\_M60



9310494603

SCARA\_RRRR\_XYZA\_M65



9637703179

SCARA kinematic model with 4 axes:

- 3 rotative axes (shoulder, elbow, wrist)
- 1 linear axis

SCARA kinematic model with 4 rotative axes

(turret, shoulder, elbow, wrist).

The under arm is moved by means of a parallelogram or a belt by the drive 3 which is attached to the turret.

SCARA kinematic model with 4 rotative axes

(turret, shoulder, elbow, wrist).

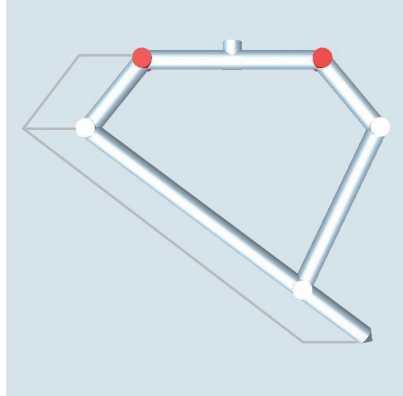
Every swivel joint is moved directly by a drive.

## 8.4 DELTA

**DELTA\_LL\_XY\_M10**

9308455051

Delta kinematic model with 2 linear axes.

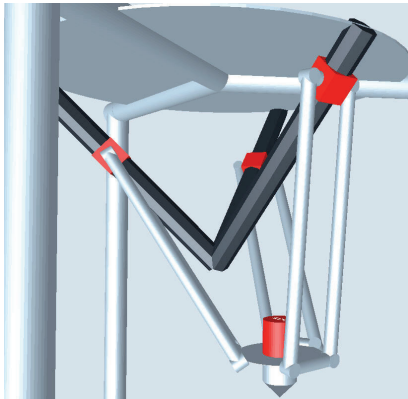
**DELTA\_RR\_XY\_M20**

9308465675

Delta kinematic model with 2 rotational axes.

## 8.5 TRIPOD

TRIPOD\_LLLR\_XYZA\_M10



12852397067

Tripod with 3 linear axes and 1 rotational axis.

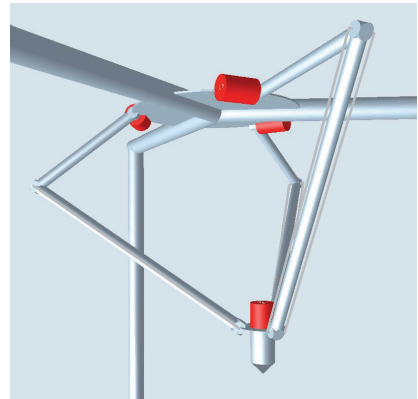
TRIPOD\_RRR\_XYZ\_M10



9310520587

Tripod with 3 rotational axes.

TRIPOD\_RRRR\_XYZA\_M10

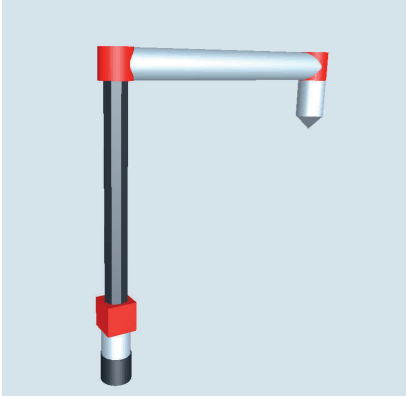


9310546955

Tripod with 4 rotational axes. Axis 4 rotates the tool around the Z axis.

## 8.6 MIXED

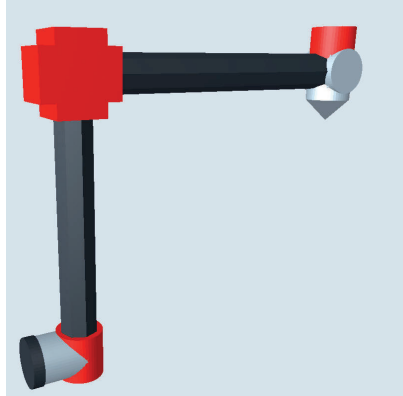
MIXED\_LRR\_ZXA\_M10



9308658315

Kinematics with 1 linear and 2 rotative axes that allows motions of the flange on the surface of a cylinder around the linear axis (vertical Z, horizontal X) and the rotation of the flange (A).

MIXED\_RLLR\_XYZA\_M10



9308673931

Kinematics with 1 rotative axis in the socket, 2 linear axes and 1 rotative axis at the flange.

## 8.7 USER kinematic model

Additional kinematic models (such as quadropod, hexapod, articulated, none, etc., through to special designs) can be integrated as a customized kinematic model. For details, refer to the "Integrating a customized kinematic model" (→ 135) chapter.

If you require additional information on special kinematic models, contact SEW-EURODRIVE.

## 9 Interpolating operating modes

This chapter first provides an overview of all interpolating operating modes. These operating modes are then described in detail and the operating modes are compared for TARGET and continuous path blending.

Finally, you learn about the applications and advantages of combining the two operating modes.

### 9.1 Overview

The following list shows the operating modes and their application:

Jog modes	If you want to move axes within Cartesian degrees of freedom (DOF). A prerequisite for this is that the relevant control signal (command bit) is set.
TARGET positioning	When you want to move to a target position axis-wise or using Cartesian coordinates.
Continuous path interpolation	When you travel through a sequence of path segments (that are generally located geometrically in space).

The following table provides an overview of the functions and features of these operating modes:

Operating modes		Functions	Features
Jog modes	KIN_JOG_AXIS	Jogging of single axes (kinematic and auxiliary axes).	
	KIN_JOG_CART	Jogging of the TCP along the coordinate axes (X, Y, Z, A, B, C) in the currently set coordinate system. Auxiliary axes can be jogged simultaneously.	
TARGET positioning	KIN_TARGET_AXIS (→ 71)	Positioning of single axes.	<ul style="list-style-type: none"> <li>Single axes and Cartesian degrees of freedom can be synchronized.</li> <li>The target position can be specified axis-wise or as Cartesian coordinates in both operating modes</li> </ul> (for details, refer to the "Changing the coordinate system" (→ 124) chapter).
	KIN_TARGET_CART (→ 72)	Positioning of the Cartesian degrees of freedom of the tool center point (TCP), such as X, Y, Z.	

Operating modes		Functions	Features
Continuous path interpolation	KIN_LIN_XY/ _YZ/ _ZX (→ 75)	Motion of the tool center point (TCP) along a straight line in a principal plane.	<ul style="list-style-type: none"> <li>Path segments are accepted by the kinematic control system and entered in a queue.</li> <li>The blending parameterized in the segments is carried out automatically by the kinematic control system.</li> </ul>
	KIN_LIN_3D (→ 75)	Motion of the tool center point (TCP) along a straight line in the 3D space.	
	KIN_CIRC_XY/ _YZ/ _ZX (→ 75)	Motion of the tool center point (TCP) along a circle.	

## 9.2 Jog mode

### 9.2.1 Specifying direction and kinematic quantities

**Direction specification:** In jog mode, the direction is specified in the *AxisGroupKin.Inst[.].In.Jog.Positive* and *AxisGroupKin.Inst[.].In.Jog.Negative* input signals by setting the corresponding array elements.

If the same array element is set in both variables, no movement will take place.

**Kinematic quantities:** The required jog speed can be varied cyclically using the *AxisGroupKin.Inst[.].In.Jog.VelocityPercentage* input signal. If the percentage is set to the value "100", the controller travels at the configured speed as far as possible.

## 9.3 TARGET modes

### 9.3.1 Principle

In TARGET modes, continuous-path control cyclically adopts new target values (TARGET interpolation).

The modes **KIN\_TARGET\_AXIS** and **KIN\_TARGET\_CART** determine how movement is made to the target point, i.e. if it is interpolated in the axis space or in the Cartesian space. This means the type of motion is at the focus.

The target point can however be specified in both operating modes by means of axis values (ACS) or by means of Cartesian coordinates (such as KCS). The kinematic controller converts the target coordinates automatically into target axis values on request (in case of **KIN\_TARGET\_AXIS**) and into Cartesian target coordinates (in case of **KIN\_TARGET\_CART**).

A detailed overview of the possible combinations of operating mode and target specifications is given in the "Changing the coordinate system" (→ 124) chapter.

The programming of TARGET modes is similar to the usual programming of single axes. Users of motion controls of individual axes (for example with MultiMotion) therefore come to terms quickly and can program complex kinematic models through to 6-axis articulated arm robots just as easily as single axes.

The target values are entered in the following input signal:

- AxisGroupKin.Inst[.].In.Target.Position*

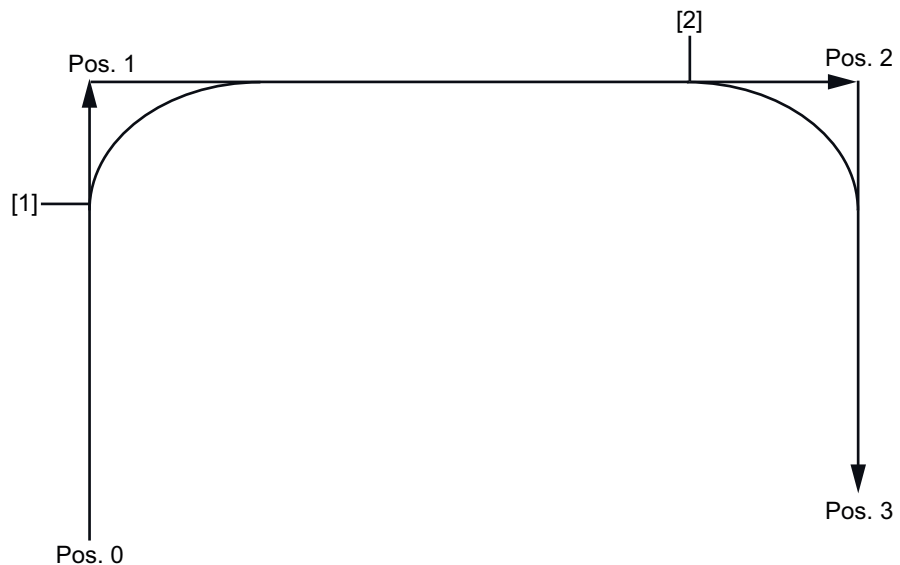


The kinematic quantities are entered in the following input signals:

- *AxisGroupKin.Inst[...].In.Target.Axis.*
- *AxisGroupKin.Inst[...].In.Target.Cart.*

### 9.3.2 Example: Pick-and-place

The following figure shows the blending of targets (TARGET blending) in the trajectory of a pick and place application:



9586252939

[1] Changeover to the target of position 2

[2] Changeover to the target of position 3

**Procedure:** The *AxisGroupKin.Inst[n].In.Target.Position* input variable is assigned the target values (positions) one after the other:

1. To start with, the target value corresponds to position 1.
2. When approaching position 1, the target value is changed to position 2.
3. When approaching position 2, the target value is changed to position 3.

The blending of targets (TARGET blending) is initiated in the user program.

Cartesian gantries are usually programed in this way for handling tasks.

Motion does not take place axis-wise in **KIN\_TARGET\_CART** mode. Instead, the Cartesian coordinates (X/Y/Z/A/B/C) of the tool center point (TCP) are moved along the Cartesian axes ((X/Y/ ...), first to position 1, then to position 2, etc.

For this purpose, the type of kinematic model (gantry kinematics, SCARA, articulated arm robot) is not relevant.

The kinematic control system automatically calculates and controls the necessary axis paths. The control of the user-programmed positioning of the X/Y/Z... coordinates of the tool coordinate system (TCS) takes place in 3D/6D space.

### 9.3.3 Advantages

In the usual programming of a Cartesian gantry, the kinematic quantities for moving the axes are set in the inverters. This means the paths result from the simultaneous movement of axes that are independent from one another but are coordinated with respect to time.

Similarly, with TARGET interpolation, kinematic quantities that are separate from one another are set for each dimension (axes for **KIN\_TARGET\_AXIS**, Cartesian for **KIN\_TARGET\_CART**) (input signals *AxisGroupKin.Inst[...].In.Target.Axis/AxisGroupKin.Inst[...].In.Target.Cart*).

The path of the tool center point (TCP) results from the simultaneous controlling of the profile generators to the currently entered target coordinates for each dimension involved. In this way, you can create harmoniously and fast paths that are extremely suited for handling tasks, for example. The user can explicitly set the maximum permissible kinematic quantities for each direction of movement.

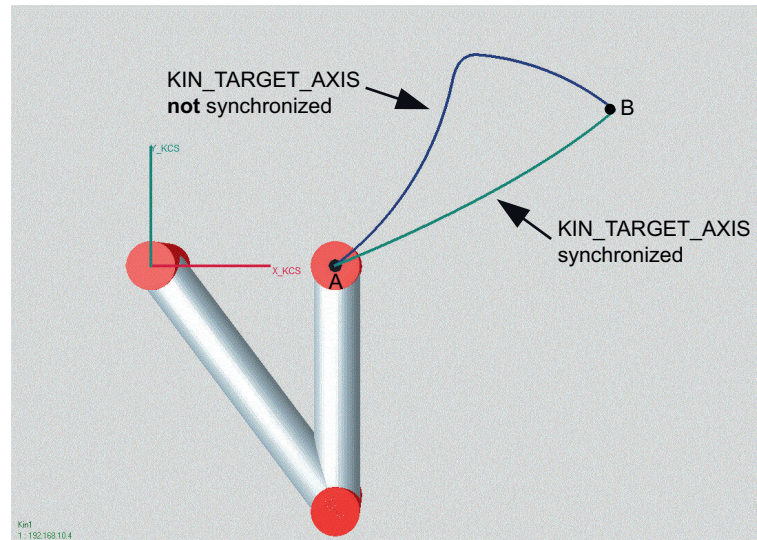
TARGET interpolation has the following advantages compared to the usual programming of Cartesian gantries:

- Same familiar path programming both for Cartesian gantries and complex kinematics.
- Option of synchronization of the dimensions involved in the motion so that all components generally reach the target simultaneously. For rest-to-rest motions, almost linear movements can therefore also result when the motion parameters are set uniformly for the kinematic quantities (*AxisGroupKin.Inst[...].In.Target.Synchronization quantities* input signal).

### 9.3.4 KIN\_TARGET\_AXIS

The **KIN\_TARGET\_AXIS** mode allows axis-wise movement to the target axis values. Each axis are assigned own kinematic quantities for this purpose.

The following figure shows 2 possible paths of the tool center point (TCP) when moving a SCARA kinematics between the points A and B:



9308521483

The axis profile is not synchronized for the upper arc.

For the lower, uniform arc, the axis profiles are synchronized (*AxisGroup-Kin.Inst[...].In.Target.Synchronization[0] = TRUE*).

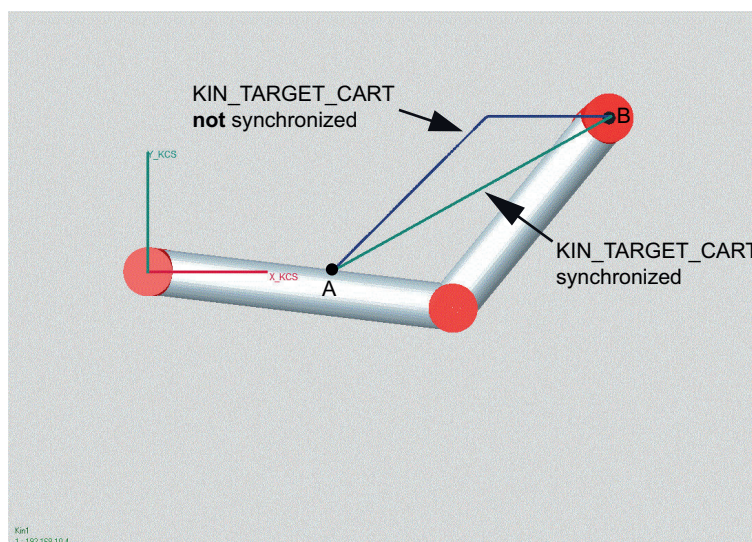
No straight line is obtained in this kinematic model in spite of synchronization in space. Each axis moves to the target axis values to reach the target point in space. All axes reach the target axis values simultaneously in case of preset synchronization.

### 9.3.5 KIN\_TARGET\_CART

The **KIN\_TARGET\_CART** mode allows for TCS movement along the coordinate axes (X, Y, Z, A, B, C) in the currently set coordinate system to the Cartesian target coordinates. Each Cartesian coordinate are assigned own kinematic quantities for this purpose.

At the same time you can move the auxiliary axes axis-wise to the target axis values.

The following figure shows 2 possible paths of the tool center point (TCP) when moving a SCARA kinematics between the points A and B:



9308533643

The Cartesian profiles are not synchronized for the upper path.

For the lower, straight path, the Cartesian profiles (motion along X, Y) are synchronized (*AxisGroupKin.Inst[...].In.Target.Synchronization[0] = TRUE*, all degrees of freedom, same jerk time *AxisGroupKin.Inst[n].In.Target.Cart.Jerk[i]*). For preset synchronization, the rest-in-rest motion results in a line in this case.

Without synchronization, every partial profile (X, Y) moves to the Cartesian target coordinates. During the non-synchronized motion from bottom left to top right, the Y coordinate is reached earlier, so that the subsequent movement is only along X to the target coordinate.

### 9.3.6 TARGET blending

#### Principle

With TARGET modes, blending takes place by checking the desired blending distance to the corner point in the user program and the corresponding update of the target specification on entering the blending area. This means that as soon as the tool is adequately near the target point, a new target point is specified to which the motion is automatically directed by the kinematic controller.

#### Library and input signals

You can use the "MC\_KinInProximity" function from the "MPLCKinematics library" (→ 114) for checking the distance to the current target point, and use it as a transition condition in a sequence. In the steps, the new target positions and target orientations are written to the *AxisGroupKin.Inst[n].In.Target.Position* input variable.

The threshold input signal of the "MC\_KinInProximity" function represents the distance from the target point. If this point is exceeded, the function will return the value FALSE. If the distance is exactly as defined in "Threshold" or if the distance is smaller, the value TRUE will be returned.

In the "Dimension" input signal, you specify the Cartesian dimensions for determining the distance.

#### Calling the function for translation and rotation

If you want to check both the translatory and rotatory distance, calling the function only suffices when "Threshold = 0". Otherwise, you have to call the function separately for the translation and the rotation. The reason is that the translational distance (for example 10 mm) and the rotative distance (for example 15°) cannot be combined in one value for the "Threshold" input.

The following examples show the function calls for translation and rotation:

- **Translation**

Check = 2#0000\_0xxx (for example 2#0000\_0111)

The X, Y and Z coordinates are considered for determining the distance (for currently configured Cartesian coordinate system).

- **Rotation**

Check = 2#00xx\_x000 (for example 2#0000\_1000)

Only the A coordinate, which means the rotation around the Z axis, is included in the determination of (rotational) distance (for currently configured Cartesian coordinate system).

## 9.4 Continuous path operating modes

### 9.4.1 Principle

In contrast to TARGET interpolation, a processing list, the so-called queue, is filled with path segments for continuous path (CP) interpolation. The path segments entered in the queue are automatically traveled through, corresponding to their parameterization and the currently set processing speed in percentage, when feed enable is active.

The continuous path segments are entered in the *AxisGroupKin.Inst[.].In.Cp.Segment* input signal. A variable ID (*AxisGroupKin.Inst[.].In.Cp.Segment.ID*) is used as a signal for the kinematic control system to enter a continuous path segment into the queue.

For example, unambiguous IDs can be assigned to each CP segment in an application, or the ID is just always incremented. For this purpose, simply changing between "0" and "1" would be sufficient. However, we do not recommend this method because hardly any conclusion can then be drawn from the *AxisGroupKin.Inst[.].Out.Cp.Queue.ActSegID* output signal concerning the currently processed CP segment.

Output signals (example) of the structure variables *AxisGroupKin.Inst[.].Out.Cp*:

Output signals	Meaning
<i>Queue.Size</i>	Current number of CP segments that have not yet been processed.
<i>Path.Distance</i>	The distance still to be covered to reach the target point of the last entered segment.
...	...

When the *AxisGroupKin.Inst[.].In.General.Kin.FeedEnable* input signal is set, the control system begins the processing directly when the first CP segment is entered. This means the required setpoints are calculated and sent to the inverters already in the first cycle.

Another CP segment can be entered in the queue with each subsequent cycle. Likewise, further CP segments can be entered till the last processing cycle of a path so that the motion is directly accelerated to move the attached CP segment without disruption.

The CP queue is implemented as a ring buffer. Therefore, CP segments can be traveled endlessly without having to stop. The only limitation is the total path length from a point in time at which the queue was completely processed and was therefore empty.

The total path length must not exceed the limit of a maximum value represented by a 64-bit floating point number. This corresponds to a value of > 1000.000.000.000.000.000 user units.

The default value of the CP queue size is 256 elements (MOVI-PLC® power: 2048 elements) that can be entered during processing. It is important that, for example with LIN -> LIN or CIRC - LIN blending, one blending curve each is automatically entered and the remaining straight segment up to the target is entered. This means in the example that usually two queue elements are assigned per LINEAR segment to be entered. When only LINEAR segments are entered, more than 100 segments can be provided at once for processing in the queue. The kinematic quantities for every CP segment are entered individually ( *AxisGroupKin.Inst[.].In.Cp.Segment* input signal). While the queue is being processed, the movement can additionally be slowed down or accelerated across the segments (*AxisGroupKin.Inst[n].In.Cp.Path.VelocityPercentage* input signal).



**9.4.2 KIN\_LIN\_XY / \_YZ / \_ZX**

The **KIN\_LIN\_XY / \_YZ / \_ZX** modes allow to geometrically exactly describe linear interpolation of the tool center point (TCP) in the main axes of the currently set coordinate system.

Enter the target coordinates in the *AxisGroupKin.Inst[.].In.Cp.Segment.Position* input variables. When doing so, the coordinates that are not used in the selected operating mode (such as Z coordinate for **KIN\_LIN\_XY**) are not read from the input signal, but are taken from the value that is programmed or reached last.

**9.4.3 KIN\_LIN\_3D**

The **KIN\_LIN\_3D** mode allows to geometrically exactly describe the linear interpolation of the tool center point (TCP) in the 3-dimensional space in the currently set coordinate system.

Enter the target coordinates in the *AxisGroupKin.Inst[.].In.Cp.Segment.Position* input variables.

**9.4.4 KIN\_CIRC\_XY / \_YZ / \_ZX**

The **KIN\_CIRC\_XY / \_YZ / \_ZX** modes allow circular interpolation of the tool center point (TCP) in the main axes of the currently set coordinate system. The circle segment is parameterized depending on the selected *AxisGroupKin.Inst[.].In.Cp.Segment.Circ.Mode* variable (see "In.Cp.Segment" (→ 98) chapter), for example by means of the circle center and the circle angle to be covered.

If a circular arc is interpolated in an inclined level, it can be changed to a workpiece coordinate system "KIN\_PCSn" by means of TARGET interpolation, for example, and then a circle segment "KIN\_CIRC\_XY" can be programmed in it. You can set the required inclination in the *AxisGroupKin.Inst[.].In.Transform.WCS\_PCSn* transformation (see the "In.Transform" (→ 92) chapter).

#### 9.4.5 Continuous path blending

Contrary to TARGET modes, blending with continuous path is not initiated by a distance check and corresponding step-wise input of new target positions.

Instead, the programming for blending with continuous path modes takes place through parameterization of the *AxisGroupKin.Inst[.].In.Cp.Segment.Blending* structure for which blending should take place. The kinematic controller automatically starts the blending motion to the respective, subsequent segment.

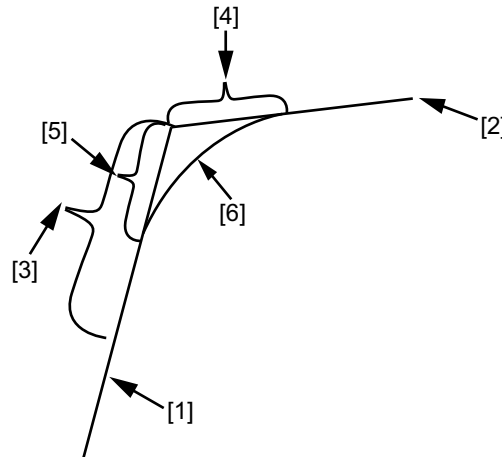
Input signals of the *AxisGroupKin.Inst[.].In.Cp.Segment.Blending* structure variable:

Input signal	Function
<i>Blending.Distance</i>	Specifies the distance from the last target point from which blending should be carried out to the new target point, including the radius of the spherical blending zone.
<i>LimitationPercentage</i>	Percentage for automatically reducing the programmed <i>Blending Distance</i> .  The percentage entered refers to the distance between the last target point (that is blended) and the target point of the segment that is transferred together with the blending parameters in the same cycle.
<i>Blending.Velocity</i>	Velocity profile and percentage scaling used for traveling along the blending curve (see chapter "In.Cp.Segment", Blending input signal (→ 99)).



### Example: Blending LIN -> LIN

The following figure shows blending from a LINEAR segment to another LINEAR segment:



9735212939

- [1] Last LINEAR segment accepted by the kinematic controller
- [2] Target point *Position* of the new LINEAR segment (in currently valid AxisGroup-Kin.Inst[...].In.Cp.Segment variable structure)
- [3] Parameterized *Blending.Distance* (in currently valid AxisGroup-Kin.Inst[...].In.Cp.Segment variable structure)
- [4] *Blending.LimitationPercentage* of the length of the new straight segment (in currently valid AxisGroupKin.Inst[...].In.Cp.Segment variable structure)
- [5] Resulting blending distance Corresponds to the parameterized *Blending.Distance* [3]. However, the blending distance is limited first by the remaining distance to the last LINEAR segment [1] at the time when the LINEAR segment is accepted. And secondly limited by the *.LimitationPercentage* [4] of the length of the new straight segment [2].
- [6] Resulting symmetrical blending curve

### 9.5 Comparison of continuous path blending and TARGET blending

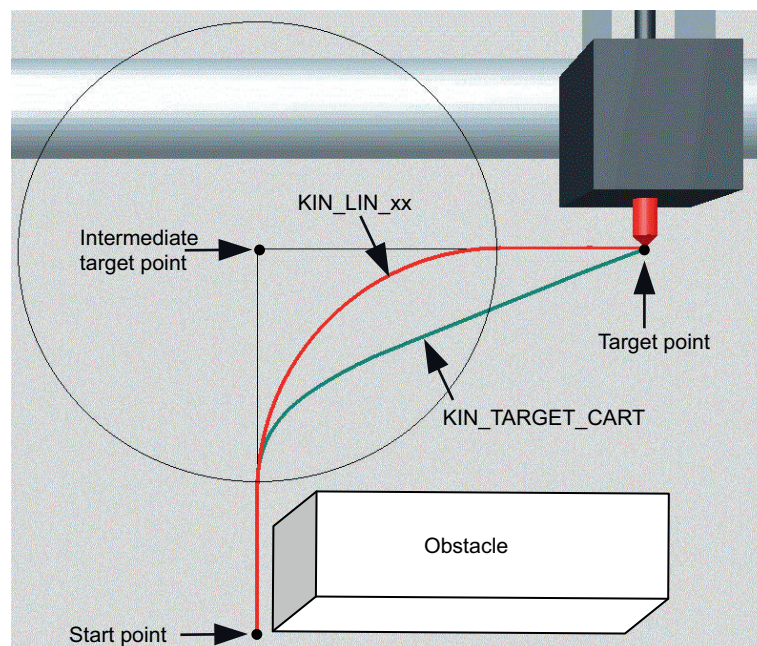
For continuous path interpolation, a symmetric blending curve is inserted during blending from one segment to another. This means the blending curve begins on entering the circular blending zone and ends on leaving it. The programmed segment is therefore reached after blending, as if blending has not been carried out.

