Legal information

Warning notice system
This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠️ DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.

⚠️ WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.

⚠️ CAUTION
indicates that minor personal injury can result if proper precautions are not taken.

NOTICE
indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel
The product/system described in this documentation may be operated only by personnel qualified for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products
Note the following:

⚠️ WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks
All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability
We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.
Preface

Contents

This document is part of the System and Function Descriptions documentation package.

Scope

This manual applies to SIMOTION SCOUT in connection with the SIMOTION Cam, Path or Cam_ext technology package for product version V4.4.

Chapters in this manual

The following is a list of chapters included in this manual along with a description of the information presented in each chapter.

Chapters in this manual

The following describes the purpose and objectives of the manual:

- **Overview of Path Interpolation**
  This chapter contains an overview of the TO functionality and a definition of the terms.

- **Basics of Path Interpolation**
  This chapter explains the basic setting options and functions of the Path Interpolation technology object.

- **Configuring the Path Object**
  This chapter explains the configuration procedure with reference to various tasks.

- **Sample Project for the Path Interpolation**
  This chapter contains a sample project for the path interpolation.

- **Programming/Homing Path Interpolation**
  This chapter explains the commands and functions in greater detail.

- **Appendix A**
  This chapter explains the specific kinematics with TrafoID 1001.

- **Index**
  Keyword index for locating information.

SIMOTION Documentation

An overview of the SIMOTION documentation can be found in the SIMOTION Documentation Overview document.
This documentation is included as electronic documentation in the scope of delivery of SIMOTION SCOUT. It comprises ten documentation packages.

The following documentation packages are available for SIMOTION V4.4:

- SIMOTION Engineering System Handling
- SIMOTION System and Function Descriptions
- SIMOTION Service and Diagnostics
- SIMOTION IT
- SIMOTION Programming
- SIMOTION Programming - References
- SIMOTION C
- SIMOTION P
- SIMOTION D
- SIMOTION Supplementary Documentation

Hotline and Internet addresses

Additional information

Click the following link to find information on the following topics:

- Ordering documentation / overview of documentation
- Additional links to download documents
- Using documentation online (find and search manuals/information)

http://www.siemens.com/motioncontrol/docu

Please send any questions about the technical documentation (e.g. suggestions for improvement, corrections) to the following e-mail address:

docu.motioncontrol@siemens.com

My Documentation Manager

Click the following link for information on how to compile documentation individually on the basis of Siemens content and how to adapt it for the purpose of your own machine documentation:

http://www.siemens.com/mdm

Training

Click the following link for information on SITRAIN - Siemens training courses for automation products, systems and solutions:

http://www.siemens.com/sitrain
FAQs

Frequently Asked Questions can be found in SIMOTION Utilities & Applications, which are included in the scope of delivery of SIMOTION SCOUT, and in the Service&Support pages in Product Support:

http://support.automation.siemens.com

Technical support

Country-specific telephone numbers for technical support are provided on the Internet under Contact:

http://www.siemens.com/automation/service&support
Table of contents

Preface................................................................................................................................................. 3
1 Fundamental safety instructions.................................................................................................................11
  1.1 General safety instructions........................................................................................................ 11
  1.2 Industrial security.................................................................................................................. 12
2 Overview of Path Interpolation .............................................................................................................. 13
  2.1 Overview of Functions........................................................................................................... 13
  2.2 Terminology........................................................................................................................... 15
3 Basics of Path Interpolation................................................................................................................... 19
  3.1 Path interpolation .................................................................................................................. 19
  3.2 Coordinate system.................................................................................................................. 21
  3.3 Modulo properties.................................................................................................................. 22
  3.4 Units.................................................................................................................................... 23
  3.5 Path interpolation types........................................................................................................ 24
    3.5.1 Path interpolation types...............................................................................................................24
    3.5.2 Structure of commands for path interpolation ....................................................................... 25
    3.5.3 Linear paths........................................................................................................................ 26
    3.5.4 Circular paths......................................................................................................................... 27
      3.5.4.1 Circular paths ........................................................................................................................ 27
      3.5.4.2 Circular path in a main plane with radius, end point, and orientation ..........................................28
      3.5.4.3 Circle using center and angle ......................................................................................................29
      3.5.4.4 Circular path using intermediate point and end point ..................................................................31
    3.5.5 Polynomial paths .........................................................................................................................32
      3.5.5.1 Polynomial paths .........................................................................................................................32
      3.5.5.2 Polynomial path - direct specification of the polynomial coefficients ...........................................34
      3.5.5.3 Polynomial paths - explicit specification of the starting point data ........................................ ......34
      3.5.5.4 Polynomial paths - attach continuously .......................................................................................36
    3.5.6 Path dynamics....................................................................................................................... 38
      3.5.6.1 Path dynamics .............................................................................................................................38
      3.5.6.2 Preset path dynamics ..................................................................................................................38
      3.5.6.3 Limiting the path dynamics..........................................................................................................39
  3.6 Stopping and resuming path motion...................................................................................... 42
    3.6.1 Path behavior at motion end.................................................................................................... 43
    3.6.2 Stopping at motion end........................................................................................................... 44
    3.6.3 Blending with dynamic adaptation...........................................................................................44
    3.6.4 Blending without dynamic adaptation.......................................................................................45
    3.6.5 Blending and substitution with insertion of intermediate segments .........................................45
  3.7 Display and monitoring options on the axis........................................................................ 50

TO Path Interpolation
Function Manual, 04/2014
3.14.3.4 Defining the position of the OCS relative to the motion sequence reference value ..........................109
3.14.3.5 Synchronizing motion on the path object to the coupled OCS ..................................................110
3.14.3.6 Performing path motions in the coupled OCS ........................................................................111
3.15 Interconnection, interconnection rules .........................................................................................113
3.16 Simulation operation ....................................................................................................................114

4 Configuring the Path Object ..........................................................................................................115
4.1 Selecting the path interpolation technology package .................................................................115
4.2 Creating axes with path interpolation .......................................................................................116
4.3 Creating a path object ..............................................................................................................117
4.4 Representation in the project navigator .....................................................................................118
4.5 Assigning path object parameters/default values ......................................................................119
4.6 Configuring a path object ........................................................................................................123
4.7 Defining limits ..........................................................................................................................126
4.8 Interconnecting a path object ..................................................................................................127
4.9 Path control panel ....................................................................................................................129
4.10 Configuring kinematic adaptation in the expert list ................................................................130
4.11 Configuring path monitoring ................................................................................................131
4.12 Calibrate path object ..............................................................................................................132
4.13 Path interpolation - context menu ..........................................................................................134