Contrary to continuous path blending, a new target point is specified for TARGET interpolation on entering the blending zone and on exceeding the specified distance to the intermediate target point. From this time, the motion from the current position and the current motion status is led to the target point with the set motion parameters. The path to the target point depends on the motion parameters and the synchronization setting.

It could be a coincidence with TARGET interpolation when the path directly proceeds in a straight line before reaching the target point, through which the target point is reached with linear interpolation.

Instead, with TARGET interpolation (as compared to continuous path interpolation), the target point is turned to depending on the motion parameters and the “strong” or “weak” synchronization setting as well as with another blending characteristic.

The following figure shows an example of the blending characteristic for TARGET blending compared to continuous path blending:



9308546187

The target straight segment of the TARGET interpolation (in the example implemented with **KIN\_TARGET\_CAR**) is **underneath** the target straight segment of the continuous path interpolation (in the example implemented with **KIN\_LIN\_xx**).

Depending on the setting, also the opposite case is possible, which means the straight target straight segment of the TARGET interpolation runs **above** the target straight segment of the continuous path interpolation.

#### Conclusion

TARGET interpolation is perfectly suited for implementing harmonious paths with short cycle times as they occur in handling processes, for example.



### ▲ WARNING

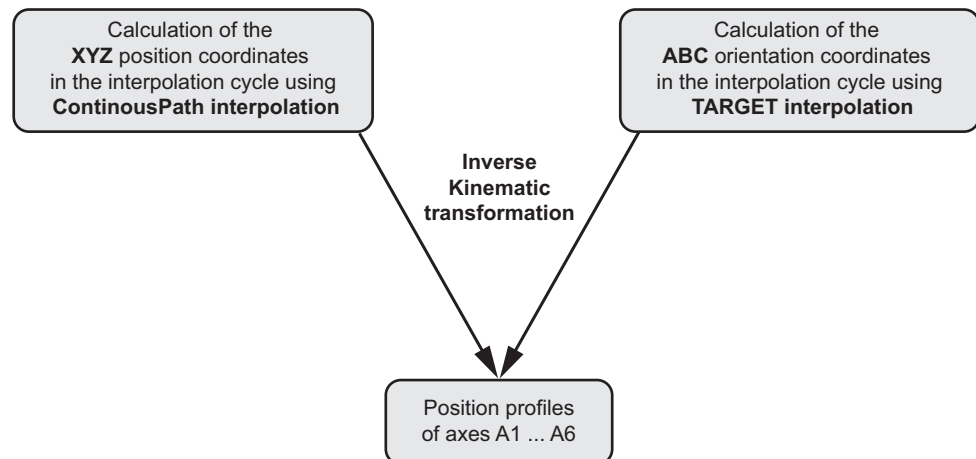
Various profiles result from changing motion parameters or from stopping and re-starting.

- Use TARGET interpolation only when the resulting path is free from obstacles. This condition applies to a predominant number of handling, palletizing, depalletizing, and tracking applications so that TARGET interpolation can be used well for these applications.

## 9.6 Combining continuous path modes and TARGET modes

You can simultaneously carry out translational motion of the tool center point (TCP) by means of continuous path interpolation and orientation changes of the TCS by means of a TARGET interpolation (for example with **KIN\_TARGET\_CART**).

The following figure shows how to combine the differently interpolated coordinates to form a common position profile by using inverse kinematic transformation:



9290278795

This combined operating mode is set when changing to a continuous path operating mode in *AxisGroupKin.Inst[...].In.Cp.Settings.PlusTargetCartABC* (see the "In.Cp.Settings" (→ 98) chapter).

The output signal *AxisGroupKin.Inst[...].Out.Cp.Settings.PlusTargetCartABC* shows if the combined operating mode is active (see the "Out.Cp" (→ 111) chapter).

This combination is useful in applications, for example, with a CP path consisting of a great number of segments where only the orientation in the first and last segment is relevant. In this case, specifying the orientations in intermediate points is not necessary. Therefore, the target orientation is written to the *AxisGroupKin.Inst[...].In.Target.Position* input variable in the user program once movement along the first CP segment is completed. The last CP segment must not be entered into the queue until the user program detects that the target orientation has been reached (see the "MC\_KinInProximity" function (→ 114) chapter).

If a specific target orientation is to be reached at the end of every CP segment, then the subsequent CP segment may not be entered into the CP queue and the next target orientation may not be written into the *AxisGroupKin.Inst[...].In.Target.Position* input variable until the current target orientation has been reached.

If the following position and orientation are assigned some time before reaching the current target orientation, a blending of the orientation is implemented.

You can use the "MC\_KinInProximity" function for this purpose:

- Threshold input = orientation blending distance
- Dimension input = 2#0011\_1000 (for example for checking ABC)

The blending of the translational path of the tool center point (TCP) is also specified in *AxisGroupKin.Inst[.].In.Cp.Segment.Blending* in this case.

## 10 MOVI-PLC® program

### 10.1 Task configuration

A suitable task configuration is automatically set up when creating a new PLC project, see chapters "Step 2: Create a MOVI-PLC® project (→ 27)" and "Step 3: Integrate the technology module (→ 28)". The task items listed below are required for the kinematics technology module. When the PLC project is not newly created but only needs to be expanded, you can also add these task items manually.

- TaskMain

→ **PRG\_TaskMain\_Kinematics();**

before AxisHandler\_Main\_MultiMotion / \_Light();

This program runs among others the following programs from the libraries:

- AxisGroupControl\_Configuration\_Kinematics(Enable:= TRUE);
- AxisGroupControl\_Main\_Kinematics(Enable:= TRUE);

- TaskPriority

→ **PRG\_TaskPriority\_Kinematics();**

before AxisHandler\_Priority\_MultiMotion / \_Light();

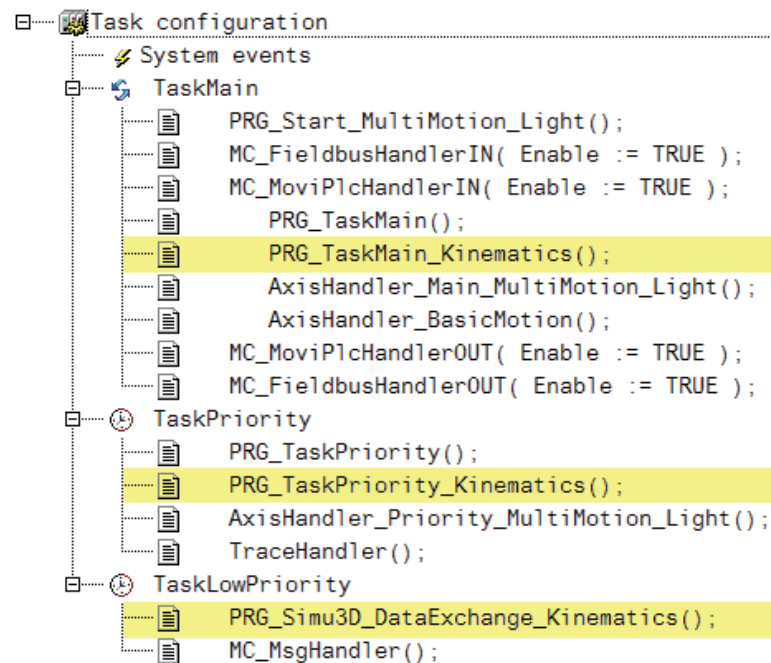
This program runs the following programs from the libraries:

- AxisGroupControl\_Priority\_Kinematics();

- TaskLowPriority

→ **PRG\_Simu3D\_DataExchange\_Kinematics();**

This program is included in the libraries.



9310191755

## 10.2 User program

The user program writes the values that are required for sequences and motion tasks, such as the operating mode as well as motion parameters, to the input variables *AxisGroupKin.Inst[.].In*, described in chapter "AxisGroupKin.Inst[.] variable interface (→ 89)".



### ⚠ WARNING

To ensure that the variables used for a motion task (such as target position, velocity, acceleration, etc.) can be consistently used through the kinematic controller, the user program **MUST** be executed in the same task in which the *AxisGroupControl\_Main\_Kinematics* program runs. This is usually the TaskMain.

Violating this prerequisite can lead to inconsistent motion tasks and consequently to unpredictable movements of the kinematics.

**Exception:** The following variables must be described in TaskPriority for consistency. This is the only way to cyclically apply the continuous profiles, which are used directly for motion control, in the interpolation cycle.

- *AxisGroupKin.Inst[.].In.Transform* see chapter "In.Transform (→ 92)"
- *AxisGroupKin.Inst[.].In.MasterPosition* see chapter "In.MasterPosition (→ 93)"
- *AxisGroupKin.Inst[.].Config.Kin.Par*, if *AxisGroupKin.Inst[.].Config.CyclicTakeover.Kin\_Par = TRUE*



### ⚠ WARNING

Even during consistent takeover of the input variables through the kinematic controller, which is guaranteed by the system, there are situations in which explicit checks must be made in the user program (usually TaskMain) before switching to the program run to see whether the kinematic controller (usually TaskPriority) has been implemented since the last execution cycle of the user program.

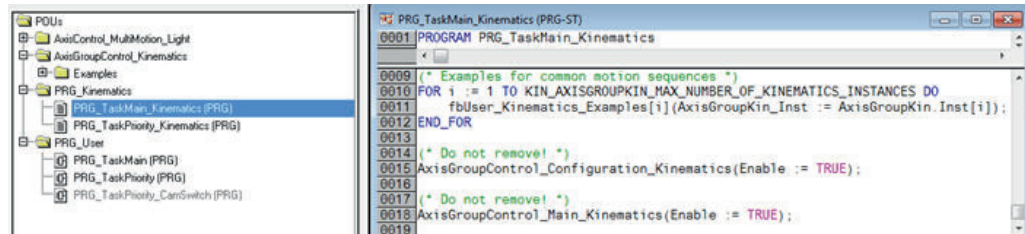
This check is required, for example, when path segments are to be entered in the kinematic controller step-wise through the user program during continuous path interpolation. The segment parameters are consistently accepted through the kinematic controller. If, however, the TaskPriority of the user program was not carried out between 2 cycles, and a new path segment was parameterized in the user program without checking the segment being accepted in each cycle, then some path segments might not be accepted by the kinematic controller.

In this situation, it must be checked in the user program whether the last parameterized path segment has been accepted, see chapters "Continuous path programming (→ 87)" and "Out.Cp (→ 111)".



## 10.3 Sample programs

When you create a new PLC project according to chapters "Step 2: Create a MOVI-PLC® project (→ 27)" and "Step 3: Integrate the technology module (→ 28)", you can refer to the folder `AxisGroupControl_Kinematics/Examples` for examples of simple user programs. These sample programs are called in the `fbUser_Kinematics_Examples` function blocks in the program.



9310150923

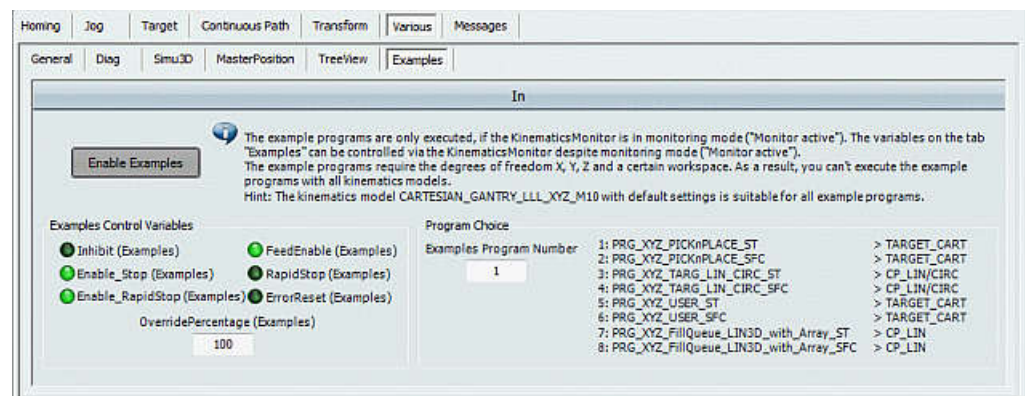
Instead, you can execute your user program, for example, in `PRG_TaskMain` in the function block folder, `PRG_User` or at another location in the TaskMain.

You can program the user program in each of the available IEC (International Electro-technical Commission) languages. The graphical sequential function chart (SFC) or structured text (ST), for example, are particularly suitable for implementing the required sequences.

To ensure that the programming examples included in the newly created PLC project are implemented, you must describe the following variables appropriately.

<code>gKinExamples_Control[..].EnableExamples</code>	TRUE
<code>gKinExamples_Control[..].ProgramNumber</code>	1..8
<code>gKinExamples_Control[..].Inverter.Inhibit_AllAxis</code>	FALSE
<code>gKinExamples_Control[..].Inverter.Enable_Stop_AllAxis</code>	TRUE
<code>gKinExamples_Control[..].Inverter.Enable_RapidStop_AllAxis</code>	TRUE
<code>gKinExamples_Control[..].Kin.FeedEnable</code>	TRUE
<code>gKinExamples_Control[..].Kin.RapidStop</code>	FALSE
<code>gKinExamples_Control[..].Kin.OverridePercentage</code>	1..100

You can easily switch these variables in the "Various/Examples" tab of the kinematic monitor.



12566156299



## INFORMATION

Ensure that your application program does **NOT** start before *AxisGroupKin.ConfigDataAvailable* = TRUE. Else, the configuration data and default assignments for the input variables are not yet available and will possibly be overwritten when your application program writes these variables in advance (for example override).

In the sample programs, the *fbUser\_Kinematics\_Examples* check is carried out in the Examples folder.

---



### 10.3.1 TARGET programming

When using the TARGET interpolation, new target variables are moved to directly after connection to the corresponding input variables (see chapter "In.Target (→ 95)"). This is why a check is required in the user program to see whether a point to be moved to or its vicinity has been reached before the target coordinates of the next point are assigned. The *MC\_KinInProximity* function is used for this check, see chapter "MC\_KinInProximity function (→ 114)".

**TARGET Sample program**

...

CASE (MotionSequenceState) OF...

10: (\* Lifting \*)

(\* General settings, such as operating mode, enable \*)

AxisGroupKin.Inst[1].In.General.Mode := KIN\_TARGET\_CART; (\* Mode\*)

AxisGroupKin.Inst[1].In.General.Inverter.Enable\_RapidStop[0] := ...; (\* Rapid stop enable \*)

AxisGroupKin.Inst[1].In.General.Inverter.Enable\_Stop[0] := ...; (\* Stop enable \*)

AxisGroupKin.Inst[1].In.General.Kin.FeedEnable := ...; (\* Kinematic feed enable \*)

(\* Target parameterization \*)

AxisGroupKin.Inst[1].In.Target.Cart.Vel./Acc./Dec./Jerk[ ...] := ...; (\* Kinematic quantities  
for all dimensions \*)

AxisGroupKin.Inst[1].In.Target.Synchronization[...] := ...; (\* Synchronization of X,Y = 1,2 for example \*)

AxisGroupKin.Inst[1].In.Target.Position := ...; (\* Target position 1 \*)

AxisGroupKin.Inst[1].In.Target.CoordSys := ...; (\* Coordinate system \*)

(\* Switching to program run as soon as motion has sufficiently advanced \*)

IF MC\_KinInProximity(...) THEN (\* Distance check, see chapter "MC\_KinInProximity function" \*)

MotionSequenceState := 20;

END\_IF

(\* \*\*\*\*\* \*)

20: (\* Transversal \*)

AxisGroupKin.Inst[1].In.Target.Position := ...; (\* Target position 2 \*)

IF MC\_KinInProximity(...) THEN (\* Distance check \*)

MotionSequenceState := 30;

END\_IF

(\* \*\*\*\*\* \*)

30: (\* Lowering \*)

AxisGroupKin.Inst[1].In.Target.Position := ...; (\* Target position 3 \*)

IF MC\_KinInProximity(...) THEN (\* Distance check \*)

MotionSequenceState := 40;

END\_IF

(\* \*\*\*\*\* \*)

40: (\* ... \*) ...

END\_CASE

...

12511008523

(see chapter Example: Pick-and-place (→ 69))

### 10.3.2 Continuous path programming

For CP interpolation, the path segments are supplied one after the other to the kinematic controller. One CP segment can be supplied per TaskMain cycle. If the parameter check was successful, the kinematic controller accepts the respective segment and enters it in a queue. The kinematic controller then automatically carries out path interpolation and parameterized blending between the individual segments entered in the queue. Contrary to TARGET interpolation, blending is not initiated in the user program.

**ContinuousPath Sample program**

```

...
CASE (ProgramState) OF...
10: (* Supply lifting segment *)

    (* General settings, such as operating mode, enable *)
    AxisGroupKin.Inst[1].In.General.Mode           := KIN_LIN_3D;  (* mode *)

    AxisGroupKin.Inst[1].In.General.Inverter.Enable_RapidStop[0] := ...; (* Rapid stop enable *)
    AxisGroupKin.Inst[1].In.General.Inverter.Enable_Stop[0]     := ...; (* Stop enable *)
    AxisGroupKin.Inst[1].In.General.Kin.FeedEnable              := ...; (* Kinematic feed enable *)

    (* ContinuousPath parameterization *)

    AxisGroupKin.Inst[1].In.Cp.Segment.Translation.Vel./Acc./Dec./Jerk := ...; (* Kinematic of the translation *)
    AxisGroupKin.Inst[1].In.Cp.Segment.Blending.Distance              := ...; (* Blending distance *)
    AxisGroupKin.Inst[1].In.Cp.Segment.Blending.LimitationPercentage := 50; (* Blending percentage *)

    AxisGroupKin.Inst[1].In.Cp.Segment.Position := ...; (* Target position 1 *)
    (* Change ID, for example increment as a signal to the kinematic controller to accept the segment *)
    AxisGroupKin.Inst[1].In.Cp.Segment.ID := AxisGroupKin.Inst[1].Out.Cp.Queue.LastMappedSegID + 1;

    ProgramState:= 20;
    (*****)
20: (* Supply transversal segment *)

    (* Switching to filling the ContinuousPath queue
    The segment or segments transferred to the kinematic controller generally have not been started at this point of time,
    or have been started only partially, meaning the robot may still be in motion. *)

    IF (AxisGroupKin.Inst[1].Out.Cp.Queue.LastMappedSegID = AxisGroupKin.Inst[1].In.Cp.Segment.ID) THEN
        (* Segment was accepted → supply next segment, see also warning in chapter "User program" *)

        AxisGroupKin.Inst[1].In.Cp.Segment.Position := ...; (* Target position 2 *)
        AxisGroupKin.Inst[1].In.Cp.Segment.ID := AxisGroupKin.Inst[1].In.Cp.Segment.ID + 1; (* Change ID *)
        ProgramState:= 30;
    END_IF
    (*****)
30: (* Supply lowering segment *)

    IF (AxisGroupKin.Inst[1].Out.Cp.Queue.LastMappedSegID = AxisGroupKin.Inst[1].In.Cp.Segment.ID) THEN
        (* Segment was accepted → supply next segment *)

        AxisGroupKin.Inst[1].In.Cp.Segment.Position := ...; (* Target position 3 *)
        AxisGroupKin.Inst[1].In.Cp.Segment.ID := AxisGroupKin.Inst[1].In.Cp.Segment.ID + 1; (* Change ID *)
        ProgramState:= 40;
    END_IF
    (*****)
40: (* Wait to reach end of path.
    Till then, AxisGroupKin.Inst[1].In.General.Mode must be a CP mode. This means LIN and CIRC segments can also be
    supplied; these are then automatically traveled to by the kinematic controller. However, as soon as another mode is
    applied, such as KIN_STOP/JOG/TARGET, AM_DEFAULT, AM_HOMING, the content of the CP queue is deleted. *)

    IF (AxisGroupKin.Inst[1].Out.Cp.Queue.LastMappedSegID = AxisGroupKin.Inst[1].In.Cp.Segment.ID)
        (* Segment was accepted *)
    AND (AxisGroupKin.Inst[1].Out.Cp.Path.EndReached) THEN
        (* Path traveled through completely *)
        ProgramState:= ...; (* and so on *)
    END_IF
END_CASE

```

12511013003

## 11 AxisGroupKin.Inst[.] variable interface

The complete control of all kinematics instances takes place via the global structure variable, AxisGroupKin, that is declared in the GlobalVar\_AxisGroupControl\_Kinematics function block in the AxisGroupControl\_Kinematics folder.

Variable	Type	Meaning			
HMI	ST_AxisGroupControl_HMIInterface (03_ApplicationModules \ MPLCAxisGroupControl.lib)	This variable must NOT be defined by the user. This variable indicates the state of the HMI control mode in the kinematic monitor.  <b>Note:</b> When the control should take place exclusively from the kinematic monitor, then the application program may NOT be executed and the global variable AxisGroupKin may NOT be defined when HMI.HMIControl = TRUE (see also the examples imported with the technology module "Kinematics", PRG_User_Kinematics_Examples).			
		Detailed information			
		HMIControl	BOOLEAN	HMI control mode is activated in the monitor	
		WatchDogIn	BOOLEAN	Mirror from engineering PC	
		WatchDogOut	BOOLEAN	Toggle bit from MOVI-PLC®	
		WatchDogTimeout	BOOLEAN	Watchdog timeout	
		WatchDogTimeoutTime	DWORD	Timeout time [ms]	
NumberOfInstances	UINT	Number of configured and consequently activated kinematics instances (see also variable <i>AxisGroupKin.Inst[.].Config.Enable</i> ).			
ConfigDataAvailable	BOOLEAN	The configuration data and default values for input variables for all kinematics instances and all axes were read from the memory card and the corresponding variables were assigned.  <b>Note:</b> As long as <i>ConfigDataAvailable</i> = FALSE, the application should NOT be executed (see chapter "Sample programs (→ 83)").			
Inst	ARRAY [1..] OF ST_AxisGroupKin_Instance (03_ApplicationModules \ MPLCAxisGroupControl_Kinematics.lib)	Configuration, input, output and HMI data of the kinematics instances.			
		The following variables are available for each instance:			
		Config	ST_AxisGroupKin_Config	Automatically defined by the system	
		In	ST_AxisGroupKin_In	Kinematic control variables (see below)	
		Out	ST_AxisGroupKin_Out	Kinematic display variables (see below)	
		HMI	ST_AxisGroupKin_HMIGeneral	HMI variables, used by the monitor	

### INFORMATION



Kinematic control is activated exclusively by means of the "AxisGroupKin" global variable interface.

The user and the application program may NOT define the "AxisInterface" variable interface of the MultiMotion axes belonging to the kinematics.

## 11.1 In.General

Type: MC\_KIN\_IN\_GENERAL (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning		
Inverter	MC_KIN_IN_GENERAL_INVERTER (04_InterpolatedMotion\MPLCKinematics.lib)	Inhibit	ARRAY[0..8] OF BOOL	Controller inhibit
		Enable_RapidStop	ARRAY[0..8] OF BOOL	MultiMotion enable/rapid stop
		Enable_Stop	ARRAY[0..8] OF BOOL	MultiMotion enable/stop
		Bits 1 to 8 correspond to axes 1 to 8. Bit 0 corresponds to all configured axes of this kinematics instance. <b>Note:</b> When Enable_RapidStop or Enable_Stop is set to FALSE for at least one configured axis of a kinematics instance, then all axes of this instance are not guided kinematically any more, instead they are decelerated via MultiMotion by means of a stop profile. This means that deceleration is usually NOT true to path. <b>Exception:</b> In KIN_JOG_AXIS mode, even individual kinematic axes can be enabled and jogged, independent of the status "referenced".		
Kin	MC_KIN_IN_GENERAL_KIN (04_InterpolatedMotion\MPLCKinematics.lib)	FeedEnable	BOOLEAN	
		FALSE	Kinematic braking of the motion until standstill in the current coordinate system with the ramps of the current motion task	
		TRUE	Execute current motion task	
		RapidStop	BOOLEAN	
		TRUE	Kinematic braking of the motion until standstill in the current coordinate system with the configured rapid stop ramps	
		FALSE	Execute current motion task	
<b>Notice:</b> In a Cartesian operating mode, (such as KIN_TARGET_CART, KIN_LIN_3D), for example in a moved PCS coordinate system, the motion of the tools is decelerated when FeedEnable is reset or when it is set relative to the current, moved PCS. The axes are usually still moving. If the axes are to come to a standstill, then the axes (such as KIN_JOG/TARGET_AXIS) must be switched over to interpolation. See also note in chapter "Out.General (→ 106)", output variable "FBError".				
	OverridePercentage	UINT	1 .. configured maximum value in [%]	
Ignore	MC_KIN_IN_GENERAL_IGNORE (04_InterpolatedMotion\MPLCKinematics.lib)	Deactivation of monitoring		
		Inverter	ARRAY[0..8] OF BOOL	
		If TRUE, the existence as well as the status of the corresponding axes is ignored.  Configured axes that are unavailable or not ready for operation and ignored are still interpolated virtually, meaning they are included in the path calculations.		

Variable	Type	Meaning		
		Contrary to simulation of the axes, the actual status of the axes is displayed in the OUT range.		
		Ignored axes are switched to AM_DEFAULT via kinematic controller and Enable_Stop as well as Enable_RapidStop of these axes are set to FALSE.		
		Bits 1 to 8 correspond to axes 1 to 8. Bit 0 refers to all configured axes of this kinematic instance.		
		SWLS.Axis	BOOLEAN	If TRUE, then NO error is generated when you leave the configured software limit switch of at least one axis (A1 .. A8), at least one Cartesian coordinate (XY-ZABC) or at least one model-dependent kinematics limitation (such as the angle of a joint in a parallel kinematics to which no drive is assigned)
		SWLS.Cart	BOOLEAN	
		SWLS.Kin	BOOLEAN	
MotorSpeed	BOOLEAN	If TRUE, no error is generated when the configured, maximum permissible motor speed is exceeded		
KinHandicap	BOOLEAN	If TRUE, Cartesian degrees of freedom are ignored in the inverse kinematics calculation that does not support the kinematics model. For example, the orientation coordinates ABC are ignored in a model that cannot skew the flange FCS relative to the KCS reference coordinate system.		
Mode	MC_KIN_MODE	<p>Operating mode for all axes of the kinematics instance.</p> <p>KIN_AM_DEFAULT</p> <p>KIN_AM_HOMING,</p> <p>KIN_STOP</p> <p>KIN_JOG_AXIS, KIN_JOG_CART,</p> <p>KIN_TARGET_AXIS, KIN_TARGET_CART,</p> <p>KIN_LIN_XY, KIN_LIN_YZ, KIN_LIN_ZX, KIN_LIN_3D,</p> <p>KIN_CIRC_XY, KIN_CIRC_YZ, KIN_CIRC_ZX</p> <p><b>Note:</b> Alternatively, a comprehensive operating mode KIN_CONTINUOUS_PATH can be used instead of a specific continuous path operating mode (such as KIN_LIN_XY, KIN_CIRC_ZX, ...). In this case, the specific continuous path operating mode must be entered in the continuous path segment structure, see chapter "In.Cp.Segment (→ 99)". The advantage is that the CP segments of the data type MC_KIN_IN_CP_SEGMENT can be copied with the complete segment description including the specific CP mode in the application program, for example from an array of CP segments completely to the <i>In.Cp.Segment</i> input of the kinematic controller.</p>		

Variable	Type	Meaning
Reset	BOOLEAN	<p>An error is reset with a rising edge.</p> <p><b>Note:</b> Additionally, with a rising edge also the variables Axis.PastAbsoluteReachedMotorSpeed SWLS.Past_Axis_Cart_Kin SWLS.PastMotorSpeedWarning SWLS.PastMotorSpeed in the <i>Out.General</i> output variable are reset, see chapter "Out.General (→ 106)". These variables are also reset when changing to a kinematics mode (for example after an axis was not operationally ready) as well as in the KIN_AM_DEFAULT mode.</p>

## 11.2 In.Homing

Type: ST\_AxisGroupKin\_In\_Homing (03\_ApplicationModules\ MPLCAxisGroupControl\_Kinematics.lib)

Variable	Type	Meaning
Start	ARRAY[0..8] OF BOOL	<p>The configured referencing of the corresponding axis or axes is started with a rising edge. The signal must remain set during referencing until the corresponding bit in the Out.Homing.Done output variable is set to TRUE, see chapter "Out.Homing (→ 111)". In case of premature resetting of the input signal, the referencing of these axes is discontinued.</p> <p>Bits 1 to 8 correspond to axes 1 to 8. Bit 0 refers to all configured axes of this kinematic instance.</p>

## 11.3 In.Transform

Type: MC\_KIN\_IN\_TRANSFORM (04\_InterpolatedMotion\MPLCKinematics.lib)

The input variables in this structure allow for setting the transformations between the coordinate systems of the kinematic controller. If significant transformations have to be modified during motion of the robot (for example tracking of moved tool coordinate system PCS1), then the assignments must be made with constant signal characteristic and cyclically in the TaskPriority.