5 Sample Project for the Path Interpolation ......................................................................................135
5.1 Overview of the example ..........................................................................................................135
5.2 Select technology package ......................................................................................................137
5.3 Create axes ................................................................................................................................138
5.4 Creating a path object ..............................................................................................................140
5.5 Defining the kinematics ...........................................................................................................141
5.6 Interconnecting a path object ..................................................................................................142
5.7 Setting the default settings of the path object .........................................................................143
5.8 Programming the path interpolation in MCC .........................................................................145
5.8.1 Programming the travel commands in MCC ......................................................................145
5.8.2 Creating the program ...........................................................................................................146
5.8.3 Programming the traversing program ................................................................................149
5.8.3.1 Programming the traversing program ........................................................................149
5.8.3.2 Creating a WHILE loop ................................................................................................150
5.8.3.3 Programming the A - B linear path ..............................................................................150
5.8.3.4 Programming the B-C polynomial path .......................................................................152
5.8.3.5 Programming the C-D linear path ................................................................................155
5.8.3.6 Programming the D-E polynomial path .......................................................................156
5.8.3.7 Programming the E-F linear path ................................................................................158
5.8.3.8 Programming the F-A return travel ............................................................................158
5.8.4 Activating the axis enables and homing the axes ................................................................160
1.1 General safety instructions

⚠️ WARNING
Risk of death if the safety instructions and remaining risks are not carefully observed
If the safety instructions and residual risks are not observed in the associated hardware documentation, accidents involving severe injuries or death can occur.

- Observe the safety instructions given in the hardware documentation.
- Consider the residual risks for the risk evaluation.

⚠️ WARNING
Danger to life or malfunctions of the machine as a result of incorrect or changed parameterization
As a result of incorrect or changed parameterization, machines can malfunction, which in turn can lead to injuries or death.

- Protect the parameterization (parameter assignments) against unauthorized access.
- Respond to possible malfunctions by applying suitable measures (e.g. EMERGENCY STOP or EMERGENCY OFF).
1.2 Industrial security

Note

Industrial security

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens’ products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com

<table>
<thead>
<tr>
<th>WARNING</th>
</tr>
</thead>
</table>

Danger as a result of unsafe operating states resulting from software manipulation

Software manipulation (e.g. by viruses, Trojan horses, malware, worms) can cause unsafe operating states to develop in your installation which can lead to death, severe injuries and/or material damage.

- Keep the software up to date.
  Information and newsletters can be found at:
  http://support.automation.siemens.com
- Incorporate the automation and drive components into a state-of-the-art, integrated industrial security concept for the installation or machine.
  For more detailed information, go to:
  http://www.siemens.com/industrialsecurity
- Make sure that you include all installed products into the integrated industrial security concept.
2.1 Overview of Functions

As of Version V4.1, SIMOTION provides path interpolation functionality. This functionality enables up to three path axes to travel along paths. In addition, a position axis can be traversed synchronously with the path.

Paths can be combined from segments with linear, circular, and polynomial interpolation in 2D and 3D.

The path interpolation technology is provided by the path object, which represents an independent functionality.

The TO pathObject is interconnected to path axes, and can also be interconnected to a position axis.

The dynamic response parameters are predefined on the path motion.

The path motions of individual path commands can be blended together to form a complete path with no intermediate stop.

The machine kinematics are adapted to the Cartesian axes of the path coordinate system via the kinematic transformation.

As of V4.1.2, the functionality is available for the synchronization of the path motions with an externally specified position value, e.g. with the motion of a conveyor. This supports the system handling for the moved conveyor.

The path interpolation technology contains transformations for the following orthogonal kinematics:

- Cartesian linear axes
- SCARA
- 2D roller picker
- 3D roller picker
- 2D delta picker
- 3D delta picker
- 2D articulated arm
- 3D articulated arm
- 2D swivel arm
- 3D cylindrical robot

As of V4.4, it is also possible to create your own transformations with the aid of the 2D or 3D user function.

During a path motion, a position axis can be traversed synchronously with the path. The axis can approach a programmed, axis-specific target position synchronously or it can execute a motion according to the path length, thus enabling implementation of path-length-based output cams and measuring inputs.
Path interpolation functions are required for such applications as feeding or withdrawal of materials to or from a machine.

The application of commands for individual path segments requires a total path plan in the user program or application.

CNC programming according to DIN 66025 is not supported by SIMOTION.
2.2 Terminology

Axis coordinates
Coordinates of the path axes or the position axis with path-synchronous motion.

Path-axis interface
Interfaces for bidirectional data exchange between the path object and interconnected path axes.

Path axis
Axis that can execute a path motion along with other path axes via a path object.

Path motion
Motion resulting from the interpolation of a path motion command; output on path axes.

Path interpolation
Motion along a path with an assignable dynamic response.
Path interpolation generates the traversing profile for the path, calculates the path interpolation points in the interpolation cycle, and uses the kinematic transformation to derive the axis setpoints for the interpolation cycle points.

Path interpolation grouping
Several path and positioning axes connected by a path object or interpolation.

Path object
The path object provides the functionality for the path interpolation and for other tasks connected with the path interpolation. It also contains the kinematics transformations implemented in the system.

Path control panel
The path control panel enables the control and monitoring of path objects without a user program. It is mainly used to commission kinematic transformations.

Continuous-path control
Motion along a path at a definable velocity.
This can include velocity-based smoothing of the segment transitions by insertion of transition segments.

Basic coordinate system (BCS)
Coordinate system of path interpolation. A clockwise, rectangular coordinate system in accordance with DIN 66217 is used.

Motion sequence
Permits the coupling of the kinematic end point with a coupled OCS and so, for example, the coupling with the actual value of a conveyor. This means, for example, a product can be taken from a running conveyor or placed there.
As of SIMOTION V4.1.2, position details in the motion commands can be related optionally to the basic coordinate system or to an object coordinate system (OCS).

Motion sequence reference value (trackingInPosition)
The value made available to the TrackingIn interface of the path object by another technology object. This can be, for example, the actual value of an external encoder.

Motion sequence value (trackingPosition)
The current position of a coupled OCS with reference to the OCS reference position

Frame transformation
A frame transformation describes the position of a coordinate system relative to another coordinate system that defines, for example, the OCS reference position relative to the basic coordinate system of the path object. The frame transformation consists of translations along the X-, Y-, and Z-axes and rotations at the individual axes.

For the transformation, the displacements are performed first and then the rotations in the following order:
- Roll at the X axis
- Pitch at the (already turned) Y axis
- Yaw at the (already twice-turned) Z axis

Main plane
X-Y, Y-Z or Z-X plane or a parallel plane. The third coordinate is not evaluated.

Interface for path-synchronous motion
Interface for bidirectional data exchange between the path object and an interconnected position axis for path-synchronous motion.

Cartesian axes
Axes X, Y, and Z of the path object

Kinematics
The term "kinematics" in the context of robots and handling devices in motion control systems refers to the abstraction of a mechanical system onto the variables relevant for motion and motion control, i.e. the motion-capable elements (articulations) and their geometric positions relative to each other (arms).

Kinematic transformation, kinematic adaptation
Conversion of specifications in Cartesian coordinates to specifications for individual path axes, and vice versa.

Circular path
Path in 2D or 3D that describes a circle or an arc path.

Linear path
Path in 2D or 3D that describes a straight path.
Coupled OCS
An object coordinate system (OCS) coupled synchronously to the trackingIn interface.

Object coordinate system (OCS)
As of SIMOTION V4.1.2, in addition to the base coordinate system (BCS), object coordinate systems (OCS) with the path object are also available. Path motions can be specified either in the BCS or in the OCS. The object coordinate systems are defined in their reference position using frame transformations for the BCS. They can be coupled with a specified motion value in the x direction of the OCS on the TrackingIn interface.

OCS reference position
Position of the OCS for the motion sequence value equal to zero. The OCS reference position for the BCS is defined using a frame transformation.

Polynomial path
Path in 2D or 3D that describes a polynomial segment.

Synchronous motion, path-synchronous motion
Synchronous coupling of an axis with a path motion; output on a position axis.

TrackingIn interface
The trackingIn input interconnection interface of the path object can be interconnected with another TO that provides an output interface with motion information. This can be, for example, the motion setpoint or actual value of an axis or the actual value of an external encoder.
3.1 Path interpolation

The path interpolation technology provides functionality for interpolating linear, circular, and polynomial paths in two dimensions (2D) and three dimensions (3D).

Objects involved in path interpolation

Figure 3-1 Role and basic principle of the path interpolator

Figure 3-2 Objects involved in path interpolation
The path interpolation technology is made available in the Path Object technology object (TO Path Object).

The TO Path Object is interconnected with 2 or 3 path axes.

In addition, the TO Path Object can be interconnected with a positioning axis for path-synchronous motion and with positioning axes for connection to a coordinate. Likewise, it can be interconnected with a cam.

The TrackingIn interface can be used to interconnect a technology object that provides motion information with a position (the motion sequence value), such as:

- External encoder
- Positioning axis

**Role of the path axis**

All single-axis functions can be executed on the path axis without limitations.

The path interpolation functionality is independent of the physical axis type. Path interpolation can be applied to electric axes, hydraulic axes, and stepper motor axes (real axes) as well as to virtual axes.

**Inclusion of path interpolation in technology packages**

Path functionality is made available in the PATH technology package, which also includes the functionality of the CAM technology package. The extensions include the TO Path Interpolation and the TO Path Axis.

Thus, the CAM_EXT technology package also contains these object types.

For additional information, see *Motion Control Basic Functions*, "Available technology objects".
3.2 Coordinate system

The path interpolation functions require a Cartesian coordinate system. A clockwise, rectangular coordinate system in accordance with DIN 66217 is used.

The user programs in this right-handed system, irrespective of the real kinematics.

![Cartesian coordinate system, right-handed system](image)

Figure 3-5 Cartesian coordinate system, right-handed system

Main planes

It is easy to program two-dimensional motions (2D) directly in one of the three main planes X-Y, Y-Z, or Z-X. In this case, the third coordinate remains constant and does not have to be programmed.

![Main planes in 3D](image)

Figure 3-6 Main planes in 3D
3.3 Modulo properties

Both path axes and positioning axes can be used as modulo axes. However, no modulo range change for the path axis may occur in the path traversal area. The kinematic transformation does not take account of any modulo range change.

Consequently, only one modulo range of the path axis can be used for the traversal area on the path object. The activation of the path interpolation defines the modulo range for the path motion.

This means that the modulo transition of the axis must not be in the traversing range of the path motions. The modulo range and the modulo starting point as well as the position of the modulo range relative to the intended path travel range must be set appropriately, for example, using the settings for reference point and reference point offset during homing.
3.4 Units

All axis-related values are displayed in the quantity and unit of the assigned (interconnected) axes.

The Cartesian coordinates are indicated in a unit of length. The default setting for Cartesian values is [mm].

The default unit for rotary values, such as rotary angle, is [°] and calculated as degrees.

The transformation calculates directly with the numerical values. There is no unit conversion for transformations provided by the system. Thus, the same units must be used for the same base value, e.g., length specification.

Note

Use the same units for all objects associated with the path object that have the same reference quantities (e.g., linear axes in mm, rotary axes in °). Avoid, for example, the mixing of metric and non-metric units for the involved axes.
3.5 Path interpolation types

3.5.1 Path interpolation types

The following interpolation modes are available for the path object:

- **Linear paths** (Page 26)
  - 2D in a main plane
  - 3D

- **Circular paths** (Page 27)
  - 2D in a main plane with radius, end point, and orientation
  - 2D in a main plane with center point and angle
  - 2D with intermediate and end points
  - 3D with intermediate and end points

- **Polynomial paths** (Page 32)
  - 2D in a main plane with explicit specification of geometric derivatives in the start point or with a geometrically continuous attachment
  - 3D with explicit specification of geometric derivatives in the start point or with a geometrically continuous attachment
  - 2D with explicit specification of polynomial parameters
  - 3D with explicit specification of polynomial parameters

The main plane (2D) or the 3D mode in which the path motion occurs can be specified with the `pathPlane` parameter of the interpolation command.