The array elements 1 to 3 correspond to the translational offsets along X,Y,Z of the reference coordinate system, which means WCS for transformations WCS\_KCS/PCS1/2 and FCS for tool transformation. The array elements 4 to 6 specify the rotation around the Z axis of the reference coordinate system, the subsequent rotation around the new (turned around Z) Y axis and the following rotation again around the new (turned around Z and Y) X axis.

For an overview of the coordinate systems as well as further information, refer to chapter "Changing the coordinate system (→ 124)".

Variable	Type	Meaning
WCS_KCS	ARRAY[1..6] OF LREAL	Transformation from World Coordinate System to Kinematics Coordinate System



Variable	Type	Meaning
WCS_PCS1	ARRAY[1..6] OF LREAL	Transformation from World Coordinate System to Piece Coordinate System 1
WCS_PCS2	ARRAY[1..6] OF LREAL	Transformation from World Coordinate System to Piece Coordinate System 2
Tool	ARRAY[1..6] OF LREAL	Transformation from Flange Coordinate System to Tool Coordinate System

The unit of components 1 to 3 is the configured unit [translational user unit].

The unit of components 4 to 6 is the configured unit [rotational user unit] = [rad] or [deg].



## NOTICE

Discontinuity of the signals in these variables have the following consequences:

- Risk of collision caused by unexpectedly large curves (for example when changing between coordinate systems or operating modes)
- Overspeed
- Buzzing drives
- Lag errors
- Before switching to a new coordinate system, the relevant transformations must always be applied already 5 cycles of TaskPriority.

## 11.4 In.MasterPosition

For target and continuous path interpolation, it is possible to switch off profile generation and to follow an externally supplied motion profile instead (see also chapters "Combination of cam disk and kinematics function (→ 132)" and "CP interpolation as slave of an external master profile (→ 133)"). The motion profiles are applied in the Task Priority at the inputs described here.

Type: MC\_KIN\_IN\_MASTERPOSITION (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
Target	ARRAY[1..8] OF LREAL	Optional master target position for TARGET modes Used if In.Target.UseMasterPosition.XYZABC_OR_A1ToA6 = TRUE, see chapter "In.Target (→ 95)" Position in [user unit], orientation in [user unit] = [rad] or [deg]

Variable	Type	Meaning
Cp	LREAL	<p>Optional master position for CP paths</p> <p>Used if <i>In.Cp.Settings.UseMasterPosition</i> = <i>TRUE</i> when changing (from an operating mode other than CP) to a CP operating mode, see chapter "In.Cp.Settings (→ 98)"</p> <p>Corresponds to the distance traveled along the CP path [user unit].</p> <p>Can begin with the desired value.</p> <p>Must be decelerated to rest until the path end is reached because the motion will otherwise be ended abruptly.</p> <p>Can easily be reset to the value 0.0, if it was &gt; 0 before.</p> <p>If the values are decremented, the value 0 must NOT be reached as this will be interpreted as resetting and therefore, in spite of applying the value 0, the value of the last cycle will be retained. This means that a reverse motion must be decelerated BEFORE reaching CP Master-Position 0. Otherwise, the allocation of a MasterPosition will drift away to a space position on multiple forward and backward movements.</p>

## INFORMATION



Discontinuity of the signals in these variables lead to motion control of:

- Buzzing drives
- Lag errors

## 11.5 In.Jog

Type: MC\_KIN\_IN\_JOG (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
Positive	ARRAY[1..8] OF BOOL	Positive jogging of up to 8 degrees of freedom (A1 .. A8 for KIN_JOG_AXIS, XYZABC + A7/8 for KIN_JOG_CART)
Negative	ARRAY[1..8] OF BOOL	Negative jogging of up to 8 degrees of freedom (A1 .. A8 for KIN_JOG_AXIS, XYZABC + A7/8 for KIN_JOG_CART)
CoordSys	MC_KIN_COORDSYS	<p>Coordinate system in which jogging takes place Cartesian for KIN_JOG_CART:</p> <p>KIN_KCS, KIN_TCS, KIN_WCS, KIN_PCS1, KIN_PCS2</p> <p>The setting KIN_ACS leads to jogging in KIN_KCS for KIN_JOG_CART, however, display of the current position in <i>Out.General.Position.ActCoordSys</i> in ACS.</p> <p>For KIN_JOG_AXIS, the axes are always jogged. The position in <i>Out.General.Position.ActCoordSys</i> is displayed in the set coordinate system.</p> <p>When set to KIN_TCS, the current position is displayed independent of the jog mode in KCS.</p>
VelocityPercentage	UINT	<p>Percentage of configured jog speeds</p> <p>0 .. configured maximum value in [%], default value 100%</p>

## 11.6 In.Target

Type: MC\_KIN\_IN\_TARGET (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning	
Position	ARRAY[1..8] OF LREAL;	Position:	[translational user unit],
		Orientation:	[rotational user unit] = [rad] or [deg]
CoordSys	MC_KIN_COORDSYS	KIN_KCS, KIN_ACS, KIN_TCS, KIN_WCS, KIN_PCS1, KIN_PCS2 For an overview of the coordinate systems, refer to chapter "Changing the coordinate system (→ 124).	
Constellation	UINT	Constellation in which a Cartesian target position should be moved to in KIN_TARGET_AXIS mode.	
		0	Current constellation
		1, 2	Explicitly defined constellation (for example for SCARA: elbow counterclockwise/clockwise)
		The current constellation is indicated in the <i>Out.General.Kin.Constellation</i> output variable, see chapter "Out.General (→ 106)". The constellation can only be changed in axis interpolation (KIN_JOG/TARGET_AXIS) or, for example, by manually shifting the axes when the machine is switched off.	
AxisPhase	ARRAY[1..8] OF DINT	Phases of the rotary axes in which a Cartesian target position should be moved to in KIN_TARGET_AXIS mode. In this case, a rotary axis is an axis that changes an angle of rotation between two bodies of the kinematic chain. If an integral multiple of 360° is added to this angle of rotation, then this leads to the same Cartesian position of the robot. In the "AxisPhase" variable, you can explicitly set in which integral multiple of 360° and in which axis phase the robot should move.	
		0	Automatic selection of the nearest phase with which the target position can be moved to without hitting the software limit switches of the axes.
		other values	Explicit selection, example: -5 means that the corresponding axis should move in the phase $-4 * 360^\circ$ , $-5 * 360^\circ$ .
		The array elements 1 to 8 correspond to the axes 1 to 8. The variable and array elements are NOT used: <ul style="list-style-type: none"> <li>For linear axes (which means axes that lead to a linear motion independent of the drive type)</li> <li>In operating modes other than KIN_TARGET_AXIS</li> <li>If <i>Config.General.AxisCartBijection</i> = TRUE (In this case, the phases of the axes are clearly identified automatically. You can set AxisCartBijection on the expert page of the configuration in the "Advanced" MotionStudio permission level.)</li> </ul>	

Variable	Type	Meaning		
UseMasterPosition	MC_KIN_IN_TARGET_USE-MASTERPOSITION	<p>If UseMasterPosition.XYZABC_OR_A1ToA6 = TRUE, no travel profile is generated for the main axes A1..A6. Instead, the target position is triggered at the In.MasterPosition.Target input in the TaskPriority cycle, see chapter "In.MasterPosition (→ 93)". The target position is expressed in Cartesian coordinates or in axis values depending on the set coordinate system "In.Target.CoordSys".</p> <p>This function allows, for example, traveling along curves that were calculated online, curve point tables, or user profiles with a kinematics. As soon as the UseMasterPosition variable is reset to FALSE, the kinematic controller resets in a jerk-free manner to the master motion supplied externally, and continues the motion by means of internally generated travel profiles. The externally supplied curves can be executed in any coordinate system, for example, even relative to a moved tool in a tracking application. Another interesting option is a special, applicative ABC orientation during ContinuousPath interpolation.</p> <p>See also chapter "Combination of cam disk and kinematics function (→ 132)".</p>		
Axis	MC_KIN_IN_TARGET_AXIS	Kinematic quantity for KIN_TARGET_AXIS mode		
		Velocity	ARRAY[1..8] OF LREAL	[User unit / sec], >= 0
		Acceleration	ARRAY[1..8] OF LREAL	[User unit / (sec*sec)], > 0
		Deceleration	ARRAY[1..8] OF LREAL	[User unit / (sec*sec)], >0, <= Config.Axis.RapidDeceleration[1..8]
		Jerk	ARRAY[1..8] OF LREAL	[ms], >= Config.Axis.Rapid-Jerk[1..8]
		<b>Note:</b> The Out.Diag.MaxAllowedAxisVelocity output variable specifies the maximum permissible axis velocity corresponding to the configured motor speed for all axes.		
Cart	MC_KIN_IN_TARGET_CART	Kinematic quantities for KIN_TARGET_CART mode		
		Velocity	ARRAY[1..6] OF LREAL	[User unit / sec], >= 0
		Acceleration	ARRAY[1..6] OF LREAL	[User unit / (sec*sec)], > 0
		Deceleration	ARRAY[1..6] OF LREAL	[User unit / (sec*sec)], >0, <= Config.Cart.RapidDeceleration[1..8]
		Jerk	ARRAY[1..6] OF LREAL	[ms], >= Config.Cart.Rapid-Jerk[1..8]

Variable	Type	Meaning	
Synchronization	ARRAY[0..8] OF BOOL	<p>The degrees of freedom for which the corresponding bits are set to TRUE, are synchronized.</p> <p>In KIN_TARGET_AXIS mode, the array elements 1 to 6 correspond to the axes A1 to A6.</p> <p>In KIN_TARGET_CART mode, the array elements 1 to 6 correspond to the Cartesian degrees of freedom XYZABC.</p> <p>Bit 0 = TRUE leads to synchronization of all degrees of freedom.</p> <p>The synchronization can also be started, canceled or changed during motion.</p> <p><b>Notes:</b></p> <ul style="list-style-type: none"><li>• In a pick and place motion implemented by means of KIN_TARGET_CART, it is usually meaningful to remove the vertical spatial coordinate, for example from the synchronization and to only synchronize, for example, X and Y.</li><li>• All the components must be set with the same jerk time (<i>In.Target.Cart.Jerk[i]</i>), so that a straight line will result for a synchronized rest-in-rest motion by means of KIN_TARGET_CART.</li></ul>	
ModuloMode	ARRAY[1..8] OF MC_KIN_MOD- ULO_MODE	In preparation	
RapidDynamic- sAtOvershooting	BOOLEAN	TRUE	For current or foreseeable overshoots exceeding a target coordinate, the configured rapid stop ramps are used for decelerating.
		FALSE	No action
ABCMapping	BOOLEAN	TRUE	In the change cycle of the KIN_TARGET_CART mode or the coordinate system change (KCS, WCS, PCS2, PCS2) within KIN_TARGET_CART mode, the CURRENT Cartesian orientation ABC (array elements 4 to 6) are mapped to the phase in which the orientation value is nearest to the orientation value in the variables "In.Target.Position[4..6]" in this cycle.
		FALSE	No action
		<p><b>Note:</b> Using this function, you can set an absolute Cartesian orientation that you expect in your application in the corresponding situation and position of the robot. Based on this absolute Cartesian orientation, you can move to clearly defined, absolute Cartesian orientation values in your sequential program. Chapter "Dealing with ambiguities of the ABC orientation values (→ 135)" explains the background of this function in detail.</p>	

## 11.7 In.Cp

Type: MC\_KIN\_IN\_CP (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
Settings	MC_KIN_IN_CP _SETTINGS	See chapter "In.Cp.Settings (→ 98)"

Variable	Type	Meaning
BackToPath	MC_KIN_IN_CP_BACKTOPATH	See chapter "In.Cp.BackToPath (→ 99)"
Segment	MC_KIN_IN_CP_SEGMENT	See chapter "In.Cp.Segment (→ 99)"
Path	MC_KIN_IN_CP_PATH	See chapter "In.Cp.Path (→ 103)"

### 11.7.1 In.Cp.Settings

Type: MC\_KIN\_IN\_CP\_SETTINGS (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
PlusTargetCartABC	BOOLEAN	If TRUE in the change cycle (of an operating mode other than ContinuousPath) in ContinuousPath, the ABC orientation coordinates are not interpolated by means of ContinuousPath, but by means of KIN_TARGET_CART via the <i>In.Target</i> variable, see chapter "In.Target (→ 95)". This function is also described in chapter "Combining continuous path modes and TARGET modes (→ 79)".
IncludeABC	BOOLEAN	If TRUE in the change cycle (from an operating mode other than Continuous Path) in ContinuousPath, the ABC orientation coordinates are interpolated at the same time using ContinuousPath. <i>PlusTargetCartABC</i> and <i>IncludeABC</i> cannot be activated both at the same time.
IncludeAxis7	BOOLEAN	If TRUE in the change cycle (from an operating mode other than Continuous Path) in ContinuousPath, Axis 7 is moved in a way that is synchronized with ContinuousPath interpolation.
IncludeAxis8	BOOLEAN	If TRUE in the change cycle (from an operating mode other than Continuous Path) in ContinuousPath, Axis 8 is moved in a way that is synchronized with ContinuousPath interpolation.
UseMasterPosition	BOOLEAN	If TRUE in the change cycle (from an operating mode other than Continuous Path) in ContinuousPath, no CP traversing profile is generated by the kinematic controller. Instead, the position profile for the path progression along the entered CP path must be entered applicatively with constant signal characteristic in the TaskPriority in the variables <i>In.Master.Position.Cp</i> , see chapter "In.MasterPosition (→ 93)".  <b>Note:</b> The input variables "In.General.Kin.FeedEnable/RapidStop" are NOT used in this case. Leaving the work envelope as well as over-speed or other errors do NOT lead to decelerating the motion.

## 11.7.2 In.Cp.BackToPath

Type: MC\_KIN\_IN\_CP\_BACKTOPATH (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
Enable	BOOLEAN	<p>Activation of repositioning to the CP path, for example after inverter fault.</p> <p>For status information about repositioning, refer to the Out.Cp.BackToPath variable, see chapter "Out.Cp (→ 111)".</p> <p><b>Note:</b> "In.Target.Axis" must be used with the desired kinematic quantities because they are used for repositioning. Repositioning is only possible when no operating mode other than ContinuousPath was applied since the occurrence of the error that led to exiting the CP path.</p>
FeedEnable	BOOLEAN	Feed enable for repositioning on the path. This variable can be set and reset during repositioning.
MaxInitialDistance	LREAL	<p>Maximum distance (translational distance in [user unit]) from the path.</p> <p>If this value is exceeded at the beginning of repositioning, an error is generated. The error message indicates the actual distance. In that case, you can set an adequately large value and try again to start repositioning.</p>

## 11.7.3 In.Cp.Segment

Type: MC\_KIN\_IN\_CP\_SEGMENT (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
ID	INT	<p>ID for unique identification of the ContinuousPath segment and as a trigger for accepting a new CP segment in the kinematic controller.</p> <p>In.Cp.Segment.ID &lt;&gt; Out.Cp.Queue.LastMappedSegID signals to the kinematic control that a new segment has been inserted that should be accepted by the controller.</p> <p>When traveling on the ContinuousPath, the ID of the respective current segment is shown in the output variables Out.Cp.Queue.ActSegID.</p>
UserID	DWORD	<p>User ID of the CP segment</p> <p>WITHOUT function for kinematic control</p> <p>When traveling on the ContinuousPath, the ID of the respective current segment is indicated in the output variables Out.Cp.Queue.ActUserID.</p>
Mode	MC_KIN_MODE	<p>Usually, the operating mode is set in the <i>In.General.Mode</i> variable. Alternatively, a comprehensive KIN_CONTINUOUS_PATH mode can be used instead of a specific ContinuousPath mode (such as KIN_LIN_XY, KIN_CIRC_ZX, etc.). In that case, the specific ContinuousPath mode must be entered here in the <i>In.Cp.Segment.Mode</i> variable.</p> <p>The advantage is that the CP segments of the data type MC_KIN_IN_CP_SEGMENT can be copied with the complete segment description including the specific CP mode in the application program, for example from an array of CP segments completely to the <i>In.Cp.Segment</i> input of the kinematic controller.</p>
Position	ARRAY[1..8] OF LREAL	Target position of the CP segment in the current coordinate system (AxisGroupKin.Out.General.Kin.CoordSys)



Variable	Type	Meaning
Circ	MC_KIN_IN_CP _SEG- MENT_CIRC	Additional circle parameters Mode: MC_KIN_CIRC_MODE
		KIN_CENTER_ANGLE <ul style="list-style-type: none"> <li>Input of the center of the circle in <i>AuxPos</i></li> <li>and the angle to be covered in <i>Angle</i>.</li> </ul> <b>Note:</b> <i>Radius</i> and <i>Segment.Position</i> are <b>NOT</b> used.
		KIN_CENTER_ENDPOS <ul style="list-style-type: none"> <li>Input of the circle end point in <i>Segment.Position</i></li> <li>and of the center of the circle in <i>AuxPos</i>.</li> </ul> <b>Notes:</b> <ul style="list-style-type: none"> <li><i>Radius</i> and <i>Angle</i> are <b>NOT</b> used</li> <li>The circle center is corrected automatically.</li> <li>The maximum permissible correction is configured: Config.Cp.CircCenterCorrectionThreshold, default value 100000</li> </ul>
		KIN_INTERPOS_ENDPOS <ul style="list-style-type: none"> <li>Input of the circle end point in <i>Segment.Position</i></li> <li>and an additional point on the circular arc in <i>AuxPos</i>.</li> </ul> <b>Note:</b> <ul style="list-style-type: none"> <li><i>Radius</i> and <i>Angle</i> are <b>NOT</b> used.</li> </ul>
		KIN_RADIUS_ANGLE <ul style="list-style-type: none"> <li>Input of the circle radius in <i>Radius</i></li> <li>and the angle to be covered in <i>Angle</i>.</li> </ul> <b>Notes:</b> <ul style="list-style-type: none"> <li><i>AuxPos</i> and <i>Segment.Position</i> are <b>NOT</b> used.</li> <li>The circle segment is attached tangentially.</li> </ul> <b>Prerequisite:</b> The <i>Out.Cp.Path.EndTangentValid</i> variable must be set.
		KIN_RADIUS_ENDPOS <ul style="list-style-type: none"> <li>Input of the circle radius in <i>Radius</i></li> <li>and the circle end point in <i>Segment.Position</i>.</li> </ul> <b>Notes:</b> <ul style="list-style-type: none"> <li><i>AuxPos</i> and <i>Angle</i> are <b>NOT</b> used.</li> <li>When a negative radius is entered, the longer of the two possible arcs to the circle end point is selected.</li> <li>When a positive radius is entered, the shorter arc is selected.</li> </ul>
		Direction MC_KIN_DIRECTION In top view at the XY, YZ and ZX plane <ul style="list-style-type: none"> <li>KIN_CW: clockwise</li> <li>KIN_CCW: counterclockwise</li> </ul>



Variable	Type	Meaning		
			<b>Notes:</b> <ul style="list-style-type: none"> <li>Direction is <b>NOT</b> used when mode = KIN_INTER-POS_ENDPOS.</li> </ul>	
		AuxPos	ARRAY[1..6] OF LREAL	for example circle center, elements 4 to 6 are reserved
		Radius	LREAL	[User unit]
		Angle	LREAL	Angle to be blended by the circular arc, configured [rotational user unit] = [Degr] or [Rad], > 0, theoretically any number of full circles can be entered with an angle > 360° (limited only by the total path length that may not exceed the limit of the precisely mapped maximum value represented by a 64-bit floating point number (> 1000.000.000.000.000.000 ... [user unit])); if EllipticDistortionFactor <> 1.0, the value of angle is automatically rounded to an integer multiple of pi (or 180 degrees).
		EllipticDistortionFactor	LREAL	Elliptic distortion of the circular arc of a circle > 0, corresponds to the expansion or reduction of the circular axis parallel to the start tangent. The path length and velocity are determined for the undistorted circle (EllipticDistortionFactor = 1.0), that is the variable Out.Cp.Path.Distance shows the remaining path length for EllipticDistortionFactor = 1.0, although <> 1 is set. → Distortion values <> 1.0 lead to distortions of the geometry as well as of the velocity profile
Blending	MC_KIN_IN_CP_SEGMENT_BLENDING	Parameter for blending TO this segment. That means, it is NOT used when CP.Queue is empty (Out.Cp.Queue.Size = 0).		
		Distance	LREAL	Max blending distance, >= 0.0

Variable	Type	Meaning		
		LimitationPer-centage	UINT	[%], used for limiting the blending distance in THIS CP segment to 1 to 99% of the length of THIS segment. This means that the Blending.Distance is automatically limited to the LimitationPercentage of the segment that directly follows on blending curve.
		Velocity.Profile	MC_KIN_CP_BLENDED_VELOCITY_PROFILE <ul style="list-style-type: none"> <li>• KIN_HANDLING: For blending with handling paths</li> <li>• KIN_LOWER: Smaller of the subsequent path velocities</li> <li>• KIN_UPPER: Larger of the subsequent path velocities</li> <li>• KIN_PRECEDENT: Precedent path velocity</li> <li>• KIN_SUBSEQUENT: Path velocity in the current segment</li> <li>• KIN_AVERAGE: Average of the precedent/ current path velocity</li> <li>• KIN_CENTRIFUGAL: Centrifugal force limit</li> </ul>	
		Velocity.Percentage	UINT	1 .. configured maximum value in [%], default value 100% percentage scaling of the Velocity.Profile
		<ul style="list-style-type: none"> <li>• <b>Note:</b> For setting <i>Velocity.Profile</i> := <i>KIN_CENTRIFUGAL</i>, the minimum of Translation.Deceleration of the previous CP segment and Translation.Acceleration of the currently entered CP segment is used for limiting the centrifugal acceleration in the blending curve. If you want quicker movement in the blending curve, then you can set a Velocity.Percentage &gt; 100%. For this purpose, set <i>Config.Cp.MaxBlendingVelocityPercentage</i> in the MotionStudio "Advanced" permission level on the expert page of the configuration to a value &gt; 100. The path velocity in the blending curve is limited in addition to the average value of Translation.Velocity of the preceding and the currently entered segment.</li> <li>• <b>Note:</b> Blending is specified with the CpSegment TO which the blending curve leads. The reason for this implementation is that the next target point in dynamic processes is often only obtained during motion. Seen from a temporal perspective, the type and kinematic quantities of the interpolation to the target point as well as the type and kinematic quantities of the blending curve TO the segment to the target point can be entered only with the new target point. This is the reason why blending cannot be parameterized with the last segment.</li> </ul>		
Translation	MC_KIN_IN_CP_SEGMENT_TRANSLATION	In this structure, the kinematic quantities for translation of the TCP along the ContinuousPath path are specified. For example, <i>Translation.Velocity</i> corresponds to the maximum path velocity in this segment.		
		Velocity	LREAL	[User unit / sec], > 0

Variable	Type	Meaning		
		Acceleration	LREAL	[User unit / (sec * sec)], > 0
		Deceleration	LREAL	User unit / (sec * sec)], > 0, <= Config.Cp.RapidTransDeceleration
		Jerk	LREAL	[ms], >= Config.Cp.RapidTransJerk
Cart	MC_KIN_IN_CP_SEG-MENT_CART	In the array components 4 to 6, you specify the motion parameters (vel, acc, dec, jerk) for the synchronized rotation (ABC) of the tool. Is only used when Out.Cp.Settings.IncludingABC = TRUE. Array components 1 to 3 are reserved.		
Axis	MC_KIN_IN_CP_SEG-MENT_AXIS	In the array components 7 to 8, you specify the motion parameters (vel, acc, dec, jerk) for the synchronized movement of the auxiliary axis (A7, A8). Is only used when Out.Cp.Settings.IncludingA7 or ..A8 = TRUE. Array components 1 to 6 are reserved.		

#### 11.7.4 In.Cp.Path

Type: MC\_KIN\_IN\_CP\_PATH (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
VelocityPercentage	UINT	0 .. configured maximum value in [%], default value 100% Percentage scaling of the programmed velocity profile can be changed during path motion.
Reverse	BOOLEAN	TRUE: After the queue was emptied and it was not yet overwritten by new elements in the ring buffer principle due to continuous filling of the queue, the path is traveled backwards, up to the point where it reaches the first queue entry.
FidelityPercentage	USINT	90% to 100% 100%: Exact path control <100%: Smoothing of curve discontinuities <b>Note:</b> Is accepted during a change (from an operating mode other than ContinuousPath) to ContinuousPath as well as during "Standstill in the current coordinate system". For small override values, an increased path accuracy can be obtained due to resource limiting as compared to the setting of override = 100%.

#### 11.8 In.Simu3D

Type: ST\_AxisGroupKin\_In\_Simu3D (03\_ApplicationModules\ MPLCAxisGroupControl\_Kinematics.lib)

Variable	Type	Meaning
Enable	BOOLEAN	Connection to the 3D simulation PC

Variable	Type	Meaning		
Ctrl	MC_KIN_SIMU3D_CONTROL_Agk (04_InterpolatedMotion\ MPLCSimu3D.lib)	Pen_TCP_KCS	BOOLEAN	Pen on TCP, lines relative to KCS
		Pen_TCP_WCS	BOOLEAN	Pen on TCP, lines relative to WCS
		Pen_TCP_PCS 1	BOOLEAN	Pen on TCP, lines relative to PCS1
		Pen_TCP_PCS 2	BOOLEAN	Pen on TCP, lines relative to PCS2
		DeleteAllPen- Lines	BOOLEAN	Delete all lines

## 11.9 In.Diag

Type: ST\_AxisGroupKin\_In\_Diag (03\_ApplicationModules\ MPLCAxisGroupControl\_Kinematics.lib)

Here you have the option to activate the output of useful messages for different functions.