The third path coordinate perpendicular to the main plane is not changed in a 2D path.
3.5.2 Structure of commands for path interpolation

The following path interpolation commands are available:

- Linear interpolation: `_movePathLinear()`
- Circular interpolation: `_movePathCircular()`
- Polynomial interpolation: `_movePathPolynomial()`

These commands contain the following parameters:

- Specification of the object instance in `pathObjectType`
- Specification of the path plane in `pathPlane`
  This parameter is used to set the path plane. The main plane (2D) or the 3D mode in which
  the path motion should occur can be specified.
- Specification of the path mode in `pathMode`
  This parameter is used to set whether the value for the end point is specified as an absolute
  value or whether it is to be evaluated relative to the start point.
- Specification of the end point in `x, y, z`
- Specification of the blending mode in `blendingMode`
- Specification of the merge behavior in `mergeMode`
- Specification of the command transition in `nextCommand`
- Specification of the command ID in `commandId`

Specifications for the linear path only (`_movePathLinear()`) :

(see Linear paths (Page 26) )
- None

Specifications for the circular path only (`_movePathCircular()`) :

(see Circular paths (Page 27) )
- Specification of the circle type in `circularType`
- Specification of the circle direction in `circleDirection`
- Specification of the intermediate point mode in `ijkMode`
- Specification of the intermediate point mode in `i, j, k`
- Specification of the arc angle in `arc`
- Specification of the circle radius in `radius`

Specifications for the polynomial path only (`_movePathPolynomial()`) :

(see Polynomial paths (Page 32) )
- Specification of polynomial mode in `polynomialMode`
- Specification of the vector components in `vector1x to vector4z`

Specifications for the dynamics

(see Path dynamics (Page 38) )
3.5.3 Linear paths

In the case of linear path interpolation, an end point is approached on a straight line starting from the current position.

Linear paths are traversed with the `movePathLinear()` command.
Example of a linear path in ST

In this example, the current position and the end point lie in the X-Y plane. Each end point is separated by 10 units in the positive direction from the current position along both axes.

```
myRetDINT :=
    _movePathLinear(
        pathObject:=pathIPO,
        pathPlane:=X_Y,
        pathMode:=relative,
        x:=10.0,
        y:=10.0
    );
```

3.5.4 Circular paths

3.5.4.1 Circular paths

For a circular path, approach is made from the current position to a specified end point following an arc.

Circular paths are traversed with the `_movePathCircular()` command.

The arc can be specified using several modes. The `circularType` parameter specifies the mode to be used.

- Circular interpolation in a main plane with radius, end point, and orientation (Page 28)
- Circular interpolation in a main plane with center point and angle (Page 29)
- Circular interpolation with intermediate and end points (Page 31)

If a circular path is not traversed because of the geometry, the **50002** error will be issued.
3.5.4.2 Circular path in a main plane with radius, end point, and orientation

To perform circular interpolation in a main plane with specification of radius, end point, and orientation, you set circularType:=WITH_RADIUS_AND_ENDPOSITION in the _movePathCircular() command.

The end point is approached on a circular path starting from the current position. The current position and the end point lie in the same main plane. Circle radius, orientation (travel in the positive or negative direction of rotation), and travel on large or small arcs are specified in the command.

The end point position is entered in the x, y, and z parameters.
Example of a circular path with radius, end point, and orientation

In this example, the current position and the end point lie in the X-Y plane. The end point is separated from the current position by -10 units along the x-axis and 10 units along the y-axis. The large circle is traveled in the positive direction.

Figure 3-10 Example of circular path with radius, end point, and orientation

myRetDINT :=
_movePathCircular(
    pathObject:=pathIPO,
    pathPlane:=X_Y,
    circularType:=WITH_RADIUS_AND_ENDPOSITION,
    circleDirection:=LONG_RUN_POSITIVE,
    pathMode:=RELATIVE,
    x:=-10.0,
    y:=10.0,
    radius:=12.0
);

3.5.4.3 Circle using center and angle

Figure 3-11 Circular path with center point and angle
To perform circular interpolation starting from the current position in a main plane with specification of center point and angle, you set \texttt{circularType:= BY\_CENTER\_AND\_ARC} in the \texttt{_movePathCircular()} command.

The center point of the circle, the angle to be traveled, and the orientation (travel in the positive or negative direction of rotation) are specified in the command.

The position of the center point of the circle is entered in the \texttt{i}, \texttt{j}, and \texttt{k} parameters.

You use the \texttt{ijkMode} parameter to set whether the circle center point coordinates are entered absolutely or relative to the start point or whether the setting in the \texttt{pathMode} parameter should be used.

**Example of a circular path with center point and angle**

In this example, the center point is separated by -10 units from the current position along the X-axis. An angle of 90 degrees in the positive direction is traveled.

```plaintext
retval := _movePathCircular(
    pathObject := pathIpo,
    pathPlane := X\_Y,
    circularType := BY\_CENTER\_AND\_ARC,
    circleDirection := POSITIVE,
    ijkMode := RELATIVE,
    i := -10.0, j := 0.0,
    arc := 90.0
);
```

![Figure 3-12 Example of a circular path with center point and angle](image-url)
3.5.4.4 Circular path using intermediate point and end point

To perform circular interpolation starting from the current position over an intermediate point to the end point, you set `circularType:=OVERPOSITION_TO_ENDPOSITION` in the `_movePathCircular()` command.

The current position, intermediate point, and end point specify the plane for the circular path.

The end point position is entered in the x, y, and z parameters.

The intermediate point is entered in the i, j, and k parameters.

You use the `ijkMode` parameter to set whether the intermediate point coordinates are to be evaluated absolutely or relative to the start point or according to the setting in `pathMode` of the end point.

Example of a circular path with intermediate point and end point

In this example, the end point of the circle is separated by 10 units from the current position in the X-direction. Each intermediate point is separated by 5 units in the X-, Y-, and Z-direction from the current position.

```
retval := _movepathcircular(
    pathObject := pathIpo,
```
3.5 Path interpolation types

3.5.5 Polynomial paths

3.5.5.1 Polynomial paths

A polynomial segment enables you to achieve a constant-velocity and constant-acceleration transition between two geometry elements and to make use of user-programmable curve shapes, e.g. from a higher-level CAD system.

In addition to the implicit start point ($P_S$) of the polynomial, the end point ($P_E$) as well as four three-dimensional vectors for defining the polynomial coefficients are specified in the command parameters of the _movePathPolynomial() command.

The vectors are entered in the command using their components. Thus, for example, vector1 is entered with command parameters vector1x, vector1y, and vector1z.

The polynomial can be defined in three ways:

- Direct specification of the polynomial coefficients (Page 34)
- Explicit specification of starting point data (Page 34)
- Attach continuously (Page 36)

For the two explicit specification of the start point data and attach continuously types, the derivatives at the start and end points of the polynomial are required. They can be determined using integrated functions.