Variable	Type	Meaning
Kin.MessageAtAutoSlowdown	<ul style="list-style-type: none"> <li>MotorSpeed: BOOLEAN</li> <li>Centrifugal: BOOLEAN</li> <li>CartRotation: BOOLEAN</li> <li>AuxAxis: BOOLEAN</li> </ul>	Message during automatic reduction of the path velocity
Config.MessageAtReadingXML	BOOLEAN	Message when reading XML files
Simu3D.MessageAtCommunicationError	BOOLEAN	Message when communication problems occur for 3D simulation

## 11.10 Out.General

Type: ST\_AxisGroupKin\_Out\_General (03\_ApplicationModules\ MPLCAxisGroupControl\_Kinematics.lib)

Variable	Type	Meaning	
Inverter	ST_AxisGroupKin_Out_General_Inverter (03_ApplicationModules\ MPLCAxisGroupControl_Kinematics.lib)	Connected	ARRAY[0..8] OF BOOL
		Referenced	ARRAY[0..8] OF BOOL
		Powered	ARRAY[0..8] OF BOOL
		Ready	ARRAY[0..8] OF BOOL
		Error	ARRAY[0..8] OF BOOL
		InGear	ARRAY[0..8] OF BOOL
		InKinMode	ARRAY[0..8] OF BOOL
		StandStill	ARRAY[0..8] OF BOOL
		SafeTorqueOff	ARRAY[0..8] OF BOOL
		HWLS	ARRAY[0..8] OF BOOL
		Simulation	ARRAY[0..8] OF BOOL
		State_Error	ARRAY[1..8] OF BYTE
		AxisControlError	ARRAY[0..8] OF BOOL
		The bits 1..8 correspond to the axes 1..8. Bit 0 refers to all configured axes of this kinematic instance.	
Axis	MC_KIN_OUT_GENERAL_AXIS (04_InterpolatedMotion\MPLCKinematics.lib)	Motor speed monitoring	
		MotorSpeed	ARRAY[1..8] OF LREAL
		PastAbsoluteReachedMotorSpeed	ARRAY[1..8] OF LREAL
		<b>Note:</b> PastAbsoluteReachedMotorSpeed is reset with a rising edge at the input In.General.Reset as well as when changing to a kinematic mode (for example after an axis was not operationally ready), and in KIN_AM_DEFAULT mode.	

Variable	Type	Meaning		
Kin	ST_AxisGroup-Kin_Out_General_Kin (03_ApplicationModules\MPLCAxisGroupControl_Kinematics.lib)	CoordSys	MC_KIN_COORDSYS	
		Constellation	UINT;	Kinematic constellation, for example for SCARA elbow counter-clockwise/clockwise
		LimitValue	ARRAY[1..12] OF LREAL	Values of the kinematic limitations (such as angle)
		ChainItem	ARRAY[1..24] OF LREAL	Detailed data of the kinematic chain, such as elbow position with reference to BCS, depending on the configured kinematics model
		TranslationVelocity	LREAL;	Translational velocity of the TCP in KCS when CoordSys = KCS, ACS, TCS, else in the current "CoordSys"
		OverridePercentage	UINT;	
		Ready	BOOL;	TRUE: Ready for kinematic modes
		Done	BOOL;	TRUE: The current kinematics motion task is executed completely.

Variable	Type	Meaning		
Position	MC_KIN_OUT_GENERAL_POSITION (04_InterpolatedMotion\MPLCKinematics.lib)	ActCoordSys	ARRAY[1..8] OF LREAL	Position in the current coordinate system  in KCS when CoordSys = TCS, else in the current "CoordSys"
		ACS	ARRAY[1..8] OF LREAL	Position in ACS
		ACS_Incr	:ARRAY[1..8] OF DINT	Position in ACS increments
		KCS	ARRAY[1..8] OF LREAL	Position in KCS
		WCS	ARRAY[1..8] OF LREAL	Position in WCS
		PCS1	ARRAY[1..8] OF LREAL	Position in PCS1
		PCS2	ARRAY[1..8] OF LREAL	Position in PCS2
		KCS_WCS_PCS_Valid	BOOLEAN	FALSE: Cartesian position and orientation not applicable, for example because it is not feasible or because at least one of the used axes 1 to 6 is not referenced. (Axes 1 to <i>Config.General.NumberOfKinAxes</i> for which <i>In.General.Ignore.Inverter[n]</i> = FALSE, are "used".  For example <i>NumberOfKinAxes</i> = 4 for KIN_SCARA_RRRL_XYZA_M10)
SWLS	MC_KIN_OUT_GENERAL_SWLS (04_InterpolatedMotion\MPLCKinematics.lib)	Work envelope monitoring: Axes / Cartesian / kinematic		
		Axis	AtLeastOn	BOOLEAN
			Positive	ARRAY[1..8] OF BOOL;
			Negative	ARRAY[1..8] OF BOOL;
		Cart	AtLeastOne	BOOLEAN
			Positive	ARRAY[1..6] OF BOOL;
			Negative	ARRAY[1..6] OF BOOL;



Variable	Type	Meaning		
		Kin	AtLeastOne	BOOLEAN
			Positive	ARRAY[1..12] OF BOOL;
			Negative	ARRAY[1..12] OF BOOL;
		Past_Axis_Cart_Kin		BOOLEAN
		MotorSpeedWarning		ARRAY[0..8] OF BOOL;
		PastMotorSpeedWarning		ARRAY[0..8] OF BOOL
		MotorSpeed		ARRAY[0..8] OF BOOL
		PastMotorSpeed		ARRAY[0..8] OF BOOL If bit 0 is set, a corresponding overrun is present for at least one axis.
		<b>Note:</b> Past_Axis_Cart_Kin, PastMotorSpeedWarning and PastMotorSpeed are reset with a rising edge at the <i>In.General.Reset</i> input and when changing to a kinematic mode (for example after an axis was not operationally ready), as well as in KIN_AM_DEFAULT mode.		

Variable	Type	Meaning		
Standstill	MC_KIN_OUT_GENERAL_STANDSTILL (04_InterpolatedMotion\MPLCKinematics.lib)	ActCoordSys	BOOLEAN	Standstill in the current coordinate system  in KCS when CoordSys = TCS, else in the current "CoordSys"  Includes standstill of auxiliary axes A7/8
		ACS	ARRAY[0..8] OF BOOL	Standstill of the axes  Bits 1 to 8 correspond to axes 1 to 8.  Bit 0 refers to all configured axes of this kinematic instance.
		KCS_TCP	BOOLEAN	Standstill of the TCP in KCS
		WCS_TCP	BOOLEAN	Standstill of the TCP in WCS
		PCS1_TCP	BOOLEAN	Standstill of the TCP in PCS1
		PCS2_TCP	BOOLEAN	Standstill of the TCP in PCS2
Mode	MC_KIN_MODE	Current operating mode of the kinematics instance: KIN_AM_DEFAULT, KIN_AM_HOMING, KIN_STOP, KIN_JOG_AXIS, KIN_JOG_CART, KIN_TARGET_AXIS, KIN_TARGET_CART, KIN_LIN_XY, KIN_LIN_YZ, KIN_LIN_ZX, KIN_LIN_3D, KIN_CIRC_XY, KIN_CIRC_YZ, KIN_CIRC_ZX, KIN_CP_BLENDED		
ModeType	MC_KIN_MODE	Type of current operating mode: KIN_AM_DEFAULT, KIN_AM_HOMING, KIN_STOP, KIN_JOG_AXIS, KIN_JOG_CART, KIN_TARGET_AXIS, KIN_TARGET_CART, KIN_CONTINUOUS_PATH		
FBError	BOOLEAN	<p>TRUE: Fault in the kinematics instance.</p> <ul style="list-style-type: none"> <li>TRUE and AxisGroupKin.Out.General.Ready = TRUE The motion is decelerated in the coordinate system.</li> </ul> <p><b>NOTICE:</b> In a Cartesian operating mode (such as KIN_TARGET_CART, KIN_LIN_3D) and, for example, in a moved coordinate system, the motion of the tool is decelerated relative to the current, moved PCS. The axes are usually still moving. If the axes are to come to a standstill, then the axes (such as KIN_JOG/TARGET_AXIS) must be switched over to interpolation. See also the note in chapter "In.General (→ 90)", Kin.FeedEnable/Rapid-Stop input variable.</p> <ul style="list-style-type: none"> <li>TRUE and AxisGroupKin.Out.General.Ready = FALSE The motion is NOT kinematically decelerated along the axes by the MultiMotion profile generator.</li> </ul> <p>FALSE: There is no error.</p>		

Variable	Type	Meaning
FBErorID	DWORD	See chapter "Error codes (→ 144)"

### 11.11 Out.Homing

Type: ST\_AxisGroupKin\_Out\_Homing (03\_ApplicationModules\MPLCAxisGroupControl\_Kinematics.lib)

Variable	Type	Meaning
Done	ARRAY[0..8] OF BOOL	Referencing performed successfully. Bits 1 to 8 correspond to axes 1 to 8. Bit 0 refers to all configured axes of this kinematic instance.

### 11.12 Out.Cp

Type: MC\_KIN\_OUT\_CP (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning		
Settings	MC_KIN_OUT_CP_SETTINGS (04_InterpolatedMotion \MPLCKinematics.lib)	For ContinuousPath settings, see also chapter "In.Cp.Settings (→ 98)".		
		PlusTargetCartABC	BOOLEAN	
		IncludingABC	BOOLEAN	
		IncludingAxis7	BOOLEAN	
		IncludingAxis8	BOOLEAN	
		MasterPosUse	BOOLEAN	
BackToPath	MC_KIN_OUT_CP_BACKTO-PATH (04_InterpolatedMotion \MPLCKinematics.lib)	BackToPath status, see also chapter "In.Cp.BackToPath (→ 99)".		
		InitialDistance	LREAL	Translational distance in [user unit]) from the path to the start of repositioning.
		Active	BOOLEAN	TRUE: Repositioning to the path is in progress. (Also TRUE if the axes do not move due to <i>In.Cp.BackToPath.FeedEnable</i> = FALSE or <i>In.Target.Axis.Velocity/Acceleration[i]</i> = 0)
		Done	BOOLEAN	TRUE: Repositioning to the path is finished, for example after an inverter fault. "Done" is reset with <i>In.Cp.BackToPath.Enable</i> .

Variable	Type	Meaning		
Queue	MC_KIN_OUT_CP_QUEUE	LastMapped-SegID	INT	ID of the last CP segment that was accepted successfully in the queue, independent of whether this segment is still in the queue or has already been completely traveled.
		<b>Note:</b> In a sequential control for filling the CP queue, the <i>Out.Cp.Queue.LastMappedSegID</i> output variable, for example, can simply be incremented in order to obtain an appropriate ID ( <i>In.Cp.Segment.ID</i> input variable) for the next CP segment to be filled.		
		ActSegID	INT	Shows the Segment.ID of the segment that is just traveled through.
		ActUserID	DWORD	Shows the Segment.UserID of the segment that is just traveled through.
		Size	UINT	Number of CP segments (WITHOUT the blending segments added by the kinematic controller) that are in the queue and have not yet been completely traveled through.
		SizeWithBlending	UINT	Number of CP segments that are currently in the queue and have not yet been completely traveled through.
		<b>Note:</b> A blending segment automatically added by the kinematic controller has the same Segment.ID/UserID as the CP segment TO which it blends. The <i>Out.Cp.Queue.SizeWithBlending</i> variable also includes the automatically added blending segments.		
Path	MC_KIN_OUT_CP_PATH	Distance	LREAL	Remaining path length until reaching the target point of the last accepted CP segment.
		EndReached	BOOLEAN	TRUE: CP path end reached. Can be different from ( <i>Cp.Queue.SizeWithBlending</i> = 0) or ( <i>Cp.Path.Distance</i> = 0.0), when for example <i>In.Cp.Path.FidelityPercentage</i> < 100%
		EndTangentValid	BOOLEAN	TRUE: The tangent of the last entered CP segment is valid. This means that relative input of CP segments is possible. (such as KIN_CIRC_xx by means of KIN_RADIUS_ANGLE)  FALSE: Relative input (such as KIN_CIRC_xx by means of KIN_RADIUS_ANGLE) is not possible.

### 11.13 Out.Simu3D

Type: ST\_AxisGroupKin\_Out\_Simu3D (03\_ApplicationModules\ MPLCAxisGroupControl\_Kinematics.lib)

Variable	Type	Meaning
Dummy	BOOLEAN	Reserved

### 11.14 Out.Diag

Type: MC\_KIN\_OUT\_DIAG (04\_InterpolatedMotion\MPLCKinematics.lib)

Variable	Type	Meaning
NoAttemptTo-Configure	BOOLEAN	TRUE, as long as no attempt has been made to configure the kinematic controller since the start of MOVI-PLC®, for example because the required axes are not connected.
NewConfig-TakenOver	BOOLEAN	Reserved
NumberOf-Successful-Configurations	UINT	Is incremented once when changing (from a NON kinematic operation mode, such as KIN_AM_DEFAULT) to a kinematic mode (such as KIN_JOG_AXIS) as well as cyclically to the KIN_AM_DEFAULT mode by using the axis increments read from the inverters.
KinModel	MC_KIN_MODEL (04_InterpolatedMotion\MPLCKinematics.lib)	Kinematic models (→ 57)
UserKinNum	INT	Number of the user kinematics when KinModel = KIN_USER_KINEMATICS
MaxAllowedAxisVelocity	ARRAY[1..8] OF LREAL	Shows the maximum permissible axes velocities in [user unit / sec], derived from the maximum motor speed and WarningPercentage. These values are the maximum possible velocities (in.Target.Axis.Velocity) for KIN_TARGET_AXIS mode.
LastComputationTime	DWORD	Computation time in [usec] of the kinematics path calculation in the last cycle of TaskPriority.
MaxComputationTime	DWORD	Maximum computation time in [usec] of the kinematics path calculation since the last configuration of the kinematic controller (changing to a kinematic mode, for example after an axis was not ready for operation, as well as cyclic in the KIN_AM_DEFAULT mode)
License	MC_KIN_OUT_DIAG_LICENSE(04_InterpolatedMotion\MPLCKinematics.lib)	Activated license as well as the technology points consumed for this kinematics instance (see Required technology level (→ 13))
		ConsumedTechnologyCredits
		Mode_Jog_Target
		Mode_Lin2D_Circ2D
		Mode_Lin3D_Circ3D_AxisCP
		Model_No_CartesianGantry
		World_Piece_CoordinateSystem

## 12 Functions and function blocks of the MPLCKinematics library

The library `04_InterpolatedMotion/MPLCKinematics.lib` includes useful function blocks for creating the user program.

- MC\_KinInProximity
- MC\_KinMonitorCpQueueSegment
- MC\_KinCoordSysMeasurement
- MC\_KinEncoderDataProcessing
- MC\_KinWcsPcs1Assignment
- MC\_KinWcsPcs2Assignment
- MC\_KinProfGen

### 12.1 MC\_KinInProximity function

The MC\_KinInProximity function can be used in the motion sequence program for checking whether the distance to the target position or to the target orientation has at least one parameterizable limit value.

For TARGET interpolation, you can use this function within a sequence step in the transitions for switching to the next step and consequently to the new specified target.

Accordingly, you can create a sequential control for ContinuousPath interpolation when the next ContinuousPath segment is only to be entered in the queue when, for example, the distance to the target orientation falls below a certain value. In this way you can implement the coordination of ContinuousPath segments and the associated orientation values. For example, in a pick and place motion, the segment for lowering the gripper is supplied to the kinematic controller only when the rotation of the gripper, controlled by means of TARGET interpolation, has reached the required angle in space (see chapter "Combining continuous path modes and TARGET modes (→ 79)").

Input signal	Type	Meaning
<i>Threshold</i>	LREAL	Limit value for checking the distance (angle or translatory distance depending on the settings in <i>CoordSys</i> and <i>Dimension</i> ).
<i>Position</i>	ARRAY[1..8] OF LREAL	Axis values or position to which the distance is checked.
<i>CoordSys</i>	MC_KIN_COORDSYS	Coordinate system where the distance check is to be carried out. <b>Note:</b> KCS will be checked automatically with the setting KIN_TCS.

Input signal	Type	Meaning
<i>Dimension</i>	BYTE	<p>The <i>Dimension</i> byte is used for the bit-wise setting of the dimensions to be included for determining the distance, such as:</p> <p>2#0000_0011: Dimensions 1, 2</p> <p>2#0011_1000: Dimensions 4, 5, 6</p> <p><b>Note:</b> The distance check in a Cartesian coordinate system (input signal <i>CoordSys</i>; KIN_KCS/WCS/PCS1/2) uses the ABC orientation values in dimensions 4/5/6. However, these are ambiguous. When using KIN_TARGET_AXIS and specifying the target in Cartesian coordinates, the kinematics will approach the target using the next axis tuple. The ABC values are continuously updated if possible. It might happen that the target axis values have been reached but the ABC representation differs from the one created in the <i>Position</i> input signal. In this case, the function will not set the return value (with small <i>threshold</i>) to TRUE.</p>
Return value	Type	Meaning
<i>MC_KinInProximity</i>	BOOLEAN	TRUE, if the distance to the target ( <i>Position</i> input signal) that is interpreted in the specified coordinate system ( <i>CoordSys</i> input signal) does not exceed the maximum threshold value.
Input/output signal type	Type	Meaning
<i>ActPos</i>	MC_KIN_OUT_GENERAL_POSITION	Current position, used for determining the distance to the <i>position</i> input signal.

## 12.2 MC\_KinCoordSysMeasurement function block

You can use the MC\_KinCoordSysMeasurement function block to measure a coordinate system, for example the position and orientation of a conveyor belt in the WCS coordinate system.

To do so, 3 points of the coordinate system to be measured from perspective of the reference coordinate system (such as WCS) have to be applied to the *Position* input signal one after the other. These points include the origin, a point on the positive x axis, and a point in the xy plane (positive y coordinate) of the coordinate system to be measured.

When the point to be entered is present at the *Position* input, you have to generate a rising edge for each point at the respective *SetPosition...* input to have the function block accept the points. A rising edge at the *ComputeTransform* input results in the calculation of the transformation that is output in the *TransformVector* / *TransformFrame* output signals as soon as the *Done* output signal changes to TRUE.

You can determine the 3 points to be entered, for example, by moving the TCP (for example of a sensor tip) in jog mode to the 3 points. The *AxisGroupKin.Inst[.].Out.General.Position.xCS* variable, for example, should be created at the *Position* input for performing measurement in relation to the xCS coordinate system. For more information, see chapter "Calibrating a coordinate system (→ 127)".

Input signal	Type	Meaning
<i>EN</i>	BOOLEAN	TRUE: The function block is executed FALSE: The function block is not executed
<i>Start</i>	BOOLEAN	TRUE: Measurement of the coordinate system is active FALSE: Reset of coordinate system measurement and all output variables.
<i>ResetError</i>	BOOLEAN	An error is reset with a rising edge <b>Note:</b> The measurement made prior to an error reset will not be discarded.
<i>SetPosition_CoordSysOrigin</i>	BOOLEAN	With the rising edge of this input signal, the point present in the <i>Position</i> signal is adopted as origin of the coordinate system to be measured.
<i>SetPosition_AlongX_Positive</i>	BOOLEAN	The point present in the <i>Position</i> input signal is accepted with the rising edge of this input signal. It is interpreted as a point on the positive x axis of the coordinate system to be measured.
<i>SetPosition_In-XYplane_Positive</i>	BOOLEAN	The point present in the <i>Position</i> input signal is accepted with the rising edge of this input signal. It is interpreted as a point in the XY plane of the coordinate system to be measured with positive Y coordinate.
<i>ComputeTransform</i>	BOOLEAN	The transformation to the coordinate system to be measured is calculated with the rising edge of this input signal.
<i>Position</i>	ARRAY[1..8] OF LREAL	The first 3 elements of this input signal are used as point (X, Y, Z coordinates) in the coordinate system to which the transformation to the coordinate system to be measured relates. <b>Example:</b> If the <i>AxisGroupKin.Inst[.].Out.General.Position.WCS</i> variable is applied to this input, the coordinate system will be measured in relation to the World Coordinate System (WCS). This means the transformation of WCS to the coordinate system to be measured is output in the <i>CoordSysVector</i> / <i>CoordSysFrame</i> output signals.
<i>TransformOffset</i>	ARRAY[1..6] OF LREAL	A transformation can be specified in this input signal. This transformation can be appended in addition to the coordinate system to be measured. This means it is included in the <i>TransformVector</i> / <i>TransformFrame</i> output signals. <b>Note:</b> Enter the tuple (0,0,0,0,0,0) in this input signal to avoid that an additional offset is created.

Output signal	Type	Meaning
<i>TransformVector</i>	ARRAY[1..6] OF LREAL	Transformation to the coordinate system to be measured in XYZABC representation.
<i>TransformFrame</i>	ARRAY[1..4, 1..4] OF LREAL	Transformation to the coordinate system to be measured in 4x4 matrix representation.
<i>Position_CoordSysOrigin</i>	ARRAY[1..3] OF LREAL	Display of the origin taken over from the coordinate system to be measured.



Output signal	Type	Meaning
<i>Position_AlongX_Xpositive</i>	ARRAY[1..3] OF LREAL	Display of the point taken over from the positive X axis of the coordinate system to be measured.
<i>Position_InXY-plane_Ypositive</i>	BOOLEAN	Display of the point taken over from the XY plane of the coordinate system to be measured with positive Y coordinate.
<i>PositionIs-Set_CoordSys-Origin</i>	BOOLEAN	Displays whether the point of origin of the coordinate system to be measured was taken over.
<i>PositionIs-Set_AlongX_Xpositive</i>	BOOLEAN	Displays whether the point on the positive X axis of the coordinate system to be measured was taken over.
<i>PositionIs-Set_InXY-plane_Ypositive</i>	BOOLEAN	Displays whether the point in the XY plane of the coordinate system to be measured with positive Y coordinate was taken over.
<i>Error</i>	BOOLEAN	TRUE: An error has occurred while measuring the coordinate system (for example the point taken over for origin is identical with the point on the positive X axis; several <i>SetPosition_...</i> input signals are set to TRUE simultaneously; not all required points were entered).
<i>Done</i>	BOOLEAN	Measurement has been finished and the transformation is available in the <i>TransformVector</i> / <i>TransformFrame</i> output signals.

Input signal/ output signal	Type	Meaning
<i>Config</i>	ST_AxisGroup-Kin_Config	Kinematic configuration of the instance for the control of which the <i>CoordSysVector</i> output signal is determined.

### 12.3 MC\_KinEncoderDataProcessing function block

The MC\_KinEncoderDataProcessing function block converts an encoder signal into user position values, for example for tracking an object on a conveyor belt.

For example, you can apply the signal of a light barrier to the *Trigger* input signal. When a trigger is detected, the *Position* output signal is set to the value of the *Trigger-PositionOffset* input signal and is then updated using the encoder values.

You can use the *Position* output signal, for example, for updating a component cyclically (such as X coordinate) of the WCS\_PCSx transformation. The kinematic controller must not be active in the PCSx coordinate system at the time when the trigger is detected because the Position output signal usually causes a jump. The MC\_KinEncoderDataProcessing function block therefore also indicates whether the kinematic controller had sufficient time to observe the current movement in the new PCSx coordinate system. Do not change to the new PCSx coordinate system until the *ObservationTimeOver* output changes to TRUE. This method is necessary to ensure smooth switchover.

## INFORMATION



The MC\_KinEncoderDataProcessing function block must be called cyclically in the TaskPriority. Input and output variables of the respective instance of MC\_KinEncoderDataProcessing must only be written or read in the sequence of motion program that usually runs in another task.

If the encoder values are read by an inverter, for example by means of SCOM objects, then the SCOM objects have to be synchronized using the TaskPriority. In the TaskPriority, they have to be transferred to the MC\_KinEncoderDataProcessing function block.

Input signal	Type	Meaning
<i>EN</i>	BOOLEAN	TRUE: The function block is executed. FALSE: The function block is not executed.
<i>Standby</i>	BOOLEAN	A trigger can only be registered when the input signal <i>Standby</i> = TRUE. Even when the input signal <i>Standby</i> = FALSE, the output signals <i>Position</i> and <i>Encoder_Incr</i> are continued to be updated.
<i>Trigger</i>	BOOLEAN	The <i>Position</i> output signal is set to the value of the <i>TriggerPositionOffset</i> input signal with a rising edge on the <i>Trigger</i> input. Output signals <i>TriggerOccurred</i> are set to TRUE and <i>ObservationTimeOver</i> to FALSE.  <b>Note:</b> As long as <i>Standby</i> = TRUE, only one rising edge on the <i>Trigger</i> input is detected. Other rising edges are ignored to avoid unexpected jumps in the <i>Position</i> output signal that is used, for example, as the <i>AxisGroupKin.Inst[.].In.Transform.WCS_PCS1</i> input signal. To ensure that a trigger is detected, you must set <i>Standby</i> to FALSE in advance. Only after the output <i>InStandby</i> has been reset to FALSE, can you set <i>Standby</i> to TRUE again. The TRUE level at the <i>Trigger</i> input is already detected as trigger signal in the cycle in which <i>Standby</i> is set to TRUE, independent of the level in the previous cycle (even if <i>Trigger</i> was set to TRUE). The intention of this behavior is to ensure that a <i>trigger</i> is created. For example, you can connect a program variable by means of an OR operation parallel to a light barrier signal at the <i>Trigger</i> input. Independent of the previous characteristic of the sensor signal, the value TRUE in the program variable results in triggering the <i>Trigger</i> signal. It is possible that a sensor damped by an object leads to triggering of the <i>Trigger</i> in the cycle when <i>Standby</i> is set to TRUE. This would be the case if the object was already partially moved past the sensor at that time. If you want to safely detect the moment when the object begins to dampen the sensor, you should use an R_TRIG function block. In this case apply output <i>fb_R_TRIG.Q</i> to the <i>Trigger</i> input.
<i>TriggerPositionOffset</i>	LREAL	Position to which the <i>Position</i> output is set when the trigger was detected.
<i>EncoderWordLow</i>	WORD	Incremental encoder low word, for example external encoder signal that is read-in cyclically in the TaskPriority by the inverter using SCOM receive.
<i>EncoderWordHigh</i>	WORD	Encoder high word
<i>EncoderDoubleWord</i>	DWORD	Alternative representation of incremental encoder low/high word.

Input signal	Type	Meaning
<i>UseDWord_InsteadLowHigh</i>	BOOLEAN	TRUE: EncoderDoubleWord is read FALSE: EncoderWordLow/High are read.
<i>Numerator</i>	LREAL	Increment delta for converting the encoder increments into the position value.
<i>Denominator</i>	LREAL	Position delta that corresponds to the increment delta.
<i>Filter</i>	INT	In preparation

Output signal	Type	Meaning
<i>Position</i>	LREAL	Position in user units set using the <i>Numerator</i> and <i>Denominator</i> input signals.  When a trigger is detected, the position is reset to the value of the <i>TriggerPositionOffset</i> input signal.
<i>Encoder_Incr</i>	DWORD	Identical to the <i>EncoderDoubleWord</i> input signal, if <i>UseDWord_InsteadLowHigh</i> = TRUE. Else, combined value of <i>EncoderWordLow</i> and <i>EncoderWordHigh</i> .
<i>InStandby</i>	BOOLEAN	Feedback that the input signal <i>Standby</i> is set.  <b>Note:</b> The output signal is necessary for programming the motion sequence in another task than the TaskPriority to ensure that the <i>Standby</i> in the TaskPriority was actually reset to FALSE. This is the only way to reliably detect a trigger after setting <i>Standby</i> .
<i>TriggerOccurred</i>	BOOLEAN	Feedback that a trigger was detected.
<i>ObservationTimeOver</i>	BOOLEAN	The ObservationTimeOver output signal is reset to FALSE in the cycle in which a trigger is detected.  <b>Notes:</b> <ul style="list-style-type: none"> <li>In that cycle, even the <i>position</i> output is reset so that the signal characteristic of <i>position</i> usually has a step change.</li> <li>If the Position signal is used as the input signal <i>AxisGroupKin.Inst[.].In.Transform.WCS_PCS1 [1]</i>, for example, then changeover to the PCS1 coordinate system will occur several TaskPriority cycles later to give the kinematic controller sufficient time to observe the current movement in the new PCS1 coordinate system.</li> <li>The <i>ObservationTimeOver</i> output signal is set to TRUE several cycles after the trigger was detected.</li> </ul> <p>Therefore, changeover to the new coordinate system (such as PCS1) is permitted after <i>Standby</i> was set to TRUE if <i>TriggerOccurred</i> = TRUE and <i>ObservationTimeOver</i> = TRUE.</p>

## 12.4 MC\_KinWcsPcs1Assignment function block

The MC\_KinWcsPcs1Assignment function block assigns the input variables *AxisGroupKin.Inst[.].In.Transform* of the AxisGroupControl Kinematics program module to the transformation from WCS to the coordinate system that is specified in the *TransformIndex* input signal.