First geometric derivative (tangential vector) \( \dot{P} = \frac{dP}{ds} \) \( |\dot{P}| = 1 \)

Second geometric derivative (vector of curvature) \( \ddot{P} = \frac{d\dot{P}}{ds} \)

Figure 3-15 Polynomial description by specification of the geometric derivatives
Smooth-path transition between two linear paths

The derivatives at the end point of the previous geometry and at the start point of the following geometry can be calculated with the \texttt{getLinearPathGeometricData()}, \texttt{getCircularPathGeometricData()} and \texttt{getPolynomialPathGeometricData()} commands.

If a polynomial path is not traversed because of the geometry, the \texttt{50002} error will be issued.

**Effect of the start and end points**

When polynomials are used, they must be linked smoothly to the previous and subsequent path segment. Depending on the choice of the start and end points, there are consequently different polynomial curves that can deviate significantly from a circular path.

The following graphic shows the curve of a polynomial path with different start points:
3.5.5.2 Polynomial path - direct specification of the polynomial coefficients

For the polynomial specification using \( \text{polynomialMode} := \text{SETTING_OF_COEFFICIENTS()} \) polynomial coefficients, the polynomial path is determined using a function of the fifth degree:

\[
P = A_0 + A_1 \cdot p + A_2 \cdot p^2 + A_3 \cdot p^3 + A_4 \cdot p^4 + A_5 \cdot p^5, \quad p \in [0,1]
\]

- **vector1**: \( A_2 \)
- **vector2**: \( A_3 \)
- **vector3**: \( A_4 \)
- **vector4**: \( A_5 \)

\( A_0 \) and \( A_1 \) result from the start point and end point, and the predefined coefficients. For the parameter area indicated above, this means:

- \( A_0 = \) start point
- \( A_1 = \) end point - start point - \( A_2 \) - \( A_3 \) - \( A_4 \) - \( A_5 \)

3.5.5.3 Polynomial paths - explicit specification of the starting point data

For the \( \text{polynomialMode} := \text{SPECIFIC_START_DATA} \) setting and the explicit specification of the starting point data, the two geometric derivatives at the start point must also be specified for the derivatives at the end point of the polynomial.

The derivatives must be specified as follows:

- **vector1**: First geometric derivative/tangential vector in start point
- **vector2**: Second geometric derivative/curvature vector in start point
- **vector3**: First geometric derivative/tangential vector in end point
- **vector4**: Second geometric derivative/curvature vector in end point

**Example of a polynomial path with explicit specification of the starting point data**

This example connects a linear path and a circular path:

![Diagram](image)

Figure 3-18 Example of a polynomial path with explicit specification of the starting point data

The two derivatives in the starting point of the polynomial must be calculated first. The \_getLinearPathGeometricData() function used for this purpose calculates the two derivatives...
for the end point of the straight line (starting point of the polynomial) using the coordinates of the straight line.

The two derivatives of the polynomial end point are then determined. The 
\texttt{getCircularPathGeometricData()} command used for the calculation uses the starting point of the arc (end point of the polynomial) as basis.

// StartPoly must be defined as
// StructRetGetLinearPathGeometricData
// EndPoly must be defined as
// StructRetGetCircularPathGeometricData
StartPoly :=
\texttt{getLinearPathGeometricData(}
    \texttt{pathObject:=pathIPO,
    pathPlane:=X\_Y,
    pathMode:=ABSOLUTE,
    xStart:=10.0,
    yStart:=10.0,
    xEnd:=20.0,
    yEnd:=20.0,
    pathPointType:=END\_POINT
}\texttt{);} 
EndPoly :=
\texttt{getCircularPathGeometricData(}
    \texttt{pathObject:=pathIPO,
    pathPlane:=X\_Y,
    circularType:=WITH\_RADIUS\_AND\_ENDPOSITION,
    circleDirection:=NEGATIVE,
    pathMode:=ABSOLUTE,
    xStart:=40.0,
    yStart:=20.0,
    xEnd:=50.0,
    yEnd:=10.0,
    radius:=10.0,
    pathPointType:=START\_POINT
}\texttt{);} 
myRetDINT :=
\texttt{movePathPolynomial(}
    \texttt{pathObject:=pathIPO,
    pathPlane:=X\_Y,
    pathMode:=ABSOLUTE,
    polynomialMode:=SPECIFIC\_START\_DATA,
    x:=40.0,
    y:=20.0,
    vector1x:=StartPoly.firstGeometricDerivative.x,
    vector1y:=StartPoly.firstGeometricDerivative.y,
    vector2x:=StartPoly.secondGeometricDerivative.x,
    vector2y:=StartPoly.secondGeometricDerivative.y,
    vector3x:=EndPoly.firstGeometricDerivative.x,
    vector3y:=EndPoly.firstGeometricDerivative.y,
    vector4x:=EndPoly.secondGeometricDerivative.x,
    vector4y:=EndPoly.secondGeometricDerivative.y
}\texttt{);}
3.5.5.4 Polynomial paths - attach continuously

Polynomial paths can be attached continuously to a previous path segment using the polynomialMode:=ATTACHED_STEADILY setting. Because the geometric derivatives at the start point of the polynomial are taken from the predecessor geometry, only the first and the second derivative at the end point needs to be specified directly.

The two derivatives for the polynomial command are specified as following:

- **vector1**: First geometric derivative/tangential vector in end point
- **vector2**: Second geometric derivative/curvature vector in end point

If the geometric derivative cannot be determined in the start point (if no current motion is available), the command is not executed and error message 50002 "Calculation of the geometry element not possible, reason 3" is output.

Example of a polynomial path attached continuously

This example connects two straight lines using a polynomial.

The two derivatives at the end point of the polynomial must be calculated first. The _getLinearPathGeometricData() function used here returns a structure with the derivatives. The function calculates the derivatives for the start point of the straight lines (end point of the polynomial) using the start and end point coordinates of the straight lines.

```c
// EndPoly must be defined as
// StructRetGetLinearPathGeometricData
```
EndPoly :=
  _getLinearPathGeometricData(
    pathObject:=pathIPO,
    pathPlane:=X_Y,
    pathMode:=ABSOLUTE,
    xStart:=30.0,
    yStart:=15.0,
    xEnd:=50.0,
    yEnd:=5.0,
    pathPointType:=START_POINT
  );
myRetDINT :=
  _movePathPolynomial(
    pathObject:=pathIPO,
    pathPlane:=X_Y,
    pathMode:=ABSOLUTE,
    polynomialMode:=ATTACHED_STEADILY,
    x:=30.0,
    y:=15.0,
    vector1x:=EndPoly.firstGeometricDerivative.x,
    vector1y:=EndPoly.firstGeometricDerivative.y,
    vector2x:=EndPoly.secondGeometricDerivative.x,
    vector2y:=EndPoly.secondGeometricDerivative.y
  );
3.6 Path dynamics

3.6.1 Path dynamics

The path dynamics can be specified through preset dynamic values or a dynamic response profile.

The dynamic limits of the individual axes for motion along the path can also be taken into consideration.

An error message is output if the dynamic values are exceeded.

![Diagram showing path dynamics during path interpolation and dynamic limiting on the axis]

3.6.2 Preset path dynamics

The path dynamics can be specified in three different ways in the respective motion command:

- Preset path dynamics via command parameters
- Preset path dynamics via velocity profile/cam
- Preset path dynamics via DynamicsIn

Preset path dynamics via command parameters

The dynamic values (velocity, acceleration, and, if applicable, jerk) are explicitly specified in the velocity profile type.

The path interpolator calculates the velocity profile for the path motion. Criteria for calculating the velocity profile include:

- The dynamic values for velocity, acceleration, and jerk specified in the path motion command
- The type of velocity profile set in velocityProfile:
  - TRAPEZOIDAL: Without jerk limitation; the path will be traveled with constant acceleration and deceleration.
  - SMOOTH: With jerk limitation; the path will be traveled with smooth acceleration and deceleration curve.
Preset path dynamics via velocity profile/cam

The path object can be interconnected with a cam for specifying a velocity profile. Velocity as well as the derived values for acceleration and, if applicable, jerk, are taken from the velocity profile.

The base value (domain) is the path length. To rule out rounding errors in the path length calculation and to enable optimized calculation of profiles over more than one motion, parameters can be programmed simultaneously for the start and end points of the cam domain of the respective motion.

At the command end, the dynamics specified in the profile are also applied to the motion. If additional follow-on motions are programmed, these dynamics are also applied to the transition to the new motion command. Possible settings for the path behavior at the motion end are ignored.

If no additional follow-on motions are programmed or if the motion is to stop at the command end, the dynamics in the profile should be selected such that a stop at the motion end is possible; a velocity of 0 with a braking dynamic that can be achieved with certainty.

In addition, the profile dynamics are limited by the dynamic values for the individual commands, taking into account the preassigned velocity profile type.

Preset path dynamics via DynamicsIn

From V4.3 and higher, the path dynamics can be specified via DynamicsIn. The position specified in the DynamicsIn vector refers to the path/path length. The position must be specified with the velocity and the acceleration at this path point. These values must be provided to the TO cyclically using system variables or TO interconnection.

The dynamic planning and dynamic response adaptation of the TO path is completely deactivated, i.e. there is no limit and no monitoring.

The dynamic limitations of the axes are still effective.

3.6.3 Limiting the path dynamics

Technological limiting

The individual axis setpoints resulting from the path interpolation are limited to the dynamic limits specified for each path axis and positioning axis involved in path-synchronous motion.

The dynamic values of the axis for the path object are only taken into account if this has been programmed accordingly (command parameter blendingMode := ACTIVE_WITH_DYNAMIC_ADAPTION and/or dynamicAdaption <> INACTIVE).

Path velocity limiting, path acceleration limiting, and path jerk limiting can be specified in the limitsOfPathDynamics system variables. Changes in the system variables take effect immediately.

The maximum dynamic values over the path result from the lesser of the dynamic parameters set in the command, the dynamic limits on the path specified via the system variables (limitsOfPathDynamics), and, if programmed, the maximum dynamic values of the axes along the path.
Note that the path velocity for active dynamic adaptation, possibly also reduced, if the dynamic limits of the axes would not be violated even without active dynamic adaptation.

Because the reserve used for the active dynamic adaptation, the maximum possible dynamic path response will not always be attained.

The limitation of dynamic values to the individual axes can lead to dynamic and distance deviations on the path. Path dynamics and axis limits should be set so that the axis limits are not exceeded during path motion.

**Allowance for dynamic limits of path axes**

A reference to the dynamic limits of the axis can be established in the path object using the `dynamicAdaption` command parameter. The following settings are possible:

- **No allowance for maximum dynamic values of path axes (INACTIVE)**
  With this setting, the axial limits are not taken into account within the path interpolation. However, path axis limiting is still active and, if a violation occurs, a setpoint-side path error can result.
  The setting is useful if:
  - There are no transformed dynamic values
  - It can be ensured in advance (e.g. during commissioning) that the axial limit values are not exceeded
  - The axial limits have been taken into account through an application, e.g. through calculation of an optimized velocity profile
  - Superimposed axis motions occur

- **Reduction in the maximum path dynamics according to the maximum dynamic values of path axes (ACTIVE_WITH_CONSTANT_LIMITS)**
  The velocity and acceleration of the path is limited in the path interpolator to the maximum values in the Cartesian coordinates calculated from the maximum value settings of the individual path axes.
  Axis-specific jerk limits in the preliminary path plan are not taken into account. However, the jerk can be limited by specifying the pathMotion monitoring on the path axis accordingly.
  This can result in a setpoint-side path error.
  If the dynamic limits of an axis are reached, i.e. if the programmed path velocity/acceleration cannot be achieved due to these limits, an alarm will be issued.
  If the dynamic limits of the path axes are changed online, the changes take effect immediately but not for the currently active or decoded motion command.

- **Segment-by-segment reduction in the maximum path dynamics according to the maximum dynamic values of path axes in these segments (ACTIVE_WITH_VARIABLE_LIMITS)**
  This setting is equivalent to `ACTIVE_WITH_CONSTANT_LIMITS`, except that the path is segmented. Overall, the path is travelled faster; the velocity is not constant over the entire path.