The function block can be used in applications where the motion sequences are programmed in more than 2 piece coordinate systems PCS (max. KIN\_TRANSFORM\_ARRAY\_SIZE = 10, can be connected with another value using a constant with the same name).

The inconsistent change of the course of *AxisGroupKin.Inst[.].In.Transform.WCS\_PCS1/2* may only take place when the kinematics is controlled in another coordinate system. Theoretically, you can change between any number of different coordinate systems. However, doing so requires continuously variable assignment to PCS1/2. To eliminate this variability, you can keep the transformations to the numerous coordinate systems of your application in an array or update them cyclically. Each transformation must have its defined position within the array. During the process, you can then use the MC\_KinWcsPcs1Assignment or MC\_KinWcsPcs2Assignment function block to assign the transformation of the required coordinate system to the PCS1 or PCS2 coordinate system that is not used at the moment.

## INFORMATION



The MC\_KinWcsPcs1Assignment function block must be executed cyclically in the TaskPriority.

The *TransformIndex* input signal, however, is assigned in the task where the motion sequence is programmed. To do so, either leave the *TransformIndex* input empty (delete the question mark after having inserted the function block) and directly access the *TransformIndex* input from the other task (such as fb\_WcsPcs1Assignment.CoordSys = COORDSYS\_CONVEYORBELT), or use a variable for the transfer.

The current transformations from WCS to the coordinate systems are assigned to an array that is generated at the input/output signal *TransformArray*. In the case of continuously variable transformations (such as using the *Position* output of the MC\_KinEncoderDataProcessing function block), this assignment must be made cyclically in the TaskPriority.

To simplify handling of the program, you can define an enumeration for the indices of the array to access the transformations used in your application by means of easily comprehensible designations (such as COORDSYS\_CONVEYORBELT, COORDSYS\_MOVINGOBJECT, COORDSYS\_RACK, etc.).

As soon as a new coordinate system is selected at the *TransformIndex* input, the *ObservationTimeOver* output signal is reset to FALSE. Once the kinematic controller had sufficient time to observe the current movement in the new coordinate system (several cycles of the TaskPriority), the *ObservationTimeOver* output is set to TRUE again.

Once you have assigned the required coordinate system (fb\_WcsPcs1Assignment.CoordSys = 'desired coord.sys') and the observation time has elapsed (fb\_WcsPcs1Assignment.ObservationTimeOver = TRUE), you can change the kinematic controller to PCS1 (*AxisGroupKin.Inst[.].In.Target.CoordSys* = KIN\_PCS1).

Input signal	Type	Meaning
<i>EN</i>	BOOLEAN	TRUE: The function block is executed. FALSE: The function block is not executed.
<i>TransformIndex</i>	INT	Index of the coordinate system to which PCS1 is to be assigned. The index refers to the array that is transferred in the <i>TransformArray</i> input/output signal.

Output signal	Type	Meaning
<i>ObservationTimeOver</i>	BOOLEAN	TRUE: Switchover of the coordinate system in PCS1 is permitted if the <i>AssignedTransform</i> output corresponds to the required <i>TransformIndex</i> . FALSE: Switchover to PCS1 not permitted.
<i>AssignedTransform</i>	INT	Index of the coordinate system that is currently assigned to the <i>AxisGroupKin.Inst[.].In.Transform.WCS_PCS1</i> variable.
<i>Error</i>	BOOLEAN	TRUE: TransformIndex has a value of < 1 or > KIN_TRANSFORM_ARRAY_SIZE. In this case, the function block does not assign any values to the <i>AxisGroupKin.Inst[.].In.Transform.WCS_PCS1</i> variable. <b>Note:</b> This error disappears automatically as soon as a valid <i>TransformIndex</i> is transferred.

Input/output signal	Type	Meaning
<i>Transform</i>	MC_KIN_IN_TRANSFORM	Variable <i>AxisGroupKin.Inst[.].In.Transform</i>
<i>TransformArray</i>	ARRAY[1..KIN_TRANSFORM_ARRAY_SIZE] OF ARRAY[1..6] OF LREAL	Array that contains the transformations of WCS to the application-specific coordinate systems.

## 12.5 MC\_KinWcsPcs2Assignment function block

The MC\_KinWcsPcs2Assignment function block behaves exactly like the MC\_KinWcsPcs1Assignment function block. The only difference is the assignment to WCS\_PCS2 instead of to WCS\_PCS1.

## 12.6 MC\_KinProfGen function block

The MC\_KinProfGen function block creates a jerk-limited position profile for a dimension. The profile generator uses kinematic position and motion values in LREAL accuracy. It can be initialized with any start velocity and start acceleration.

Input signal	Type	Meaning
<i>EN</i>	BOOLEAN	TRUE: The function block is executed. FALSE: The function block is not executed.
<i>FeedEnable</i>	BOOLEAN	TRUE: <i>ActPos</i> is moved to the <i>TargetPos</i> FALSE: The movement is decelerated using the ramps <i>Dmax</i> and <i>JerkTime</i> .
<i>RapidStop</i>	BOOLEAN	TRUE: The movement is decelerated using the ramps <i>D_Rapid</i> , <i>JerkTime_Rapid</i> . <i>FeedEnable</i> is internally set to FALSE. FALSE: No effect, that is <i>FeedEnable</i> applies.

Input signal	Type	Meaning
<i>TargPos</i>	LREAL	Target position in freely selectable user unit. The user unit is not explicitly selected as "mm", "m", or similar. Instead, the same unit is used throughout for all signals of the function block.
<i>Vmax</i>	LREAL	Maximum velocity for movement to the target position. Setting > 0 (user unit/user-defined time base) When the movement is initialized with a velocity > <i>Vmax</i> , the profile generator decelerates to <i>Vmax</i> .
<i>Amax</i>	LREAL	Maximum acceleration (increase in kinetic energy). Setting > 0 (user unit/user-defined time base <sup>2</sup> )
<i>Dmax</i>	LREAL	Maximum deceleration (decrease in kinetic energy). Setting > 0 (user unit/user-defined time base <sup>2</sup> )
<i>JerkTime</i>	LREAL	Jerk time (time for change of acceleration). Setting ≥ 0 (user-defined time base)
<i>D_Rapid</i>	LREAL	≥ <i>Dmax</i> is used instead of <i>Dmax</i> to avoid overshoot, or when <i>RapidStop</i> = TRUE.
<i>JerkTime_Rapid</i>	LREAL	≤ <i>JerkTime</i> is used instead of <i>JerkTime</i> to avoid overshoot, or when <i>RapidStop</i> = TRUE.
<i>RapidDynamic-sAtOvershooting</i>	BOOLEAN	FALSE: <i>Dmax</i> / <i>JerkTime</i> is used even when overshoot is detected. TRUE: <i>D_Rapid</i> / <i>JerkTime_Rapid</i> is used even when overshoot is detected.
<i>ModuloMode</i>	MC_KIN_DIRECTION	KIN_SHORT, KIN_CW, KIN_CCW

Output signal	Type	Meaning
<i>ActPos</i>	LREAL	Current position value of the profile generator
<i>ActVel</i>	LREAL	Current velocity value of the profile generator (user unit/user-defined time base)
<i>ActAcc</i>	LREAL	Current acceleration value of the profile generator (user unit/user-defined time base <sup>2</sup> )
<i>TargPosReached</i>	BOOLEAN	TRUE: Target position reached with <i>ActPos</i> = <i>TargetPos</i> and <i>ActVel</i> = 0 and <i>ActAcc</i> = 0. FALSE: Target position not reached.
<i>Overshooting</i>	BOOLEAN	TRUE: The course of the profile <i>ActPos</i> currently moves away from the target position <i>ActPos</i> . FALSE: No overshoot.
<i>ModuloPhase-Change</i>	SINT	0: No phase change -1: negative phase change 1: positive phase change
<i>Error</i>	BOOLEAN	TRUE: The motion is decelerated using <i>D_Rapid</i> / <i>JerkTime_Rapid</i> . <b>Note:</b> Error reset only possible by re-initialization.
<i>ErrorID</i>	DWORD	Error ID



Output signal	Type	Meaning
<i>Init</i>	MC_KIN_PROF_GEN_INIT	<p>The profile generator is initialized by assigning the following structure components:</p> <ul style="list-style-type: none"> <li>• <i>New_Init</i>: BOOLEAN Level-controlled initialization of the profile generator TRUE: Re-initialization FALSE: No initialization <b>Note:</b> <i>New_Init</i> must be explicitly reset to FALSE, else initialization takes place in every cycle. The first profile values are already calculated in the cycle in which initialization takes place (<i>New_Init</i> = TRUE). These values are output in <i>ActPos</i>.</li> <li>• <i>DeltaT</i>: LREAL Virtual progress of time per execution of a profile generator Setting &gt; 0 (user-defined time base) Examples: Interpolation time: 5 ms (user-defined time base: s) <i>DeltaT</i> = 0.005 Interpolation time: 10 ms (user-defined time base: ms) <i>DeltaT</i> = 10</li> <li>• <i>InitPos</i>: LREAL Initial position (user unit)</li> <li>• <i>InitVel</i>: LREAL Initial velocity (user unit/user-defined time base)</li> <li>• <i>InitAcc</i>: LREAL Initial acceleration (user unit/user-defined time base<sup>2</sup>)</li> <li>• <i>Modulo</i>: BOOLEAN TRUE: Absolute modulo positioning Note: (Modulo = TRUE) and (New_Init = TRUE) requires: ModuloUnderflow ≤ InitPos ≤ ModuloOverflow False: Absolute positioning</li> <li>• <i>ModuloUnderflow</i>: LREAL Modulo limit value for negative phase change</li> <li>• <i>ModuloOverflow</i>: LREAL Modulo limit value for positive phase change</li> </ul>

## 13 Applications

### 13.1 Referencing a roller gantry

Proper interpolating Cartesian travel is only possible for SCARA or articulated-arm kinematics, for example, when all single axes are correctly referenced and configured. Only then the trigonometric calculations of the kinematic transformations will lead to correct results, so that for example a linear interpolation really results in travel along a line in 3D space.

The roller gantry kinematics is special because interpolating Cartesian travel along the KCS-X, KCS-Y, KCS-Z axes is possible even if the axes have not been referenced correctly. For Cartesian travel, the axes must be referenced. This can be done, for example, by referencing via type 5/8 at the point where the axes are actually located. In the Cartesian system, the TCP is in the zero position if the axis increments show the value 0 (if *AxisGroupKin.Inst[.].Config.Axis.KinModelOffset[i]*, *Cart.Offset\_KCS[i]* is 0).

Instead of referencing the single axes, you can move the TCP of the roller gantry in the Cartesian system to reference cams. By referencing again at the point where the reference cams switch, you move the Cartesian zero position of the TCP to that point.

Alternatively, if you know the Cartesian positions reached with the switching of the reference cams, for example, you may travel to the point where you want the zero position to be, and reference again at that point. If your machine is mechanically unable to travel to the required zero position, you can define a suitable transformation from WCS to KCS, and execute your program in the WCS coordinate system.

### 13.2 Open parallel kinematics

When the ends of the partial chains of parallel kinematics, for example the connections of both arms of a DELTA kinematic model (see DELTA (→ 64)) cannot come into contact because of the arm lengths, distances or axis values, then it is not possible to determine a Cartesian position of the TCP with the correct configuration. Correspondingly, no Cartesian interpolation is possible in this situation (such as KIN\_JOG/TARGET\_CART, KIN\_LIN/CIRC).

If you temporarily ignore the drawback in this situation by setting the input variable *AxisGroupKin.Inst[.].In.General.Ignore.KinHandicap* to TRUE, you can still move the axes in axis interpolation (KIN\_JOG/TARGET\_AXIS). As soon as the kinematic chain of the parallel kinematics can be closed, you must reset the variable *AxisGroupKin.Inst[.].In.General.Ignore.KinHandicap* to FALSE.

When a Cartesian position can be calculated, the output variable *AxisGroupKin.Inst[.].Out.General.Position.KCS\_WCS\_PCS\_Valid* = TRUE.

### 13.3 Changing the coordinate system

Motion sequences are programmed and executed with reference to a coordinate system. For example, the default coordinate system is KIN\_KCS, that is usually located in the kinematics base. Regarding the control technology, it makes no difference whether the selected coordinate system is at standstill or whether it moves in relation to the WCS, as long as the used kinematics can execute the programmed motions.

There are 6 different coordinate systems available for programming (see the following figure). A distinction is made between kinematics-related coordinate systems and facilities-related coordinate systems.

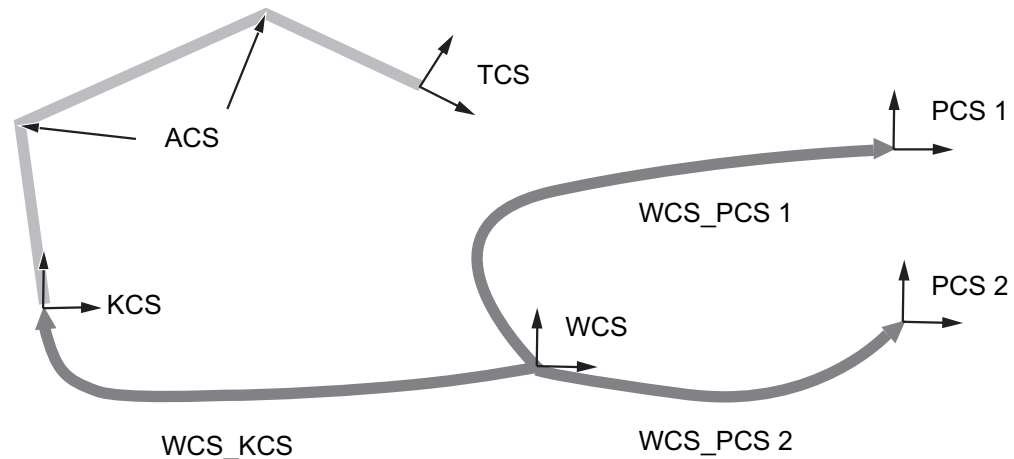


**Kinematics-related coordinate systems:**

- KIN\_KCS: Kinematics Coordinate System (often in the base of the kinematics)
- KIN\_ACS: Axis Coordinate System (axis values)
- KIN\_TCS: Tool Coordinate System (in the tool TCP)

**Facilities-related coordinate systems:**

- KIN\_WCS: World Coordinate System (shared for the system)
- KIN\_PCS1: Piece Coordinate System 1 (workpiece 1)
- KIN\_PCS2: Piece Coordinate System 2 (workpiece 2)



12549710731

**Notes:**

- Programming via KIN\_ACS means that target positions or a circle center must be entered as axis values. The specification of the interpolation for reaching the target points is carried out via the operating mode irrespective of whether you enter axis values or Cartesian coordinates.

This results in the following combination of input coordinates and type of interpolation.

Type of interpolation	Coordinate input	
	CoordSys = KIN_ACS	Cartesian coordinate system
<b>KIN_TARGET_AXIS</b> This interpolation results in a change of the KinConstellation if the axis values of the start and the end of the interpolation belong to different constellations.	Axis-by-axis travel to the target axis values	Axis-by-axis travel to the target axis values that correspond to the input Cartesian position. The conversion of the Cartesian position to the axis values is based on the <i>KinConstellation</i> and <i>AxisPhase</i> entered by the user. The axis interpolation is carried out on the shortest way to the determined axis values.

Type of interpolation	Coordinate input	
	CoordSys = KIN_ACS	Cartesian coordinate system
<b>Cartesian interpolation</b> The original KinConstellation is maintained for this interpolation.	Cartesian travel (such as linear) to the Cartesian position that corresponds to the specified axis values, or conversion of the entered axis values to a circle center.	Cartesian travel to the Cartesian target position. Cartesian interpolation is carried out for each component (X, Y, Z, A, B, C).

Selecting the coordinate system KIN\_TCS is useful for the Cartesian jog mode KIN\_JOG\_CART, for example, to relatively move the TCP.

The transformations between WCS and KCS/PCS1/PCS2 are supplied to the *AxisGroupKin.Inst[.].In.Transform.WCS\_KCS*, *WCS\_PCS1*, *WCS\_PCS2* variable as 6-dimensional vectors. The vectors express the position of KCS/PCS1/PCS2 in the WCS coordinate system. The first 3 components present the lag of the origin from WCS to the X, Y, Z direction. Component 4 expresses the rotation of WCS around the WCS Z axis. The turned coordinate system is then rotated around the new Y axis by the value in component 5. Then the system is rotated around the new X axis by the value in component 6.

The transformations *WCS\_KCS*, *WCS\_PCS1* and *WCS\_PCS2* can be static or can be changed during runtime to adapt the workpiece coordinate system PCS1 to a workpiece based on sensor information. In this case the change of the transformation currently used must be carried out continuously. Once control is carried out in WCS, PCS1 or PCS2 (*AxisGroupKin.Inst[n].Out.General.Kin.CoordSys* output), the kinematics controller tries to compensate for the change in the relevant transformations by moving the involved axes accordingly. Hence, sudden changes of the relevant transformations cause sudden axis movements. With control in WCS, the relevant transformation is *WCS\_KCS*. With control in PCS1, the relevant information is *WCS\_KCS* and *WCS\_PCS1*, and with control in PCS2 it is *WCS\_KCS* and *WCS\_PCS2*. The coordinate systems currently not used (for example *WCS\_PCS2* with control in PCS1) may be subject to inconsistent changes to switch to another workpiece, for example.

## INFORMATION



Before you switch the control to this coordinate system, you have to ensure a continuous progression of the relevant transformations over several cycles for the control in a certain coordinate system. This time is required for monitoring the current motion in the new coordinate system to provide for a jerk-free transition. Depending on the program structure, 3 to 5 TaskPriority cycles are required.

To assist you in ensuring the required monitoring time, the *MC\_KinEncoderDataProcessing*, *MC\_KinWcsPcs1Assignment*, *MC\_KinWcsPcs2Assignment* function blocks provide the *ObservationTimeOver* output signal.

You can change the coordinate system by simply specifying the appropriate coordinate system at the input *AxisGroupKin.Inst[.].In.Target.CoordSys* or *AxisGroupKin.Inst[.].In.Jog.CoordSys* depending on the selected operating mode. Beginning from this cycle, the programmed motions refer to the new coordinate system.

The two workpiece coordinate systems KIN\_PCS1/2 allow for changing between the coordinate systems of any number of workpieces. This is achieved by switching the coordinate system that is not used to another workpiece.

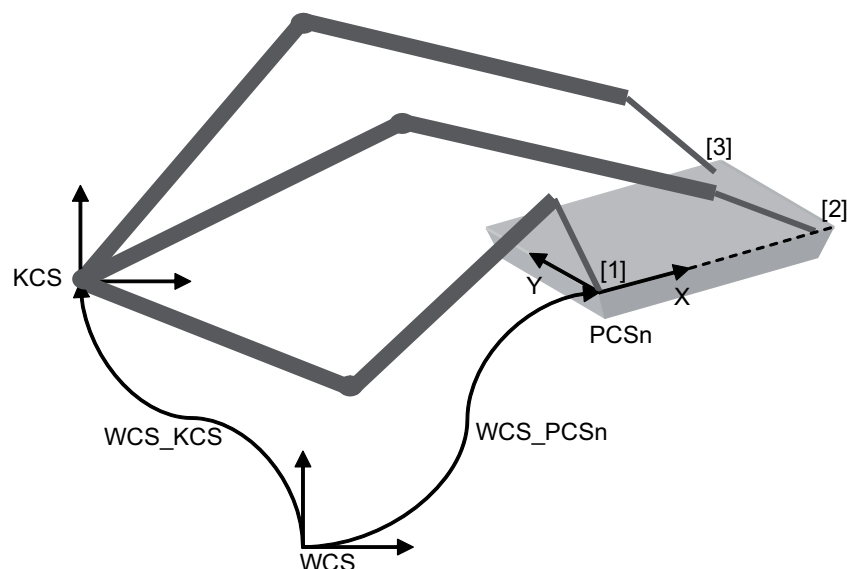
In addition to the WCS\_KCS/PCS1/PCS2 transformations, the tool transformation *AxisGroupKin.Inst[.].In.Transform.Tool* is part of the kinematic chain. It represents the transformation between the flange coordinate system FCS and the tool coordinate system TCS. For Cartesian interpolation (KIN\_JOG/TARGET\_CART, KIN\_LIN\_xx, KIN\_CIRC\_xx), the tool transformation must be changed only constantly as otherwise, inconsistent compensating motions of the axes occur. For axis interpolation (KIN\_JOG/TARGET\_AXIS, KIN\_AXIS\_CP) or in KIN\_AM\_DEFAULT, the tool transformation must be changed rapidly. In that case, the Cartesian position of the robot also shifts (*AxisGroupKin.Inst[.].Out.General.Position* → all except ACS/ACS\_Incr).

## 13.4 Calibrating a coordinate system

A measurement can be used, for example, to have to program travel paths in this coordinate system only once. The coordinate system can be located at the corner of a pallet, for example. If the pallet is moved, its position merely has to be determined in relation to WCS. The programmed palletizing pattern, however, can still be used.

The MC\_KinCoordSysMeasurement function block is available for measuring the position of a coordinate system in relation to another coordinate system (such as PCS1 in WCS, that is for determining the WCS\_PCS1 transformation). The function block requires 3 positions (as viewed from the coordinate system), to which the transformation to the coordination system measured refers to. These 3 positions include the origin, a point on the positive X axis, and a point in the XY plane with a positive Y coordinate.

To determine those positions, you can move the TCP of the kinematics to the positions in jog mode and read the respective actual positions from the coordinate system to which you want the transformation to refer.



12549714827


- [1] Origin of the PCS1/2 coordinate system
- [2] Point on the positive X axis of the PCS1/2 coordinate system
- [3] Point in the XY plane of the PCS1/2 coordinate system with positive Y coordinate

### Notes:

If, for example, PCS1/2 is to be measured with reference to WCS with the TCP of the kinematic model, the WCS\_KCS transformation must already be determined correctly and supplied to the *AxisGroupKin.Inst[.].In.Transform.WCS\_KCS* variable.

The highest accuracy can be reached if the point on the positive X axis is as far away from the origin as possible, and the point in the XY plane is as far away from the X axis as possible. If you use the TCP of the kinematics for the measurement, the measuring points should not be significantly outside the work envelope where the handling or processing in the coordinate system is carried out. The reason is to reduce inaccuracies when modeling the kinematics (such as arm lengths, deflections).

The orientation that the TCP of the kinematics uses to travel to the measuring points is theoretically irrelevant because only the X, Y, Z coordinates are used to calculate the transformation. However, the orientation should not differ significantly from the range of orientations where the handling or processing is carried out to reduce inaccuracies in modeling the kinematics and the tool.

For more information, refer to chapter "MC\_KinCoordSysMeasurement function block (→  115)".

### 13.5 Program structure for tracking applications with PCS1/2

Tracking refers to the execution of handling or processing tasks with a synchronization to moved workpieces.

The synchronization is carried out via control in the PCS1/2 workpiece coordinate system that is moved with the workpiece. For this purpose, you must supply the `WCS_PCS1/2` transformation cyclically to the variable `AxisGroupKin.Inst[...].In.Transform.WCS_PCS1/2`. If the motion of the workpiece is known (for example constant velocity along the WCS X direction), you can cyclically increment the corresponding components of the `WCS_PCS1/2` vector in the TaskPriority, for example. If the motion of the workpiece varies, the motion might have to be detected cyclically via sensors. You have to smoothen noisy sensor signals appropriately before they are written into the `WCS_PCS1/2` vector.

The `MC_KinEncoderDataProcessing` function block is available for editing encoder signals. The function block must be executed cyclically in the TaskPriority. The function block converts the encoder values to the covered distance in the user units you have specified.

The following figure shows the separation of the program in the cyclical assignment of the transformations in the interpolation task and the switching of the respectively required coordinate system in the TaskMain.

TaskPriority	TaskMain
...	...
(* Execution of FB MC_KinEncoderDataProcessing, therefore e.g. cyclic reading of SCOM *)	CASE (MotionSequenceState) OF...
fbMC_Receive_CAN(...);	10: ... (* Motion commands *) ...
fbMC_KinEncoderDataProcessing(...);	20: (* Switching to PCS1 in a TARGET mode, IF OBSERVATION TIME OVER *)
(* Assignment WCS_PCS1 *)	AxisGroupKin.Inst[...].In.Target.CoordSys := KIN_PCS1;
AxisGroupKin.Inst[...].In.Transform.WCS_PCS1[1] := fbKinEncoderDataProcessing.Position;	30: ... (* Motion commands *) ...
...	40: (* Switching to PCS2 in a TARGET mode, IF OBSERVATION TIME OVER *)
(* Assignment WCS_PCS2 *)	AxisGroupKin.Inst[...].In.Target.CoordSys := PCS2;
AxisGroupKin.Inst[...].In.Transform.WCS_PCS2[1] := 500.0;	50: ... (* Motion commands *) ...
...	END_CASE

12549766155

In this example, the X component of the `WCS_PCS1` transformation is updated cyclically based on encoder values. The X component of the `WCS_PCS2` transformation, however, has a fixed value. All the other components of the transformation must be assigned accordingly. During the motion sequence in PCS2, you can switch the `WCS_PCS1` transformation to another moved workpiece, for example. In order to switch between a random number of moved workpieces (for example picking bakery

products from a conveyor belt), you may also carry along the PCS2 coordinate system with the workpiece. The sudden assignment of a new transformation to WCS\_PCS1/2 (for another workpiece) is permitted while the control is carried out in a different coordinate system.

#### Notes:

For implementation, also observe the monitoring times for jerk-free switching explained in chapter "Changing the coordinate system".

It is not possible to change the coordinate system in a ContinuousPath mode. The switchover is carried out via KIN\_TARGET\_CART or KIN\_TARGET\_AXIS in automatic mode. You can switch to ContinuousPath once the motion in the new coordinate system has stopped for a moment (*AxisGroupKin.Inst[.].Out.General.Standstill.ActCoordSys = TRUE*). This does not mean that the axes or the TCP have to stand still relatively to KCS. Merely the motion in a moved coordinate system must stand still for a moment.

For further information, refer to chapter "MC\_KinEncoderDataProcessing function block".

## 13.6 Program structure for tracking applications with more than 2 PCS

The MC\_KinWcsPcs1/2Assignment function blocks allow for conveniently managing more than 2 workpiece coordinate systems.

For example, if your application requires workpiece coordinate systems for

- Placing on belt 1
- Placing on belt 2
- Picking from belt 1
- Picking from belt 2
- Pallet place 1
- Pallet place 2
- etc.,

then you can alternately assign the transformation to the WCS\_PCS1 and WCS\_PCS2 inputs. In order to facilitate this management, you can enter all the required transformations in a central array, or update them cyclically if there are moved objects:

```
TransformArray: ARRAY [1..KIN_TRANSFORM_ARRAY_SIZE] OF ARRAY
[1..6] OF LREAL;

TransformArray[LAYDOWN_CONVEYOR_1] := ...;
TransformArray[LAYDOWN_CONVEYOR_2] := ...;
TransformArray[PICKING_CONVEYOR_1] := ...;
TransformArray[PICKING_CONVEYOR_2] := ...;
TransformArray[PALLET_1] := ...;
TransformArray[PALLET_2] := ...;
etc.
```

In order to use meaningful indexes, you should create an enumeration of your coordinate systems.