From system variable `kinematicsData.transformationsOfDynamics` of the path object, you can read out whether the maximum dynamic values of the axis are transformed values. If not, the path dynamics are always limited with the path object dynamic limits, regardless of the setting in the `dynamicAdaption` command parameter.
Difference between ACTIVE_WITH_CONSTANT_LIMITS and ACTIVE_WITH_VARIABLE_LIMITS

The following trace shows the difference between ACTIVE_WITH_CONSTANT_LIMITS and ACTIVE_WITH_VARIABLE_LIMITS. Two circular paths are traveled with a 2D portal; the maximum velocities of the path axes follow:

- **Axis_X**: 500 mm/s
- **Axis_Y**: 200 mm/s

A path velocity of 400 mm/s is defined in the path commands.

![Figure 3-22 Example: Limiting the path dynamics](image)

**Override**

A velocity override (system variable `override.velocity`) and an acceleration override (system variable `override.acceleration`) are available on the path object.

![Figure 3-23 Trace: ACTIVE_WITH_CONSTANT_LIMITS, ACTIVE_WITH_VARIABLE_LIMITS](image)
3.7 Stopping and resuming path motion

The _stopPath() command can be used to stop the current path motion. A stopped, but not canceled, path motion can be continued with the _continuePath() command.

When the path motion is resumed, the motion properties (velocity profile, acceleration, etc.) of the interrupted path command are applied. With SIMOTION V4.2 and higher, other dynamism parameters can be specified directly at the command _continuePath().

In the case of canceled path motions, if you want the application to start at the abort position, the last calculated setpoint position on the path is indicated in the abortPosition system variable.

Dynamic response for _stopPath()

The _stopPath() command can be used to define the dynamic response during deceleration. If the braking dynamic in the _stopPath() command is smaller than the braking dynamic in the active motion command, faults can occur in some situations.

If the dynamic response defined in the _stopPath() command in the previously defined path segments (i.e. in the path segment of the active command or in the path segment of the buffered commandPuffer) can be used to stop the path object, the path object will stop with error.

If the dynamic response defined in the _stopPath() command cannot stop the path object by the end of the previously defined path, the following can occur:

- The path interpolation is terminated.
- Each axis is delayed using the maximum dynamic response defined in the axis.
- The 50006 error message is generated.

As an example: The following path consists of three motion commands that blend in each other. This means, if the first command is active, the second command will be placed in the buffer. If the second command is active, the third command will be placed in the buffer. If the third command is active, it will be completed.

1.) _movePathLinear()

2.) _movePathCircular()

3.) _movePathLinear()

Figure 3-24 Dynamic response for _stopPath()

If _stopPath() with reduced dynamic response is called within the first linear path segment, the path object will be delayed using the dynamic response defined in the _stopPath() command. Under some circumstances, the path object stops in the circle section, it remains, however, on the path profile.

If _stopPath() with reduced dynamic response is called within the second linear path segment, the path object cannot stop before the end of the defined path. The axes are delayed with maximum dynamic response, the path profile may possibly by left, the 50006 error will be issued.
3.8 Path behavior at motion end

3.8.1 Path behavior at motion end

If the path dynamics are specified via a velocity profile, the behavior at the motion end is determined from the dynamics specified in the profile at the path end point.

If the path dynamics are specified via dynamic response parameters, the transition can be set. In addition to stopping at the end command, two sequential path segments can be dynamically linked together so that they do not have to be decelerated, see also *Axis Manual, Positioning with Blending section*.

No intermediate segments for the fillet are generated by the path interpolation for this blending.

Taking into account the axial limits, there are three transition types that can be set in the `blendingMode` parameter of the next command.

- Stopping at motion end (Page 44) (`blendingMode:=INACTIVE`)
- Blending with dynamic adaptation (Page 44) (`blendingMode:=ACTIVE_WITH_DYNAMIC_ADAPTION`)
- Blending without dynamic adaptation (Page 45) (`blendingMode:=ACTIVE_WITHOUT_DYNAMIC_ADAPTION`)

The `blendingMode` parameter is only evaluated if the command is programmed with `mergeMode:=SEQUENTIAL` or `mergeMode:=NEXT_MOTION`.

The blending mode is specified in the motion command in which blending is to be performed.

Dynamic planning is executed by means of two motion commands. In SIMOTION V4.3 and higher, dynamic planning can be set by means of three motion commands (current, next and second motion command) (default setting when creating a new path object). This allows short intermediate commands to be blended without velocity reduction.

---

Figure 3-25 Dynamic planning via 3 motion commands
3.8.2 Stopping at motion end

The motion is ended in the target position of the path command. The path velocity and acceleration is zero. Any new path motion becomes active only after END_OF_INTERPOLATION (end of setpoint generation).

3.8.3 Blending with dynamic adaptation

During blending, the system supports a constant-velocity transition (with velocity profile type TRAPEZOIDAL) or a constant-velocity and constant-acceleration transition (with velocity profile type SMOOTH).

![Path velocity vs time diagram](image)

Figure 3-26 Example of blending with dynamic adaptation: Straight line - straight line

With this setting, the dynamic limits of the axis are taken into account directly when calculating the travel profile for path blending.

The axial limits for velocity and acceleration are also taken into account in the blending velocity.

For non-tangential path transitions (corners), the path velocity is reduced such that a velocity jump greater than the maximum acceleration does not occur for any of the participating axes. The result is a velocity-dependent smoothing of the path end point.

Note that with active dynamic adaptation, the dynamic axis response is set to the smaller value from axis acceleration and axis deceleration. Therefore, when an axis has a maximum acceleration of 1000 mm/s² and a maximum deceleration of 500 mm/s², the value for the deceleration is used for the calculation.
3.8.4 Blending without dynamic adaptation

With this setting, the dynamic limits of the axis are not taken into account in path blending. The path velocity is controlled as a scalar variable that is independent of direction and curvature.

A non-tangential attachment of path segments has no effect on the path velocity profile; for this reason, the velocity is not reduced during blending.

Because the setpoints that are generated for the individual axes are limited to the axis-specific dynamic limits for the axes, this can result in an axis setpoint error relative to the setpoint from the path interpolation. This ultimately leads to an axis-specific deviation from the path in the blending range.

For example, this mode is applicable if the dynamic limits of the axes are to be adhered to on the path (when approaching positions, for example) but an axis-specific axis setpoint error relative to the path is acceptable at the segment transitions in the blending range.

3.8.5 Blending and substitution with insertion of intermediate segments

Blending with insertion of blending segments

When blending with insertion of transition segments, a blending segment is inserted between the two path segments. Either circular or polynomial segments can be inserted. As an alternative, an exact stop or blending can be carried out on the transitions without changing the path geometry.

Transition segments can even be inserted in substitute path motions.