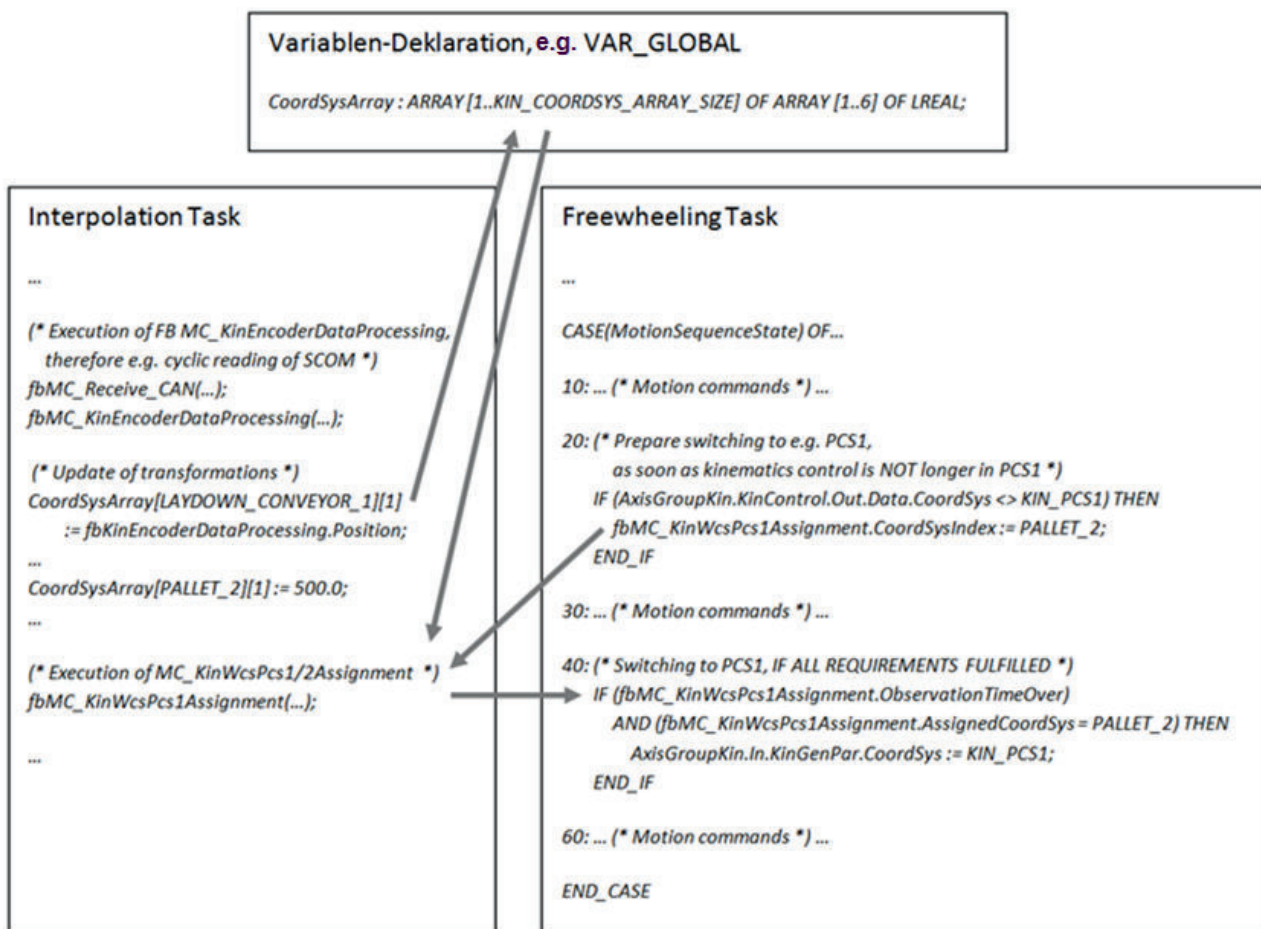
```
TYPE MY_COORDSYS: (
  LAYDOWN_CONVEYOR_1,
```

```
LAYDOWN_CONVEYOR_2,
PICKING_CONVEYOR_1,
PICKING_CONVEYOR_2,
PALLET_1,
PALLET_2,
(* etc. *));
```

The MC\_KinWcsPcs1/2Assignment function blocks are used for assigning the respectively required transformation from the *TransformArray* to the WCS\_PCS1 and WCS\_PCS2 inputs. You merely have to pass the index (*TransformIndex* input) of the respectively required coordinate system to the function blocks (such as LAYDOWN\_CONVEYOR\_1).

There are corresponding output signals for the handshake between the sequential program in the TaskMain and the execution of the MC\_KinWcsPcs1/2Assignment function blocks in the TaskPriority. The function blocks indicate the currently assigned transformation (*AssignedTransform*) and whether a monitoring time required for a jerk-free switchover has elapsed (*ObservationTimeOver*) since the *TransformIndex* change. Once the required transformation is assigned and the monitoring time has elapsed, switchover of the coordinate system may take place in the motion sequence program in the TaskMain.

The following figure shows the program structure for tracking applications with more than 2 PCS:



12549853579



For further information, refer to chapters "MC\_KinEncoderDataProcessing function block" and "MC\_KinWcsPcs1/2-Assignment function block".

### 13.7 Using a tool transformation

The specified Kinematic models (→ 57) comprise the transformation from the kinematics coordinate system (KCS) to the kinematics flange (FCS). A gripping or processing device is mounted to the flange. With some models, a simple tool (for example offset of the FCS to the suction cup of a vacuum gripper along the FCS Z coordinate) can already be mapped in the kinematics parameters.

With more complex tools, you explicitly have to pass the workpiece or tool transformation of the FCS flange coordinate system to the TCS tool coordinate system to the kinematics controller. The tool center point (TCP) is located in the origin of the TCS. Programmed paths are traveled by the TCP. The transformation is expressed as a vector and describes the relative position of the TCS in relation to FCS. The first 3 components express the lag of the flange center point (FCP) to the TCP. The fourth component corresponds to the rotation of FCS around its Z axis, the fifth component the rotation around the new Y axis, and the sixth component the rotation around the new X axis. You must assign the input variable *AxisGroupKin.Inst[...].In.Transform.Tool* to the vector.

With changing tool transformation, the signal characteristic of the transformation vector may not have any step changes during Cartesian interpolation and 5 cycles (of TaskPriority) before Cartesian interpolation. The reason is that the kinematic control will attempt to compensate these step changes. You have to smoothen noisy sensor signals for the change of the tool transformation appropriately before supplying them to the kinematics controller.

**Note:**

In the event of a tool change, the tool transformation must be switched to the new tool while kinematic control

- is not in interpolation
- or is in axis interpolation.

In the first case, the new transformation is transferred to the interpolation in case of a new change. In the second case, the new transformation is accepted directly, so that the Cartesian position alters rapidly in the same cycle. No lag error occurs because the axes are interpolated at the very moment.

You can use a tool transformation that is variable at runtime to compensate a change of the transformation (for example linear axis to change the length) caused by a drive in the tool.

Another application of a variable tool transformation is the continuous alignment of the tool along the tangent of a path. In the user program, you can determine the tangent cyclically from the current positions and cyclically adapt the tool transformation as required. Application-specific factors come into play, for example, if the path has bends.

### 13.8 Combination of cam disk and kinematics function

Usually, Cartesian gantry kinematics are used if contours are traveled via cam disks that are calculated online or with tables. Curve progressions are determined and traveled in a synchronized manner for all axes positioned at 90° to each other.



You can combine the cam disk function with the kinematic control by supplying the position setpoints to the kinematic control as Cartesian X, Y, Z space coordinate. The position setpoints are usually sent to the axes directly. As in “normal” kinematics operation, the kinematics controller performs the inverse kinematics transformation in order to determine the corresponding axis progressions and controls the axes of any kinematics. As a result, the TCP travels along the required contour.

For this purpose, you must select the *AxisGroupKin.Inst[.].In.Mode = KIN\_TARGET\_CART* mode and copy the position setpoints in the interpolation cycle, that is in TaskPriority, to the *AxisGroupKin.In.MasterPosition.Target* input variable. With the *AxisGroupKin.Inst[.].In.Target.UseMasterPosition* setting, the axes are moved to the position required to travel to the Cartesian setpoint position in each cycle. The curve progressions must be continuous as with conventional cam disk applications.

**Note:**

However, you have the option of resetting the variable *AxisGroupKin.Inst[.].In.Target.UseMasterPosition* to FALSE at any time, so that the kinematic controller takes over path control in a smooth and jerk-free manner and the interpolating kinematic moves to the Cartesian target coordinate.

At any point of time you can also change to operating mode KIN\_TARGET\_AXIS in order to switch to axis interpolation smoothly and jerk-free with the setting KIN\_SYNC\_XYZABC\_OR\_A1TOA6.

In combination with the kinematic controller, as opposed to the conventional cam disk functionality, you can easily travel with complex kinematic contours on moving objects. For this purpose, you must change to the coordinate system KIN\_PCS1/2. Even the kinematics itself can be moved by external axes (variable transformation WCS\_KCS). This means it might make sense to combine the cam disk functionality with the kinematic controller even for a Cartesian gantry.

## 13.9 CP interpolation as slave of an external master profile

Usually, with ContinuousPath interpolation, the kinematic controller controls the path progression. Optionally you can enter the CP path as usual but control the path progression externally, for example via virtual encoder, master axis, or sensor signal.

For this purpose, the input variable *AxisGroupKin.Inst[.].In.Cp.Settings.UseMasterPosition* must be set when changing to a ContinuousPath mode. The position profile is supplied cyclically in the interpolation task to the *AxisGroupKin.Inst[.].In.MasterPosition.Cp* variable. The position profile must be continuous and must not decline.

## 13.10 Continuous change of the kinematics parameters

You can change the kinematics parameters during runtime. For this purpose, you have to set the *AxisGroupKin.Inst[.].Config.CyclicTakeover.Kin\_Par* variable to TRUE. For this level, you must accept the *AxisGroupKin.Inst[.].Config.Kin.Par* array. Cyclically changing parameters can be useful in the following situations:

- Adaptation of arm lengths due to temperature variations
- Adaptation of kinematics parameters (such as angle) due to a high load on the TCP
- Adaptation of parameters of kinematics with a low stiffness depending on the position
- Adaptation of arm lengths due to an integrated drive that does not count to the kinematics axes

### 13.11 Kinematics with more than 6 drives

The kinematic models can comprise a maximum of 6 drives. However, you can control kinematics with more than 6 drives. The following options are available for this purpose:

- If several drives are required to move an axis (for example high torques or indoor cranes), you can configure a kinematic axis as a virtual axis, for example, and move several real MultiMotion axes synchronized with the virtual axis in the Multi-Motion operating mode "Tracking".
- The position progressions of axes that move the entire kinematics can be supplied to the WCS\_KCS transformation. In this way you can travel programmed paths in WCS or PCS1/2, for example. With redundant axes, you can control the external axis to travel the entire kinematics depending on the path progression in WCS, for example. The axis values actually reached are then fed back to WCS\_KCS so that the path is traveled exactly in WCS.
- Cyclical adaption of the tool transformation for a tool with integrated drive
- Cyclical adaptation of all parameters of a kinematics (such as telescope arms moved by drives)

## 13.12 Integrating user kinematics

If none of the models listed in chapter "Kinematic models (→ 57)" corresponds to your kinematics model, you can integrate a user kinematics. The `MPLCKinematics_UserKin` library has to be replaced for this purpose. In this case, we request to contact SEW-EURODRIVE for consultation.

## 13.13 Dealing with ambiguities of the ABC orientation values

The description of a tool orientation via ABC orientation values is ambiguous.

### Examples:

- (A = 90°, B = 0°, C = 0°) corresponds to the same orientation of the tool in space such as (A = 450°, B = 0°, C = 0°). Depending on the start ABC values, a different motion is the result if component-wise interpolation to the target values (A = 150°, B = 0°, C = 0°) is required based on this orientation.
- (A = 90°, B = 90°, C = 90°) describes the same orientation as (A = 0°, B = 90°, C = 0°). Depending on the start ABC values, a different motion is the result if component-wise interpolation to the target values (A = 0°, B = 100°, C = 0°) is required based on this orientation.

### NOTICE



#### Ambiguities

If ambiguities are not taken into account, then rotation in unexpected direction might be the result and consequently cables or hoses could wind up or break.

You have the following options to deal with these ambiguities:

### 13.13.1 One-to-one assignment

Configuration of a one-to-one assignment of Cartesian orientation coordinates and axis values

`AxisGroupKin.Inst[...].Config.General.AxisCartBijection = TRUE`

`AxisGroupKin.Inst[...].Config.General.MapABCToStartUpValuesAtConfig = FALSE`

This is the default setting for many kinematic models.

#### Advantages:

- No problems with ambiguities
- No limitation of orientation angles

#### Disadvantages:

- Not possible for all kinematic models
- Not all degrees of the transformations can be used
- When axes move clear of software limit switches, then only axis interpolation is possible

### 13.13.2 No one-to-one assignment

No one-to-one assignment and no interpolation start with saved orientation values

*AxisGroupKin.Inst[...].Config.General.AxisCartBijection = FALSE*

*AxisGroupKin.Inst[...].Config.General.MapABCToStartupValuesAtConfig = FALSE*

This is the default setting for kinematic models that do not support one-to-one assignment.

#### Advantages

- Possible with all kinematic models
- All degrees of the transformations can be used

#### Disadvantages

- If the axes may have been rotated before the machine was started up and changed to a kinematic mode, then axis-by-axis homing (KIN\_TARGET\_AXIS, Target.CoordSys = KIN\_ACS) is required with a subsequent change to AM\_DEFAULT (*Out.General.Kin.Ready = FALSE*), for example, and another change to a kinematic operating mode.
- If Cartesian orientation values outside the range  $]-180^\circ, 180^\circ[$  must be traveled to, then the previously described homing procedure is required after each change to a kinematic operating mode.

### 13.13.3 Applying saved orientation values

No one-to-one assignment but saved orientation values are applied at interpolation start

*AxisGroupKin.Inst[...].Config.General.AxisCartBijection = FALSE*

*AxisGroupKin.Inst[...].Config.General.MapABCToStartupValuesAtConfig = TRUE*

This is not the default setting.

#### Advantages:

- Possible with all kinematic models
- All degrees of the transformations can be used
- No homing is required if it is certain that the axes have not been rotated, that the motor encoder has not been replaced, and that the MOVI-PLC® has not been replaced since leaving a kinematic mode (*Out.General.Kin.Ready = FALSE*).

#### Disadvantages:

- If the axes might have been rotated before the machine was started up and changed to a kinematic mode, or a motor encoder or MOVI-PLC® was replaced, then axis-by-axis homing (KIN\_TARGET\_AXIS, Target.CoordSys = KIN\_ACS) is required. Afterwards, you must explicitly set the starting orientation using the procedure described below.

#### Procedure for setting a desired starting orientation:

When you change to the KIN\_TARGET\_CART mode (or ContinuousPath with enabled *Cp.Settings.PlusTargetCartABC*) or to another Cartesian coordinate system in the KIN\_TARGET\_CART operating mode, the current orientation ABC values are mapped close to the target ABC values in the first cycle. To enable this mapping, *AxisGroupKin.Inst[...].In.Target.ABCMapping* must be TRUE. In order for the mapping to work reliably component-wise, the gap between the first orientation and the current orientation in the required ABC representation must not be bigger than  $45^\circ$ . As a target in the first cycle, you can for example, determine the current orientation in the required ABC representation and switch to a significantly different orientation in the next cycle.

You can adjust the configuration settings

- Config.General.AxisCartBijection
- Config.General.MapABCToStartUpValuesAtConfig

on the expert page of the configuration so that they meet your requirements (see chapter "Default assignment of input variables (→ 39)").

### 13.14 Storing resolver position values

The AxisGroupKin\_WriteSysPosUserUnitsToDDB\_MX function block in the Axis-GroupControl Kinematics technology module can be used for storing the resolver position values in the MOVI-PLC®. The function block can be executed, for example, in TaskMain. As a complement, you can use the AxisGroupKin\_HomingWithResolver\_MX function block that is used in the start-up sequence in your sequential program to reference the axes with the stored values.

#### INFORMATION



- The brake must be applied if the power supply of the MOVI-PLC® or the inverter is disconnected. If the axis slips with the brake applied or if the axis position changes due to a conversion, and the stored position values are used, then referencing will be incorrect.
- The inverter must be set to reference travel type 5/8 (no reference travel).
- The function is available only for MOVIAXIS® in the current version.

## 14 Troubleshooting

### 14.1 3D simulation

#### 14.1.1 Unable to establish connection

##### Problem

You cannot establish a connection. A red "0" is displayed in the bottom left corner of the simulation window.

##### Remedy

Carry out the following steps one after the other until the problem is solved:

##### Check the communication settings:

- Establish an Ethernet connection between MOVI-PLC® and the simulation PC.

**Note:** 3D simulation is not possible via USB or fieldbus.

Communication with 3D simulation takes place solely via the Ethernet engineering interface (X37 when using MOVI-PLC® advanced, LAN 3 when using MOVI-PLC® power).

- Set the IP address of the engineering interface of the simulation PC in the configuration of the kinematics technology module.

**Note:** You can read the IP address set for the network adapter used in MotionStudio:

- From the menu, choose [Network] / [Configure communication connections].
- Choose "Ethernet", click the [Edit] button and then [Network adapter].

A window opens displaying the properties (IP address) of the available network adapters.

##### Check the firewall settings:

Another reason can be the firewall setting.

The firewall possibly blocks the communication between MOVI-PLC® and simulation PC.

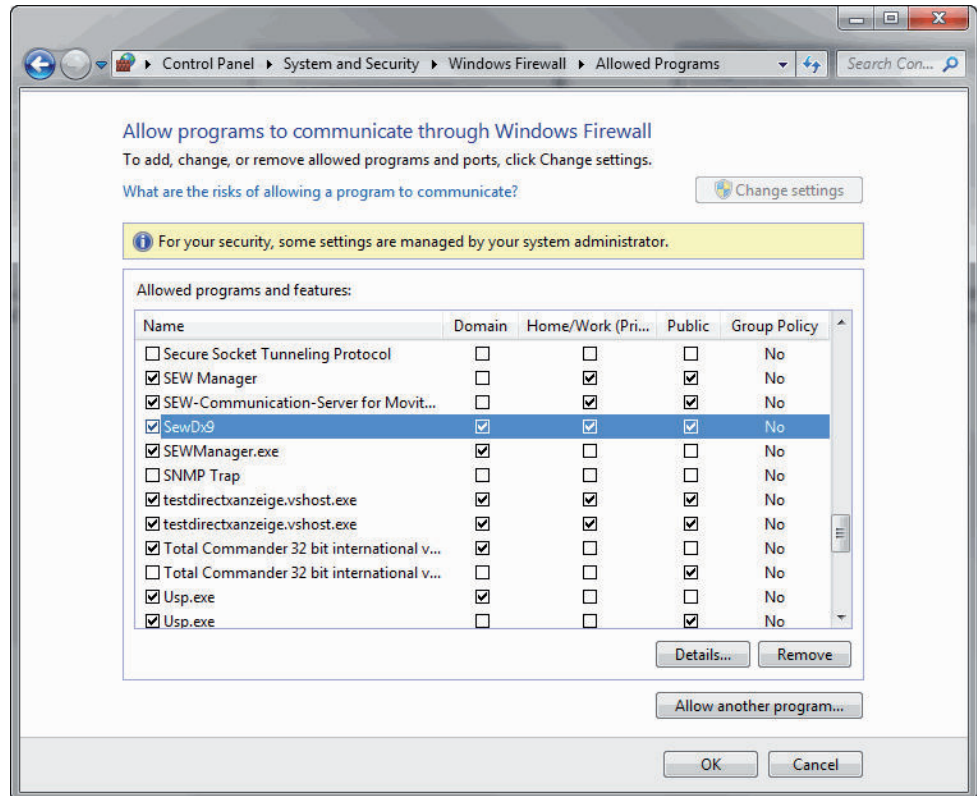
- Deactivate the firewall completely.

Instead, you can create an exception for the 3D simulation program "SEWDx9" using the control panel of your Windows operating system.

- Choose "SEWDx9" from the list.
- If the list does not include this entry, choose the exe file from the following path:

C:\Program Files (x86)\SEW\MotionStudio\SewDx9.exe

- Activate the check boxes "Domain", "Home/Work", and "Public":



9308487307

**Note:** If the entry "SewDx9" appears multiple times in the list, the check boxes for all entries should be activated.

#### Technology level not sufficient

- Ensure that the technology level provided on the memory card is sufficiently high. The 3D simulation cannot be implemented with less than 12 technology points.

The required technology level for the configured functionality can be obtained either from chapter "Required technology level (→ 13)" or from the display in the configuration wizard.

### 14.1.2 No model is displayed

#### Problem

You can establish a connection (indicated by a green number in the bottom left corner of the simulation window) but no model appears.

#### Remedy

- Ensure that the kinematic axes are referenced and that the kinematic configuration can be successfully executed. Possible error messages with information on the source of error appear in the MessageHandler (in the context menu of MOVI-PLC® in the device tree).



### 14.2 Load precontrol

#### 14.2.1 Sagging axes

##### Problem

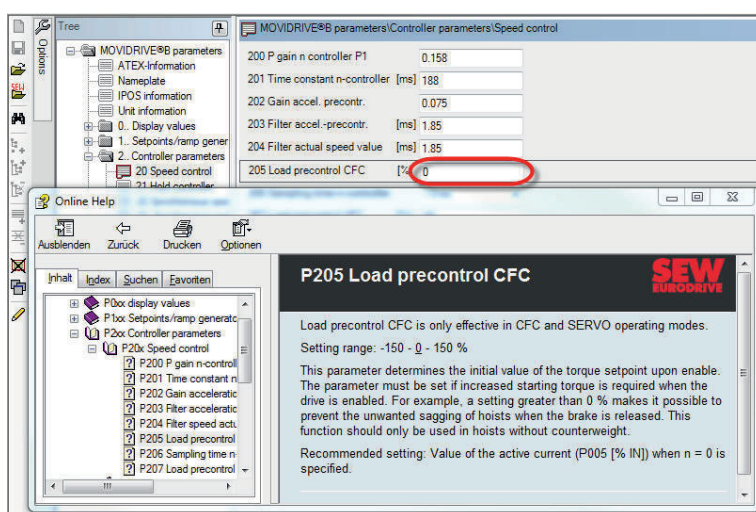
The axes "sag" when they are switched on after emergency stop or inverter inhibit.

##### Remedy

You can set load precontrol to prevent the "sagging" of an axis under load after InverterInhibit or SafeTorqueOff.

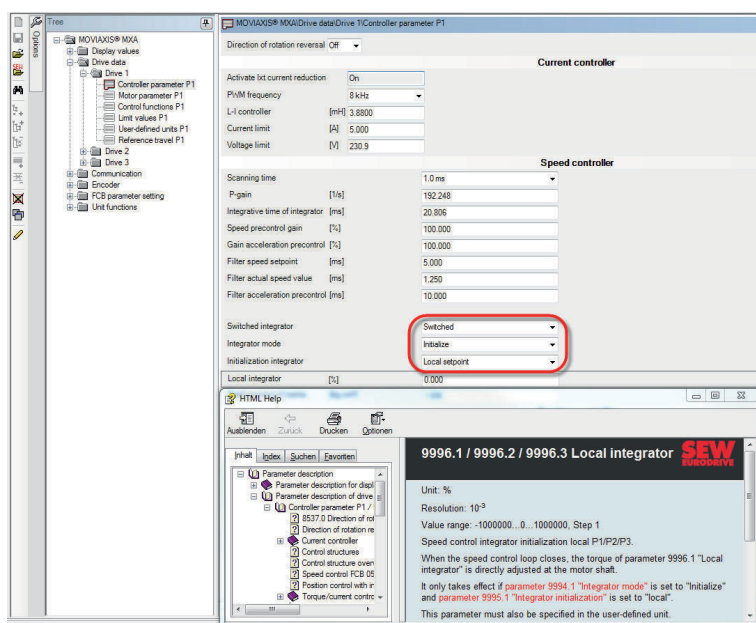
- Set the parameters for load precontrol in the parameter tree of MOVITOOLS® MotionStudio for the device in question.

##### MOVIDRIVE®:



9308586123

##### MOVIAXIS®:



9310111627



### 14.3 Setting inverter parameters

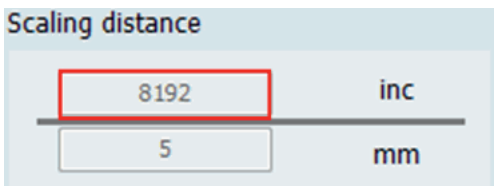
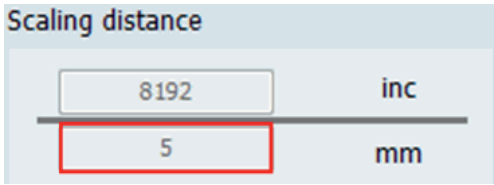
Inverter parameters are set by using wizards.

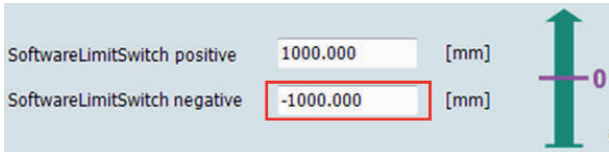
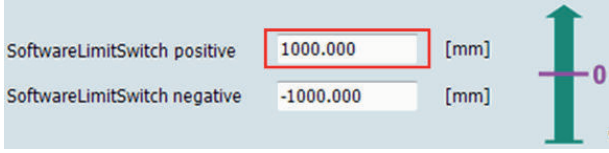

- "DriveStartup for MOVI-PLC®" (see Step 4: Start up the single axes (→ 30))
- as well as "Technology Editor MultiMotion Light (or MultiMotion)" / "Technology Module Kinematics" (see Step 3: Integrate the technology module (→ 28))

For transparency in special applications and situations, this chapter illustrates (for some selected configuration settings) the MultiMotion and inverter parameters to which the set values are sent.

The values of the following variables of the kinematic configuration (file AppConfigA(n).xml) are assigned to the corresponding variables of the MultiMotion configuration (files AxisConfigA(n).xml .. AxisConfigA(n+7).xml), are included in the configuration files of the kinematic technology module as well as the corresponding MultiMotion axes:

- AxisGroupKin.Inst[.].Config.Axis.Numerator[1..8]
- AxisGroupKin.Inst[.].Config.Axis.Denominator[1..8]
- AxisGroupKin.Inst[.].Config.Axis.LimitSwitchNegative[1..8]
- AxisGroupKin.Inst[.].Config.Axis.LimitSwitchPositive[1..8]
- AxisGroupKin.Inst[.].Config.Axis.RapidDeceleration[1..8]

Kinematics	<i>AxisGroupKin.Inst[.].Config.Axis.Numerator[1..8]</i> 	
MultiMotion	<i>AxisInterface.Axis[1..8].Config.General.UserUnits.Numerator</i>	
MOVIAXIS®	User units are always assigned in MX in the following way:	
	Numerator position: 1	9543.1
	Velocity numerator: 1000	9536.1
	Acceleration numerator: 100	9550.1
MOVIDRIVE®	Is not assigned to any parameters for MDX.	-
Kinematics	<i>AxisGroupKin.Inst[.].Config.Axis.Denominator[1..8]</i> 	
MultiMotion	<i>AxisInterface.Axis[1..8].Config.General.UserUnits.Denominator</i>	
MOVIAXIS®	User units are always assigned in MX in the following way:	
	Denominator position: 1	9544.1
	Velocity denominator: 1	9537.1
	Acceleration denominator: 1	9551.1

MOVIDRIVE®	Is not assigned to any parameters for MDX.	-
Kinematics	<i>AxisGroupKin.Inst[...].Config.Axis.LimitSwitchNegative[1..8]</i> 	
MultiMotion	<i>AxisInterface.Axis[1..8].Config.General.LimitSwitch.Negative</i>	
MOVIAXIS®	Is not assigned to any parameters for MX.	—
MOVIDRIVE®	Is not assigned to any parameters for MDX.	—
Kinematics	<i>AxisGroupKin.Inst[...].Config.Axis.LimitSwitchPositive[1..8]</i> 	
MultiMotion	<i>AxisInterface.Axis[1..8].Config.General.LimitSwitch.Positive</i>	
MOVIAXIS®	Is not assigned to any parameters for MX.	—
MOVIDRIVE®	Is not assigned to any parameters for MDX.	-
Kinematics	<i>AxisGroupKin.Inst[...].Config.Axis.RapidDeceleration[1..8]</i> 	
MultiMotion	<i>AxisInterface.Axis[1..8].Config.General.AxisLimits.Enable-StopDeceleration → A</i> <p><b>Note:</b> The Kinematics technology module is not intended for independent deceleration of the axes with the MultiMotion stop ramps (that is by resetting the <i>AxisGroupKin.Inst[...].In.General.Enable_Stop</i> input signal) because this generally causes a path deviation. Instead, you should decelerate by resetting the <i>AxisGroupKin.Inst[...].In.General.Kin.FeedEnable</i> input signal.</p> <p>You can decelerate axis by axis with the emergency stop ramps by resetting the input <i>AxisGroupKin.Inst[...].In.General.Enable_RapidStop</i>.</p>	
	<i>AxisInterface.Axis[1..8].Config.General.AxisLimits.Enable-RapidStopDeceleration → B</i>	
MOVIAXIS®	<b>A</b> → Application threshold, maximum acceleration	9571.1
	<b>A</b> → Application threshold, maximum deceleration	9572.1
	<b>B</b> → Emergency stop (maximum) deceleration	9576.1

MOVIDRIVE®	<b>A</b> → System limit max deceleration [ms] – P131 t11 down CW	8471.0
	<b>A</b> → System limit max deceleration [ms] – P133 t11 down CCW	8473.0
	<b>A</b> → System limit max acceleration [ms] – P130 t11 up CW	8470.0
	<b>A</b> → System limit max acceleration [ms] – P132 t11 up CCW	8472.0
	<b>B</b> → Application limit max deceleration [ms] – P136 t13	8476.0
	<b>B</b> → Emergency stop max deceleration [ms] – P137 t14	8477.0

Some of the selected parameters that were not configured by the kinematics technology module are listed below. The value 0 is assigned to the MultiMotion / inverter parameter.

Kinematics	Not available. → the value 0 is assigned to the MultiMotion/inverter parameter.	
MultiMotion	SystemMaxVelocity	
MOVIAXIS®	System limit maximum positive velocity	9579.1
	System limit maximum negative velocity	9579.10
	Application limit maximum positive velocity	9716.1
	Application limit maximum negative velocity	9716.10
MOVIDRIVE®	System limit max velocity [RPM] – P302	8517.0

Kinematics	Not available. → the value 0 is assigned to the MultiMotion/inverter parameter.	
MultiMotion	SystemMaxAcceleration	
MOVIAXIS®	System limit maximum acceleration	9573.1
MOVIDRIVE®	Is not assigned to any parameters for MDX.	—

Kinematics	Not available. → the value 0 is assigned to the MultiMotion/inverter parameter.	
MultiMotion	SystemMaxDeceleration	
MOVIAXIS®	System limit maximum delay	9574.1
MOVIDRIVE®	Is not assigned to any parameters for MDX.	—

## 15 Error codes

### 15.1 General errors

Error	Meaning	Hex code
E_KIN_GENERAL_INTERNAL_ERROR	Internal error, consult SEW Service.	FD0000
E_KIN_GENERAL_LICENSE_MODE	No license points were consumed for executing the requested operating mode during configuration.	FD0008
E_KIN_GENERAL_MODE	Required operating mode is not permitted in this situation.	FD0010
E_KIN_GENERAL_DIRKIN	Error while executing the direct kinematic transformation.	FD0020
E_KIN_GENERAL_INVKIN	Error while executing the inverse kinematic transformation.	FD0030
E_KIN_GENERAL_AXIS_COUPLING	Error while coupling several axis values of the kinematics to motor positions.	FD0038
E_KIN_GENERAL_WORKSPACE_NO_AUTOGOON	Leaving the permitted work envelope; details in MessageHandler as well as in out variables.	FD0040
E_KIN_GENERAL_MOTORSPEED_LIMIT_NO_AUTOGOON	Exceeding the permissible motor speed; details in MessageHandler as well as in out variables.	FD0050
E_KIN_GENERAL_AXISINCREMENTS_TOOLARGE_NEG	Axis increments near the negative 32-bit limit value.	FD0060
E_KIN_GENERAL_AXISINCREMENTS_TOOLARGE_POS	Axis increments near the positive 32-bit limit value.	FD0070
E_KIN_GENERAL_MESSAGEBUFFER_CORRUPT	<i>MessageBuffer</i> damaged by unauthorized overwriting.	FD0080.
E_KIN_GENERAL_MESSAGEBUFFER_FULL	<i>MessageBuffer</i> filled completely with the current error series.	FD0090
E_KIN_GENERAL_ABC_STEP_AT_COORDSYS_CHANGE	Change of the ABC orientation values while switching between coordination systems due to an ABC singularity in the position.	FD00A0
E_KIN_GENERAL_ABC_STEP_AT_JOG_CART	Changed ABC orientation values in the KIN_JOG_CART mode due to ABC singularity in the position.	FD00B0
E_KIN_GENERAL_ABC_STEP_AT_TARGET_CART	Change of the ABC orientation values in KIN_TARGET_CART mode due to an ABC singularity in the position.	FD00C0
E_KIN_GENERAL_TEST_ENABLE	Function not permitted without test activation.	FD0100
E_KIN_GENERAL_AXISCARTBIJECTION_OUT_OF_AXIS_SWLS	The one-to-one assignment of axis values and Cartesian coordinates is violated because of leaving Axis.SWLS. Move the axes in the permitted work envelope by means of KIN_JOG/TARGET_AXIS.	FD0110
E_KIN_GENERAL_AXISCARTBIJECTION_TRANSFORM	The one-to-one assignment of axis values and Cartesian coordinates is violated due to an unauthorized transformation.	FD0120

## 15.2 Configuration errors

Error	Meaning	Hex code
E_KIN_CONFIG_INTERNAL_ERROR	Internal error, consult SEW Service.	FD1000
E_KIN_CONFIG_NO_CONFIG	The read kinematics configuration is incorrect.	FD1008
E_KIN_CONFIG_IN_PROCESS	A reconfiguration trigger is not permitted while a kinematics configuration is being adopted.	FD1010
E_KIN_CONFIG_CYCLE_TIME	The configured <i>CycleTime</i> is $\leq 0$ ms.	FD1020
E_KIN_CONFIG_LICENSE_GET	The number of license points corresponding to the configured functions cannot be consumed because the memory card does not have enough license points, or because another technology function has already consumed license points so there are not enough license points left.	FD1030
E_KIN_CONFIG_LICENSE_CORRUPT	The license check is damaged.	FD1040
E_KIN_CONFIG_LICENSE_KINTYPE	The licensing of the selected kinematics has not been requested in the configuration.	FD1050
E_KIN_CONFIG_LICENSE_COORD-SYS	The licensing of the use of WCS, PCS1, PCS2 has not been requested in the configuration.	FD1060
E_KIN_CONFIG_KINTYPE	The configured kinematics is not supported.	FD1070
E_KIN_CONFIG_DIRKIN	Error while executing the direct kinematic transformation while the configuration was being adopted.	FD1080
E_KIN_CONFIG_AXIS_DECOUPLING	Error while decoupling several motor positions to axis values of the kinematics.	FD1088
E_KIN_CONFIG_KINPAR_USE	Incorrect configuration of the use of parameters, for example KIN_USE_DEGR for a linear axis.	FD1090
E_KIN_CONFIG_KINPAR_OUT_OF_RANGE	Incorrect parameter values, for example negative arm length.	FD10A0
E_KIN_CONFIG_KINLIMIT_USE	Incorrect setting for the use of a kinematics limit, for example KIN_USE_NONE for an existing limit in the selected kinematic type.	FD10B0
E_KIN_CONFIG_KINLIMIT_MINMAX	Incorrect setting for the software limit switch of the kinematic limits, for example SWLS_Neg > SWLS_Pos.	FD10C0
E_KIN_CONFIG_AXIS_USE	Incorrect configuration of the use of an axis, for example <i>AxisUse</i> = KIN_USE_NONE for an existing axis in the selected kinematic type.	FD10F0
E_KIN_CONFIG_AXISPOS_MINMAX	Incorrect setting for the software limit switch for an axis, for example SWLS_Neg > SWLS_Pos.	FD1100
E_KIN_CONFIG_AXIS_JOGVEL100	Incorrect setting for <i>AxisJogVel100Percent</i> $\leq 0$	FD1130
E_KIN_CONFIG_AXIS_JOGACCDEC	Incorrect setting for <i>AxisJogAccDec</i> $\leq 0$	FD1138
E_KIN_CONFIG_AXIS_RAPIDDEC	Incorrect setting for <i>AxisRapidDeceleration</i> $\leq 0$	FD1140
E_KIN_CONFIG_AXIS_RAPIDJERK	Incorrect setting for <i>AxisRapidJerkTime</i> $\leq 0$	FD1150
E_KIN_CONFIG_NUMERATOR	Incorrect setting for numerator = 0	FD1160
E_KIN_CONFIG_DENOMINATOR	Incorrect setting for denominator = 0	FD1170

Error	Meaning	Hex code
E_KIN_CONFIG_MOTORSPEED_CONVERSION	Incorrect setting for <i>MotorSpeedConversionFactor</i> $\leq 0$	FD1180
E_KIN_CONFIG_MOTORSPEED_MAX	Incorrect setting for <i>MotorSpeedMaxLimit</i> $\leq 0$	FD1190
E_KIN_CONFIG_MOTOR-SPEED_WARN_PERC	Incorrect setting for <i>MotorSpeedWarningPercentage</i> $< 1$	FD11A0
E_KIN_CONFIG_MOTORSPEED_LIMIT_PERC	Incorrect setting for <i>MotorSpeedLimitPercentage</i> $< \text{MotorSpeedWarningPercentage}$	FD11B0
E_KIN_CONFIG_CARTUNIT_USE	Incorrect setting for <i>CartUnitUse</i> , for example KIN_USE_DEGR for a translation dimension.	FD11C0
E_KIN_CONFIG_CART_JOGLEVEL100	Incorrect setting for <i>CartJogVel100Percent</i> $\leq 0$	FD11D0
E_KIN_CONFIG_CART_JOGACCDEC	Incorrect setting for <i>CartJogAccDec</i> $\leq 0$	FD11D8
E_KIN_CONFIG_CART_RAPIDDEC	Incorrect setting for <i>CartRapidDeceleration</i> $\leq 0$	FD11E0
E_KIN_CONFIG_CART_RAPIDJERK	Reserved	FD11F0
E_KIN_CONFIG_CARTPOS_MINMAX	Incorrect setting for the software limit switch for the Cartesian work envelope in KCS, for example SWLS_Neg > SWLS_Pos	FD1200
E_KIN_CONFIG_CART_MODULO_USE	Incorrect setting for <i>CartModuloUse</i> , for example TRUE for a translation dimension.	FD1230
E_KIN_CONFIG_CART_MODULO_RANGE	Incorrect setting for <i>CartModuloUnderfollow/Overflow</i> ; the range does not represent an integer multiple of $2 \cdot \pi$ or $360^\circ$ .	FD1240
E_KIN_CONFIG_CP_RAPIDTRANSD-EC	Incorrect setting for <i>CpRapidTransDeceleration</i> $\leq 0$	FD1250
E_KIN_CONFIG_CP_RAPIDTRANSD-JERK	Reserved	FD1260
E_KIN_CONFIG_AXISCARTBIJECTION_NOT_WITH_MAPABC	<i>AxisGroupKin.Inst[..].Config.General.AxisCartBijection</i> and <i>MapABCToStartupValuesAtConfig</i> MUST NOT be TRUE both.	FD1270
E_KIN_CONFIG_AXISCARTBIJECTION_MODEL_NOT_SUPPORTED	<i>AxisGroupKin.Inst[..].Config.General.AxisCartBijection</i> not supported for the configured kinematic model.	FD1280
E_KIN_CONFIG_AXISCARTBIJECTION_AXIS_SWLS_TOO_LARGE	<i>AxisGroupKin.Inst[..].Config.General.AxisCartBijection</i> not possible because of too large Axis.SWLS areas.	FD1290
E_KIN_CONFIG_AXISCARTBIJECTION_CART_OFFSET_DIRREV	<i>AxisGroupKin.Inst[..].Config.General.AxisCartBijection</i> not possible because of <i>Cart.Offset_KCS</i> or <i>Direction-Reversal_KCS</i> $\neq 0$ .	FD12A0
E_KIN_CONFIG_CART_MODULO_NOT_WITH_AXISCARTBIJECTION	<i>Cart.ModuloUse</i> has to be FALSE, since <i>General.Axis-CartBijection</i> is TRUE.	FD1238

### 15.3 General parameter errors

Error	Meaning	Hex code
E_KIN_GENPAR_INTERNAL_ERROR	Internal error, consult SEW Service.	FD2000
E_KIN_GENPAR_OVERRIDE_CHANGE	Change of <i>Override</i> while <i>Standstill.ActCoordSys</i> = <i>FALSE</i> ; the new override is adopted with standstill.	FD2010
E_KIN_GENPAR_OVERRIDE_SMALL	<i>Override</i> = 0 is not permitted.	FD2018
E_KIN_GENPAR_OVERRIDE_LARGE	<i>AxisGroupKin.Inst[.].In.General.Kin.OverridePercentage</i> > <i>AxisGroupKin.Inst[.].Config.General.MaxOverridePercentage</i> is not permitted.	FD2020
E_KIN_GENPAR_JOGPERC_LARGE	<i>AxisGroupKin.Inst[.].In.Jog.VelocityPercentage</i> > <i>AxisGroupKin.Inst[.].Config.General.MaxJogVelocityPercentage</i> is not permitted.	FD2030
E_KIN_GENPAR_COORDSYS	A change of the coordinate system is not permitted and is not executed in this situation.	FD2040
E_KIN_GENPAR_LICENSE_COORDSYS	The licensing of the use of WCS, PCS1, PCS2 has not been requested in the configuration.	FD2050
E_KIN_GENPAR_MODULO_MODE	<i>AxisGroupKin.Inst[.].In.Target.ModuloMode</i> = <i>KIN_DIRECTION_NIL</i> is not permitted for a Cartesian dimension in modulo mode.	FD2060



## 15.4 Target parameter errors

Error	Meaning	Hex code
E_KIN_TARGPAR_INTERNAL_ERROR	Internal error, consult SEW Service.	FD3000
E_KIN_TARGPAR_VEL	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Velocity</i> < 0.	FD3008
E_KIN_TARGPAR_ACC	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Acceleration</i> ≤ 0.	FD3010
E_KIN_TARGPAR_DEC_SMALL	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Deceleration</i> ≤ 0.	FD3020
E_KIN_TARGPAR_DEC_LARGE	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Deceleration</i> > <i>AxisGroupKin.Inst[.].Config.Axis / Cart.RapidDeceleration</i> , depending on the current KIN_TARGET_AXIS / CART mode.	FD3030
E_KIN_TARGPAR_JERK	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Jerk</i> < <i>AxisGroupKin.Inst[.].Config.Axis / Cart.RapidJerk</i> , depending on the current KIN_TARGET_AXIS / CART mode.	FD3040
E_KIN_TARGPAR_MODULO_OUT_OF_RANGE	Incorrect setting for <i>AxisGroupKin.Inst[.].In.Target.Position</i> > <i>Cart.ModuloUnderflow</i> or <i>AxisGroupKin.Inst[.].In.Target.Axis / Cart.Velocity</i> > <i>CartModulo.Overflow</i> .	FD3050
E_KIN_TARGPAR_OUT_OF_AXIS_SWLS	TARGET_AXIS to Cartesian target position not possible as the axis values corresponding to the target position lie outside the axis SWLS.	FD3060
E_KIN_TARGPAR_AXISCARTBIJECTION_NOT_WITH_MAPABC	<i>AxisGroupKin.Inst[.].Config.General.AxisCartBijection</i> und <i>AxisGroupKin.Inst[.].In.Target.ABCMapping</i> must not both be set to TRUE.	FD3070



## 15.5 ContinuousPath parameter errors

Error	Meaning	Hex code
E_KIN_CPPAR_INTERNAL_ERROR	Internal error, consult SEW Service	FD4000
E_KIN_CPPAR_LICENSE_PLUS-TARGETCART	The licensing of the use of KIN_TARGET_CART for the re-orientation of the TCP during execution of a translation via ContinuousPath is not required in the configuration.	FD4010
E_KIN_CPPAR_UNCLEAR_PLUS-TARGETCART_OR_INCLUDINGABC	No clear selection as to whether ABC should be interpolated during ContinuousPath in synchronized mode or as a parallel process by means of TARGET_CART.	FD4018
E_KIN_CPPAR_BACKTOPATH_AXIS_COUPLING	Error while coupling several axis values of the kinematics to motor positions during a CP BackToPath motion.	FD4028
E_KIN_CPPAR_BACKTOPATH_AXIS_DECOUPLING	Error while decoupling several motor positions to axis values of the kinematics during a CP BackToPath motion.	FD4029
E_KIN_CPPAR_BACKTOPATH_MODE	No repositioning to CP path because no ContinuousPath mode is active.	FD4030
E_KIN_CPPAR_BACKTOPATH_DIST	No repositioning to ContinuousPath path because $AxisGroupKin.Inst[.].In.Cp.BackToPath.MaxInitialDistance < \text{distance to the start of repositioning}$ .	FD4040
E_KIN_CPPAR_PATH_FIDELITY	Incorrect setting for $AxisGroupKin.Inst[.].In.Cp.Path.FidelityPercentage < 90$ or $> 100$ .	FD4050
E_KIN_CPPAR_TRANSPERC_LARGE	$In.Cp.Path.VelocityPercentage > Config.Cp.MaxPathVelocityPercentage$	FD4060
E_KIN_CPPAR_MASTERPOS	The values at the $AxisGroupKin.Inst[.].In.MasterPosition.Cp$ input must not decline.	FD4070
E_KIN_CPPAR_DIRKIN	ContinuousPath segment is not applied due to error while executing the direct kinematic transformation.	FD4080
E_KIN_CPPAR_TRANSVEL	ContinuousPath segment not applied because $AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Velocity \leq 0$ .	FD4090
E_KIN_CPPAR_TRANSACC	ContinuousPath segment not applied because $AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Acceleration \leq 0$ .	FD40A0
E_KIN_CPPAR_TRANSDEC_SMALL	ContinuousPath segment not applied because $AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Deceleration \leq 0$ .	FD40B0
E_KIN_CPPAR_TRANSDEC_LARGE	ContinuousPath segment not applied because $AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Deceleration > Config.CpRapidTransDeceleration$ .	FD40C0
E_KIN_CPPAR_TRANSJERK	ContinuousPath segment not applied because $AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Jerk < Config.Cp.RapidTransJerk$ .	FD40D0
E_KIN_CPPAR_QUEUE_SIZE	Continuous Path segment not applied because the internal queue is completely filled and the corresponding path has not been traveled yet.	FD40E0

Error	Meaning	Hex code
E_KIN_CPPAR_QUEUE_SIZE_REVERSE	Moving reversely close to the queue limitation -> abrupt stop	FD40E8
E_KIN_CPPAR_QUEUE_NOT_FILL_A T_ERROR	Continuous path segment not applied due to existing kinematics error.	FD40F0
E_KIN_CPPAR_PATH_LENGTH	ContinuousPath segment not applied because the total length of the traveled and planned path since the queue was last emptied exceeds the maximum permitted value.	FD4100
E_KIN_CPPAR_LIN_DIST	LIN segment not applied due to length = 0.	FD4110
E_KIN_CPPAR_LIN_BLENDING_DIST	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Translation.Deceleration</i> ≤ 0.	FD4120
E_KIN_CPPAR_LIN_BLENDING_MAX- PERC	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Bleeding.Limitation-Percentage</i> < 1 or < 99.	FD4130
E_KIN_CPPAR_CIRC_MODE	CIRC segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.CircMode</i> is not permitted in this situation.	FD4140
E_KIN_CPPAR_CIRC_PLANE	CIRC segment not applied because the end tangent of the previous ContinuousPath segment is not in the circular plane.	FD4150
E_KIN_CPPAR_CIRC_ANGLE	CIRC segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.CircAngle</i> ≤ 0.	FD4160
E_KIN_CPPAR_CIRC_RADIUS	CIRC segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Radius</i> ≤ 0 is not permitted for the CIRC mode set.	FD4170
E_KIN_CPPAR_CIRC_CENTER_COR- RECTION	Reserved.	FD4180
E_KIN_CPPAR_CIRC_DIRECTION	CIRC segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.CircDirection</i> = KIN_DIRECTION_NIL / SHORT	FD4190
E_KIN_CPPAR_CIRC_DISTORTION	CIRC segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Circ.EllipticDistortion</i> ≤ 0.	FD41A0
E_KIN_CPPAR_ROTATION_VEL	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Rotation.Velocity[j]</i> ≤ 0, j = 4 to 6	FD4320
E_KIN_CPPAR_ROTATION_ACC	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Cart.Acceleration[j]</i> ≤ 0, j = 4 to 6	FD4330
E_KIN_CPPAR_ROTATION_DEC_SMALL	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Cart.Deceleration[j]</i> ≤ 0, j = 4 to 6	FD4340
E_KIN_CPPAR_ROTATION_DEC_LARGE	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Carttion.Deceleration[j]</i> > <i>AxisGroupKin.Inst[.].Config.Cart.RapidDeceleration[j]</i> , j = 4 to 6	FD4350

Error	Meaning	Hex code
E_KIN_CPPAR_ROTATION_JERK	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Cart.Jerk &lt; AxisGroupKin.Inst[.].Config.Cart.RapidJerk[j]</i> , <i>j = 4 to 6</i>	FD4360
E_KIN_CPPAR_AXIS_VEL	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Axis.Velocity[j] &lt;= 0</i>	FD43B0
E_KIN_CPPAR_AXIS_ACC	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Axis.Acceleration[j] &lt;= 0</i>	FD43C0
E_KIN_CPPAR_AXIS_DEC_SMALL	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Axis.Deceleration[j] &lt;= 0</i>	FD43D0
E_KIN_CPPAR_AXIS_DEC_LARGE	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Axis.Deceleration[j] &gt; AxisGroupKin.Inst[.].Config.Axis.RapidDeceleration[j]</i>	FD43E0
E_KIN_CPPAR_AXIS_JERK	ContinuousPath segment not applied because <i>AxisGroupKin.Inst[.].In.Cp.Segment.Axis.Jerk &lt; AxisGroupKin.Inst[.].Config.Axis.RapidJerk[j]</i>	FD43F0

## 15.6 Profile generator errors

Error	Meaning	Hex code
E_KIN_PROFGEN_INTERNAL_ERROR	Internal error, consult SEW Service.	FD5000
E_KIN_PROFGEN_INIT_DELTAT	Incorrect setting for <i>KinProfGen.Init.DeltaT</i> $< 0$ .	FD5010
E_KIN_PROFGEN_INIT_MODULO_RANGE	Incorrect setting for <i>KinProfGen.Init.ModuloUnderflow</i> $\leq$ <i>Overflow</i> .	FD5020
E_KIN_PROFGEN_INIT_OUT_OF_MODULO_RANGE	Incorrect setting for <i>KinProfGen.Init.InitPos</i> outside the range of <i>KinProfGenInit.ModuloUnderflow</i> ( <i>Overflow</i> ).	FD5030
E_KIN_PROFGEN_MODULO_TARGET_OUT_OF_RANGE	Incorrect setting for <i>KinProfGen.TargPos</i> outside the range of <i>KinProfGenInit.ModuloUnderflow</i> ( <i>Overflow</i> ).	FD5040
E_KIN_PROFGEN_MODULO_MODE	Incorrect assignment of <i>KinProfGen.ModuloMode</i> = <i>KIN_DIRECTION_NIL</i>	FD5050
E_KIN_PROFGEN_MODULO_TOO_FAST_POSITIVE	More than the complete modulo range would have to be passed in positive direction in one cycle.	FD5060
E_KIN_PROFGEN_MODULO_TOO_FAST_NEGATIVE	More than the complete modulo range would have to be passed in negative direction in one cycle.	FD5070
E_KIN_PROFGEN_VEL_MAX	Incorrect assignment of <i>KinProfGen.Vmax</i> $< 0$ .	FD5080
E_KIN_PROFGEN_ACC_MAX	Incorrect assignment of <i>KinProfGen.Amax</i> $\leq 0$ .	FD5090
E_KIN_PROFGEN_DEC_MAX	Incorrect assignment of <i>KinProfGen.Dmax</i> $\leq 0$ .	FD50A0
E_KIN_PROFGEN_DEC_RAPID	Incorrect assignment of <i>KinProfGen.D_Rapid</i> $\leq 0$ .	FD50B0
E_KIN_PROFGEN_JERK	Incorrect assignment of <i>KinProfGen.JerkTime</i> $< 0$ .	FD50C0
E_KIN_PROFGEN_JERK_RAPID	Incorrect assignment of <i>KinProfGen.JerkTime_Rapid</i> $< 0$ .	FD50D0

## 15.7 3D simulation errors

Error	Meaning	Hex code
E_KIN_SIMU3D_INTERNAL_ERROR	Internal error, consult SEW Service.	FDA000
E_KIN_SIMU3D_LICENSE_GET	The 10 license points required for 3D simulation cannot be consumed because the memory card does not have enough license points, or because another technology function has already consumed license points so that there are not enough license points left.	FDA010
E_KIN_SIMU3D_LICENSE_CORRUPT	The license check is damaged.	FDA020
E_KIN_SIMU3D_CONNECT_TCP		FDA030
E_KIN_SIMU3D_CONNECT_UDP		FDA040
E_KIN_SIMU3D_BUILDING_SCENE		FDA050
E_KIN_SIMU3D_REG_CYCLIC_TELEGR		FDA060
E_KIN_SIMU3D_RECV_TELEGR_ID_TIMEOUT		FDA061
E_KIN_SIMU3D_RECV_TELEGR_ID		FDA062
E_KIN_SIMU3D_SEND_CYCLIC_TELEGR_TCP		FDA070
E_KIN_SIMU3D_SEND_CYCLIC_TELEGR_UDP		FDA071

## 15.8 AxisGroupControl kinematic errors

Error	Meaning	Hex code
E_KIN_AXISGROUPKIN_INTERNAL_ERROR	Internal error, consult SEW Service.	FDF000
E_KIN_AXISGROUPKIN_READING_AXIS_VALUES_NOT_POSSIBLE	Unable to successfully read the increments from the axes. Check whether the inverter is ready.	FDF010
E_KIN_AXISGROUPKIN_WRITING_AXIS_VALUES_NOT_POSSIBLE	Unable to successfully write the initial values for the MC_KinControl function.	FDF020
E_KIN_AXISGROUPKIN_CONFIGURATION_PRG_DONE_NOT_FALSE	Faulty handshake to the configuration program.	FDF030
E_KIN_AXISGROUPKIN_CONFIGURATION_PRG_DONE_NOT_TRUE		FDF040
E_KIN_AXISGROUPKIN_READ_CONFIG_FROM_SD_CARD_NOT_DONE	Reading of configuration from the memory card not executed.	FDF041
E_KIN_AXISGROUPKIN_CONFIG_IS_NOT_TAKEN_OVER	Kinematics configuration not adopted because the configuration is incorrect or the cyclical task is at standstill, for example.	FDF050
E_KIN_AXISGROUPKIN_CLEAR_LAG_FALSE_NOT_POSSIBLE	Faulty handshake to lower-level AxisControl.	FDF060
E_KIN_AXISGROUPKIN_LAG_IS_NOT_CLEARED		FDF070
E_KIN_AXISGROUPKIN_NOT_ALL_AXES_IN_INTERPOLATION_MODE	Unable to switch one or more axes to interpolation mode (see MultiMotion AxisInterface).	FDF080
E_KIN_AXISGROUPKIN_BACKTOPATH_NOT_ACTIVE	Faulty handshake to MC_KinControl.	FDF090
E_KIN_AXISGROUPKIN_BACKTOPATH_CLEAR_LAG_FALSE_NOT_POSSIBLE	Faulty handshake to lower-level axis driver.	FDF0A0
E_KIN_AXISGROUPKIN_BACKTOPATH_LAG_NOT_CLEARED		FDF0B0
E_KIN_AXISGROUPKIN_BACKTOPATH_NOT_IN_INTERPOLATION	Unable to switch one or more axes to interpolation mode (see MultiMotion AxisInterface).	FDF0C0
E_KIN_AXISGROUPKIN_NUMBER_OF_KIN_AXES_NOT_VALID	Incorrect setting for number of kinematic axes $\leq 0$ or $> 6$ .	FDF0D0
E_KIN_AXISGROUPKIN_NUMBER_OF_AUX_AXES_NOT_VALID	Incorrect setting for the number of auxiliary axes $< 0$ or $> 2$ .	FDF0E0
E_KIN_AXISGROUPKIN_MODE_NOT_VALID	Selected operating mode is not supported by AxisGroupControl Kinematics.	FDF0F0

Error	Meaning	Hex code
E_KIN_AXISGROUPKIN_INSTANCEID_NOT_VALID	The InstanceID set in AxisGroup Configuration is not valid. The InstanceID may not be smaller than 1 and greater than the maximum number of instances (3 .. 12).	FDF100
E_KIN_AXISGROUPKIN_INSTANCEID_DUPLICATE	The InstanceID set in the AxisGroup Configuration was assigned for at least one other instance.	FDF110
E_KIN_AXISGROUPKIN_INSTANCENAME_DUPLICATE	The InstanceName set in the AxisGroup Configuration was assigned for at least one other instance.	FDF120
E_KIN_AXISGROUPKIN_NUMBER_OF_INSTANCES_NOT_VALID	<i>NumberOfInstances</i> is greater than <i>MaxNumberOfAGKInstances</i> .	FDF130

## 16 Glossary and list of abbreviations

Below, you find tables containing a glossary and list of abbreviations.

The glossary includes the most important terms used in this manual and their definitions.

The list of abbreviations includes commonly used abbreviations and the entire terms. You can find the definitions of these terms in the glossary.

### 16.1 Glossary

A	Meaning
ABC representation	Three angles to describe the orientation of a coordinate system based on a reference coordinate system. For transformation, the reference coordinate system is rotated in a mathematically positive direction around the Z axis at the angle "A". Afterwards, it is rotated around the Y axis of the rotated coordinate system at the angle "B". Afterwards, a rotation around the X axis of the re-rotated coordinate system is carried out at the angle "C".
Arm segment	A section in a kinematic chain.
Articulated arm robot	Articulated arm robots have 5 to 6 degrees of freedom and include multiple rotary joints.
Articulated kinematic model	Configured area in which the kinematic model is allowed to move. It is limited by the kinematic model, the kinematic limits, axis limits and Cartesian limits.
Axis arrow	Arrow in 3D simulation that indicates the work envelope (within the software limit switches), the axis zero (0 motor increments), and the direction in which the motor increments increase (with one-to-one assignment of a motor to an axis).
Axis coordinate system (ACS)	A coordinate system in which the position of the kinematic model and the auxiliary axes is described by the axis coordinates.
Axis limits	Limit values for axis coordinates. If the values are exceeded, the kinematic control generally goes to error status.
Axis space	Multi-dimensional mathematical space that is generated by the axes. Interpolation in the axis space means that a profile for the target values is defined for each axis.
Axis speed	Speed of the up to 6 kinematic axes and the 2 kinematic auxiliary axes (rotary axes/linear axes).
Axis zero	Position of an axis with 0 motor increments (with one-to-one assignment of a motor to an axis).
B	Meaning
Base coordinate system (BCS)	Reference coordinate system for direct kinematic transformation, generally in connection with the robot base. The base coordinate system (BCS) can be moved or rotated in the kinematic configuration relative to the kinematic coordinate system (KCS) using the <code>Cart.Offset_KCS</code> variables. Contrary to ACS/KCS/WCS/PCS1/2, BCS cannot be selected for motion control, however is shown in 3D simulation.
Blending	See "blend".



<b>B</b>	<b>Meaning</b>
Blending	Smooth transition of the path progression and path velocity to the next path segment. With ContinuousPath interpolation, the path velocity can be maintained at a constant level in the blending area.
Blending curve	Path segment in the blending range.
Blending motion	Movement in the blending range or the blending curve.
Blending range	Space in which one path segment blends with the next one.
Boot project	A MOVI-PLC® project that is automatically started and run when the motion and logic controller starts up.

<b>C</b>	<b>Meaning</b>
Cartesian	Cartesian refers to the coordinates or the movement along the axes of motion of a Cartesian coordinate system. In a Cartesian coordinate system, the axes of motion (such as X, Y, Z) are perpendicular to each other, meaning they intersect at a 90° angle.
Cartesian coordinate system	At SEW-EURODRIVE, one of the following coordinate systems: KCS, WCS, PCS1, PCS2
Cartesian gantry	Kinematic model in which 2 or 3 linear axes are perpendicular to each other, generating a Cartesian work envelope.
Cartesian limits	Limit values for the Cartesian coordinates of the tool center point (TCP). If the values are exceeded, the kinematic control generally enters the error status.
Circular interpolation	Defining the path as a circular arc from the start to the end point.
Constellation	Position of a kinematic chain with which the TCS takes a determined pose. Generally, kinematic chains can reach the same pose with different positions, known as constellations. For example, the elbow joint of SCARA kinematics can be retracted on either side, which corresponds to two different constellations.
Continuous path mode	Path interpolation mode in which the path is generally described geometrically in space (for example segments of a line/circle and geometrically defined blending ranges).
Coordinate axis	Axis of a Cartesian coordinate system
CP queue	Processing list of CP segments in the kinematic controller.
Curve discontinuity	Discontinuity in the path curvature as a function of path progression.

<b>D</b>	<b>Meaning</b>
Degree of freedom (DOF)	Freely selectable, independent movement options for the kinematic model.
Delta kinematics	Parallel kinematics that are characterized by partial kinematic chains in a triangular configuration.
Distance check	Determining the distance to a target point either based on the shortest possible route in Cartesian space or along a trajectory, as well as checking for values that exceed the limit values. The distance may be translatory or rotary.

<b>F</b>	<b>Meaning</b>
Flange center point (FCP)	Origin of the kinematic model's flange coordinate system.

<b>F</b>	<b>Meaning</b>
Flange coordinate system (FCS)	Cartesian coordinate system generated from the base coordinate system (BCS) using direct kinematic transformation.
Full circle	Motion along the entire circular track. (1 full circle $\triangleq$ 360 degrees)
<b>G</b>	<b>Meaning</b>
Gravity coordinate system (GCS)	Coordinate system where the Z axis points vertically upward in space (against the Earth's gravitational pull). In the default view of 3D simulation, the Z axis points upward and X axis to the right. The orientation of world coordinate system (WCS) can be configured in reference to the gravity coordinate system (GCS).
<b>H</b>	<b>Meaning</b>
Hexapod kinematics	Parallel kinematics characterized by 6 parallel kinematic chains.
Homing	Movement of the kinematic model to its home position. Depending on the application, this may be axis-by-axis or Cartesian movement, and may consist of just one motion task or a series of movements.
<b>I</b>	<b>Meaning</b>
Intermediate point	End point of a path segment that is not yet the last segment of the path.
Interpolation	Defining a certain path between two points.
Interpolation cycle	Cycle used for interpolation, meaning that interpolation points are calculated in it.
Interpolation point table	List of points that define a path.
<b>K</b>	<b>Meaning</b>
Kinematic chain	Series of segments connected by joints.
Kinematic limitation	Limiting within the kinematic model. The spatial work envelope of the kinematic model is defined by the kinematic limits, the limits of the axes, and the Cartesian limits. The concrete meaning of kinematic limits depends on the kinematic model. For example, kinematic limitation can be used to limit the motion of a joint in parallel kinematics that is not directly moved by a drive. If the values are exceeded, the kinematic control generally enters error status.
Kinematic mode	Operating mode in which the axes of the kinematic model are interpolated by the kinematic controller. Unlike with control of individual axes, profile generation takes into account the kinematic model's other axes.
Kinematic model axis	Axis with a zero point and direction of movement according to the definition in the specific kinematic model.
Kinematic zero point	Position of the kinematic model at which all kinematic model axes are at zero.
Kinematics	The kinematic model describes the position and the movement (such as velocity, acceleration) of bodies. In robotics, joints are used to connect bodies to kinematic chains that are often called "kinematic models".

<b>K</b>	<b>Meaning</b>
Kinematics coordinate system (KCS)	Coordinate system that is generally permanently connected to the base of the kinematic model. KSC is defined with reference to the world coordinate system (WCS). The transformation, meaning the position of the robot in the world, can be changed during runtime.
Kinematics error	An error that occurs in kinematic mode.
Kinematics function	Function of the kinematic controller, such as linear interpolation.
Kinematics instance	Instance of the kinematic controller. You can control a kinematic model with a kinematics instance.
<b>L</b>	<b>Meaning</b>
Library manager	A tool in the PLC Editor that allows you to integrate libraries into the MOVI-PLC® project, or to delete them.
Load precontrol	Controller parameter for setting the torque setpoint when enabling a drive to prevent sagging of an axis subject to load.
<b>M</b>	<b>Meaning</b>
Mapping	In the sense of an illustration. Example: Illustration of the current orientation coordinates in an ABC representation of the target values.
Mixed kinematic model	Kinematic model that does not clearly feature the characteristics of any other kinematic model. In particular, this does not refer to the following models: Cartesian gantry, roller gantry, SCARA, delta, quadropod, hexapod or articulated.
Model axes	Axes of the kinematic model in contrast to real axes.
Motion control	Control of the trajectory.
Motion sequence program	Program that supplies the sequence of movements by the kinematic model and other axes to the kinematic controller.
Motion state	State of a movement including the current pose, speed and acceleration of the tool center point (TCP) and all axes.
<b>O</b>	<b>Meaning</b>
Orientation	Rotational alignment of a coordinate system. Specified by rotation relative to a reference coordinate system, for example in an ABC representation.
<b>P</b>	<b>Meaning</b>
Pallet pattern	Arrangement of objects such as packages on a pallet.
Palletizing	Consolidating packages (such as crates) on a carrier (such as a pallet) using a pallet pattern.
Parallel kinematics	A kinematic model in which the kinematic chain between the base coordinate system (BCS) and the flange coordinate system (FCS) consists of a parallel connection of at least two kinematic chains.
Path accuracy	Measure of the geometric agreement of the tool center point's (TCP) path with the kinematic model of the specified path.
Path interpolation	Defining intermediate points between the start point and end point on the specified path.

P	Meaning
Path progression	A distance the kinematic model has traveled along the specified path through the tool center point (TCP) or a rotation of the tool coordinate system (TCS) performed.
Path type	Geometric form of a path in the Cartesian space (such as straight line or circular arc).
Path velocity	Speed at which the tool center point (TCP) of the kinematic model moves along the path.
Pick and place	Picking up and positioning objects.
Pick-up	Control element used to influence the system/machine (such as button for jog mode in positive and negative direction).
Piece coordinate system (PCS1/2)	Coordinate system in a workpiece, such as pallet or object transported on a conveyor. PCS1/2 is defined with reference to the world coordinate system (WCS). The transformation, meaning the position of the workpiece, may be changed at runtime.
Pose	Position (X, Y, Z) and orientation (A, B, C) of the tool coordinate system (TCS). Often simply called the "position".
Q	Meaning
Quadropod kinematics	Parallel kinematics characterized by 4 parallel kinematic chains.
R	Meaning
Rest-in-rest	Motion sequence in which the motion starts and ends at standstill (for example in pick and place applications) – in contrast to a sequence of motion in which an object that is already moving is synchronized with the movement of another object.
Robot base	Base segment of a robot that can be attached to its surroundings (such as the floor, wall or ceiling) or is moved to a linear axis, etc.
Robot flange	Position on a robot for mounting a tool.
Roller gantry	Kinematic model in which two translatory degrees of freedom are generally controlled by stationary drives using a revolving toothed belt. Additional degree of freedom of movement can be added upstream or downstream of this assembly.
Rotation	Rotation; changing the orientation coordinates A, B and C of the tool coordinate system (TCS) of the kinematic model.
S	Meaning
SCARA kinematics	Abbreviation for <b>S</b> elective <b>C</b> ompliance <b>A</b> ssembly <b>R</b> obot <b>A</b> rm. Kinematic chain that, like a human arm, is characterized by shoulder and elbow joints which, in robotics, are two rotary joints with parallel rotary axes.
Smoothing	Reducing a discontinuity such as a curve discontinuity in the trajectory.
Space coordinate	Cartesian coordinate.
Specified target	Specification of the target position and target orientation.
Straight segment	Path segment in continuous path (CP) mode that corresponds to a straight line in Cartesian space.

<b>T</b>	<b>Meaning</b>
Target mode	Kinematics mode characterized by independent or synchronized positioning of axes or Cartesian coordinates at target values.
Target point	The point to which the tool center point of the kinematic model should travel.
Telescope arm	Robot arm with integrated telescope axis.
Telescope axis	Part of a kinematic chain whose length is variable.
Tool center point (TCP)	Origin of the kinematic model's tool coordinate system (TCS).
Tool change	Changing the tool that is attached to the robot flange.
Tool coordinate system (TCS)	Coordinate system in the tool of kinematics. The tool coordinate system (TCS) is defined with reference to the flange coordinate system (FCS). The transformation from FCS to TCS can be changed at runtime.
Tool transformation	Coordinate transformation from the flange coordinate system (FCS) to the tool coordinate system (TCS).
Tracking	Tracking a moved (virtual or real) object with the kinematic model's tool. Typical tracking applications include picking and placing objects from/to conveyor belts and machining objects that are in motion with up to 6 degrees of freedom.  Note: Kinematic tracking applications should not be confused with the MultiMotion "Tracking" operating mode, which is not used by the kinematic control system.
Trajectory	Movement of the tool center point (TCP) of a kinematic model along the specified path. This includes both the path type and the path progression as a function of time.
Transition	Transition between two states.
Translation	Lag; change in the position coordinates X, Y, Z, for example for the tool center point (TCP) of the kinematic model.
Translational velocity	Velocity at which the position of a tool center point (TCP), etc., changes.
Tripod kinematics	Parallel kinematics characterized by a tripod.
Tuple	A tuple is a list of a finite number of objects in a defined order.
<b>U</b>	<b>Meaning</b>
USER kinematic model	From kinematic models that deviate from the standard (such as quadropod, hexapod, articulated, none, etc.) to special designs.
<b>V</b>	<b>Meaning</b>
Velocity profile	Progress of the velocity of an axis, a Cartesian coordinate, the tool center point (TCP), etc. along the trajectory as a function of time.
<b>W</b>	<b>Meaning</b>
Work envelope monitoring	Monitoring for compliance with the configured work envelope limits.
Working envelope	Configured area in which the kinematic model is allowed to move. This area is limited by the kinematic model, the kinematic limits, axis limits, and Cartesian limits.

W	Meaning
World coordinate system (WCS)	Stationary coordinate system relative to the Earth that is independent of the movement of the kinematic model. The world coordinate system (WCS) is the reference coordinate system for the kinematics coordinate system (KCS) and the tool coordinate system (PCS1/2). The world coordinate system (WCS) can be placed on a conveyor belt from which multiple robots are removing objects, for example. The orientation of world coordinate system (WCS) can be configured with reference to the gravity coordinate system (GCS) for correct force/torque calculations and for alignment in the 3D simulation.

## 16.2 List of abbreviations

Abbreviation	Term
ACS	Axis Coordinate System
BCS	Base Coordinate System
CP	Continuous Path
DOF	Degree of Freedom
FCP	Flange Center Point
FCS	Flange Coordinate System
GCS	Gravity Coordinate System
KCS	Kinematics Coordinate System
PCS1/2	Piece Coordinate System
SCARA	Selective Compliance Assembly Robot Arm
TCP	Tool Center Point
TCS	Tool Coordinate System
WCS	World Coordinate System

## Index

### A

Application modules	
Diagnostics .....	41

### B

Bus system.....	11
-----------------	----

### C

Configuration	
Create new.....	30
Control mode.....	43
Copyright.....	9

### D

Danger	
Changing to control mode .....	44
Documents, applicable.....	9

### E

Embedded safety notes .....	8
Exclusion of liability .....	8

### H

Hazard symbols	
Meaning .....	7

### I

Information	
Designation in the documentation.....	7

### L

Liability .....	8
Liability for defects .....	8, 9

### M

Monitor	
mode .....	43
MotionStudio	
Handling.....	26
MOVITOOLS®, see MotionStudio .....	26

### N

Notes	
Meaning of the hazard symbols .....	7

### O

Other applicable documentation .....	9
--------------------------------------	---

### P

Product names .....	9
---------------------	---

### R

Requirements	
Startup .....	26

### S

Safety notes .....	10
Designation in the documentation.....	7
General .....	10
Structure of embedded .....	8
Structure of the section-related .....	7
Section-related safety notes.....	7
Signal words in the safety notes .....	7

### Start

Configuration.....	30
Diagnostics .....	41
Startup	
Procedure .....	25
Requirements.....	26

### T

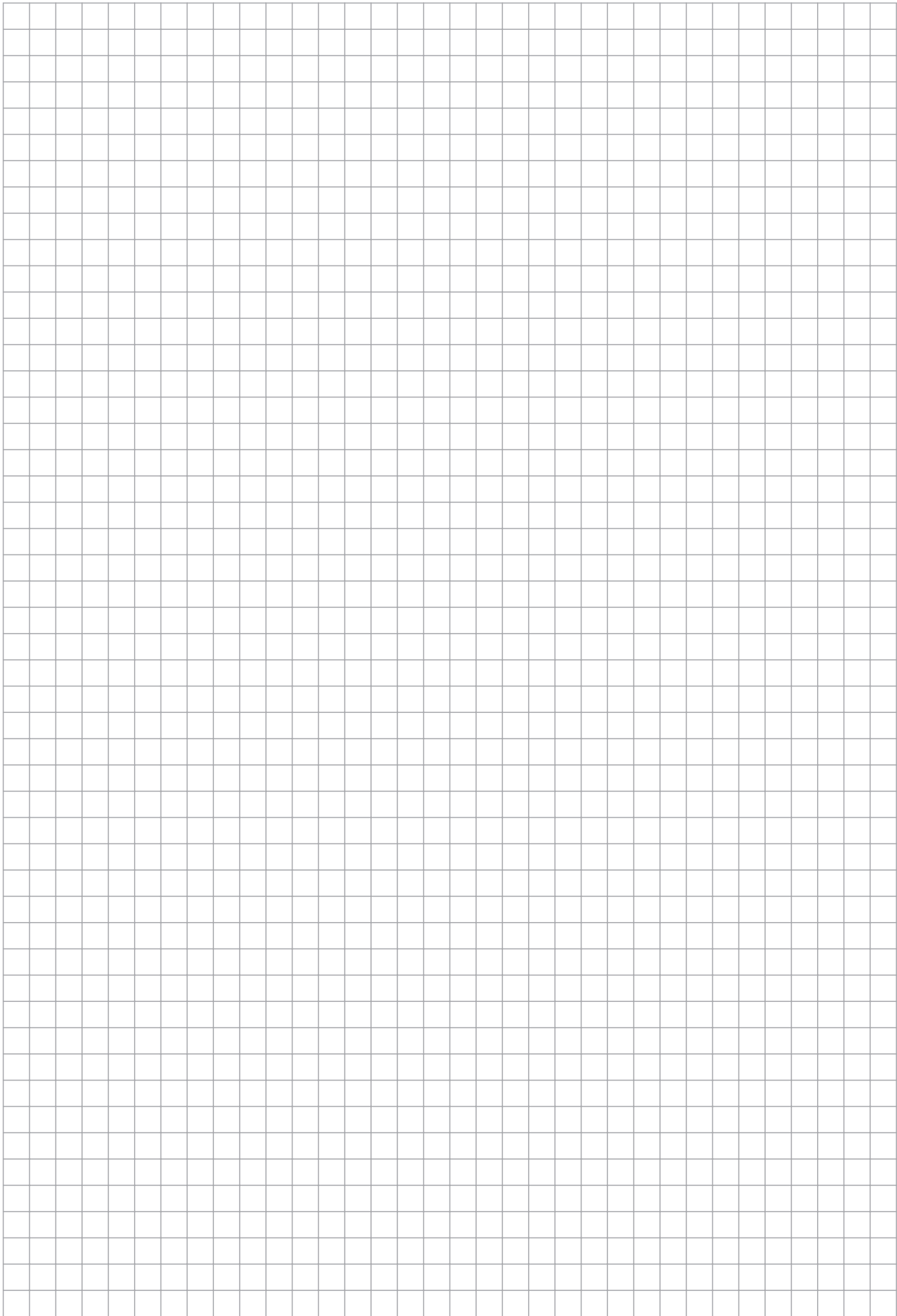
Target group.....	10
Trademarks .....	9

### U

Unintended start-up of the machine.....	44
Use, designated .....	11

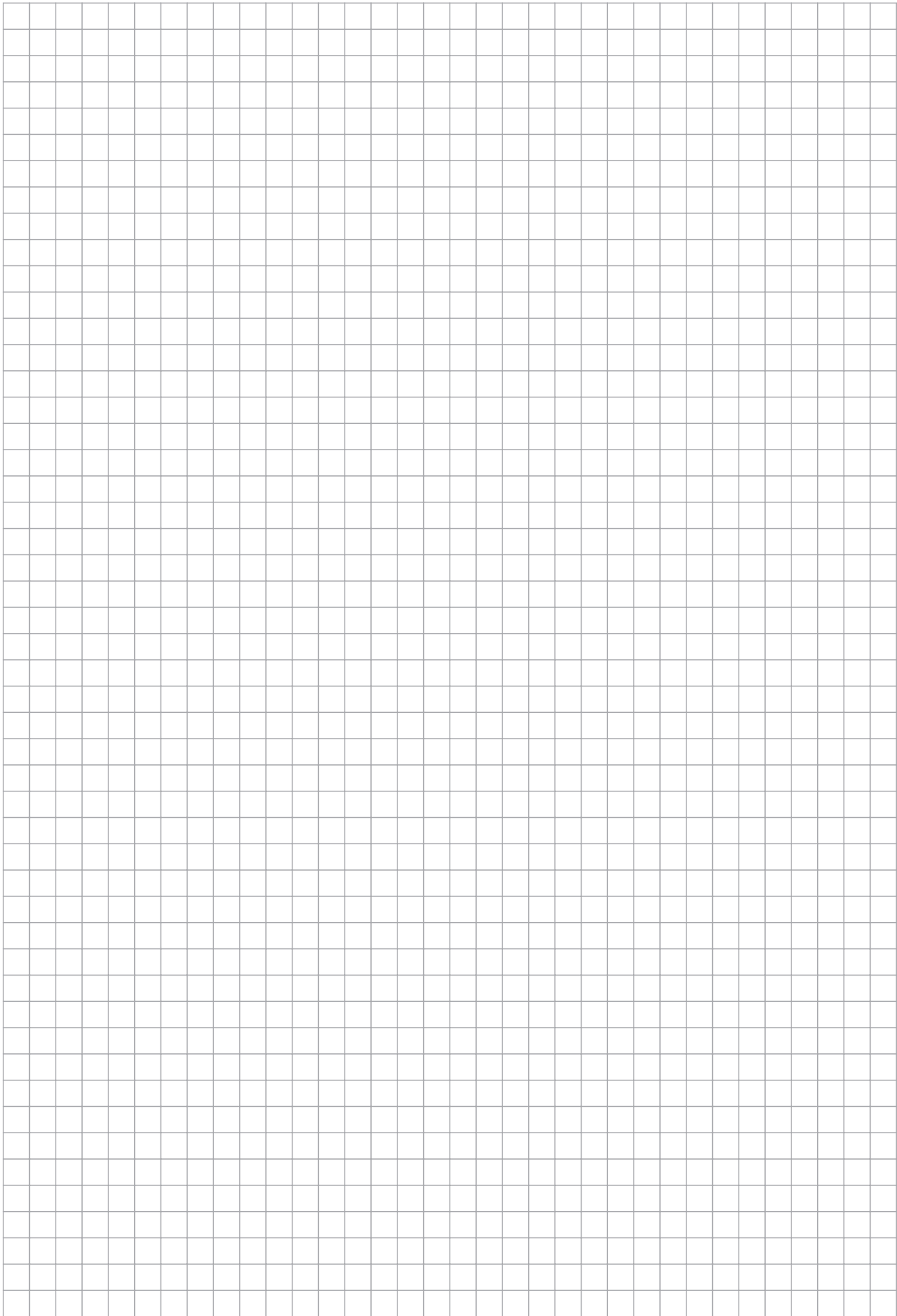
### W

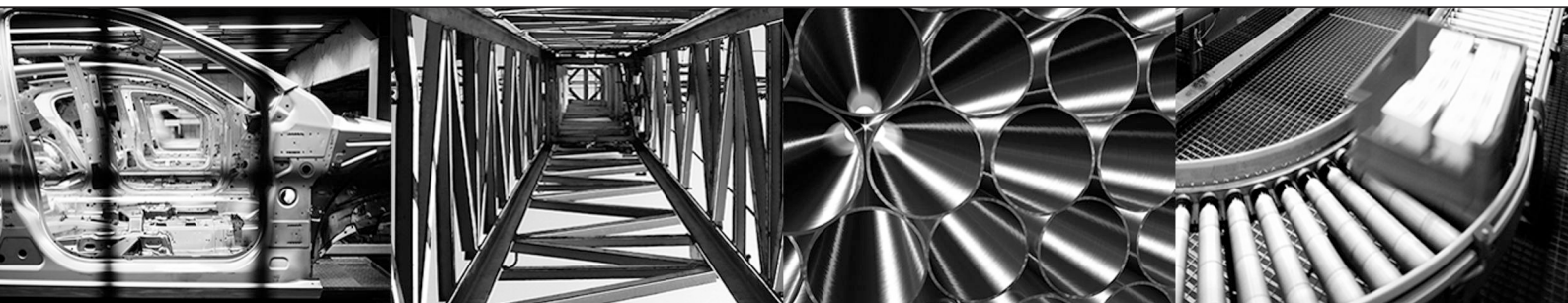
Warning notes	
Meaning of the hazard symbols .....	7













**SEW-EURODRIVE**  
Driving the world

**SEW**  
**EURODRIVE**

SEW-EURODRIVE GmbH & Co KG  
P.O. Box 3023  
76642 BRUCHSAL  
GERMANY  
Phone +49 7251 75-0  
Fax +49 7251-1970  
sew@sew-eurodrive.com  
→ [www.sew-eurodrive.com](http://www.sew-eurodrive.com)