Blending segment:

A polynomial segment can serve as a blending segment. A circular segment is also possible between two linear sets.

The command transition is the start of the blending. The path length of the blending segment and "remaining segment" is fully output on the 2nd motion command.
Blending with insertion of a polynomial segment:
Blending with insertion of a polynomial segment is possible between all path types. The transition between the segments and the polynomial path is consistent in terms of position, velocity and acceleration.

Blending with insertion of a circular segment:
Blending with insertion of a circular segment is only possible between 2 linear sets. The transition between the linear segments and the circuit is consistent in terms of position and velocity.

To determine the circular blending segment, the starting point for the blend segment (SU), the end point of blend segment (EU) and the blend radius, beginning with the common start and end point (SE) of segments 1 and 2, are required (see figure below). The distances between SE and SU, as well as SE and EU, correspond to the programmed blending clearance a. When blending, clearance is not the geometric distance between the points, but rather path lengths of segments 1 and 2 starting from SE (only relevant for non-linear segments). As in the past, the end point SE is programmed as target point of segment 1.

The insertion of a circular segment is rejected by others as linear path commands and a technological alarm is issued:
“50013 blending segment not possible, reason 2: Circular blending segment can only be inserted between linear sets”.

Substitution with insertion of transition segments
Behavior prior to V4.3
With mergeMode:=IMMEDIATELY, there is immediate replacement of the current path motion by the new path motion with the dynamic response parameters of new motion command, regardless of the setting in the blendingMode parameter.

Behavior as of V4.3
The system allows the insertion of transition segments even in substitute path motions. In this case, the settings for the replacement in blendingMode and transitionType are effective. With the setting blendingMode:=INACTIVE the substitute behavior as prior to V4.3 is effective. With the setting blendingMode:=ACTIVE_WITHOUT_DYNAMIC_ADAPTATION or
blendingMode:=ACTIVE_WITH_DYNAMIC_ADAPTION the behavior depends on the setting in parameter transitionType:

- transitionType:=DIRECT: (Compatibility mode, default setting) Substitute behavior as prior to V4.3, no transition segment, direct transition;
- transitionType:=STOP: Delay of the active path motion to standstill, start the new path motion; the dynamic response values of the new path motion are immediately effective, i.e., already also effective for the stop motion;
- transitionType:=POLYNOMIAL | CIRCULAR: A transition segment is inserted by the system, starting from the current position on the path; the blending distance to the current path point is applied.

![Diagram showing blending when substituting commands](image)

Figure 3-29 Blending when substituting commands

The virtual blending point formed is used as the end point of the current motion and as the starting point of the newly programmed motion. Blending with polynomial segment or circular segment is possible depending on the specifications.

The following applies here:

- Circular segment is possible if a linear set is blended,
- Polynomial segment is always possible

**Blending dynamics**

The blending velocity in the area of the blending geometry can be traversed with reference to the first or second path command. The selection is determined by setting the higher or the lower velocity of the two commands to run.

Acceleration and jerk for the velocity transition are assumed by the second path command.

**Blending with mergeMode = SEQUENTIAL / NEXT_MOTION / IMMEDIATELY, transitionType:= DIRECT / STOP / POLYNOMIAL / CIRCULAR and transitionVelocityMode:= HIGH VELOCITY / LOW VELOCITY**
Extended behavior in substitute motions

An additional mode is available in substitute path motions. This mode first stops the substituted motion and then starts the new motion in the stop position. The result is a geometrically variable change point for the new substitute command.

Figure 3-30  Behavior in the case of a stopped substitute motion

Overlap of blending distance

The behavior during detection of an overlap of blending distances can be configured as shown in the following diagram.

When blending several segments, overlapping blending distances may occur where necessary:
If an overlap of the blending distances is detected, the blending radius is reduced by the system to the maximum value (e.g. for \( b \) to \( \text{MAX}(s-a, s/2) \)), and the warning "50013 blending distance modified" is output in the alarm window. The alarm can be deactivated.
3.9 Display and monitoring options on the axis

Display and monitoring options for path motion on the axis
An active path motion is indicated on the path axis in system variable pathMotion.state.

Display of path-synchronous motion on the positioning axis
An active synchronous axis motion is indicated on the positioning axis in system variable pathSyncMotion.state.

Monitoring for setpoint error
The path axis or positioning axis can be monitored for setpoint errors (discrepancy between the setpoint specified by the path object and the setpoint output on the axis).

The difference between the setpoint and the actual value is not monitored.

Limiting and monitoring the setpoint error:

- With setting enableCommandValue := NO_ACTIVATE:
  - The dynamic limitation is performed without taking the jerk into account.
  - The resulting setpoint error is not monitored.

- With setting enableCommandValue := WITHOUT_JERK:
  - The dynamic limitation is performed without taking the jerk into account.
  - The resulting setpoint error is monitored.

- With setting enableCommandValue := WITH_JERK:
  - The dynamic limitation is performed taking the jerk into account.
  - The resulting setpoint error is monitored.

<table>
<thead>
<tr>
<th>Path motion on the path axis</th>
<th>Synchronous motion on the positioning axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation of monitoring (configuration data)</td>
<td>pathAxisPosTolerance. enableCommandValue</td>
</tr>
<tr>
<td>Tolerance value (configuration data)</td>
<td>pathAxisPosTolerance. commandValueTolerance</td>
</tr>
<tr>
<td>Alarm when violation occurs</td>
<td>40401 Tolerance of the axis-specific path setpoints exceeded</td>
</tr>
<tr>
<td>Setpoint errors exceeded (system variable)</td>
<td>pathMotion. limitCommandValue</td>
</tr>
<tr>
<td>Setpoint discrepancy between path object specification and axis output value (system variable)</td>
<td>pathMotion. differenceCommandValue</td>
</tr>
<tr>
<td>Relevant path object (system variable)</td>
<td>pathMotion.activePathObject</td>
</tr>
</tbody>
</table>
3.10 Allowance for axis-specific traversing range limits

The traversing range limits of the path and positioning axes, i.e. active software limit switches, are taken into account in the participating axes and not in the path object.

If a participating axis detects a possible violation of its axis-specific working area, an alarm is triggered along with an appropriate error response.
3.11 Behavior of path motion when an error occurs on a participating path axis or positioning axis

If an error occurs on a path axis or the positioning axis for path-synchronous motion causing the axis motion to stop and the command to be canceled, the path interpolation is canceled and the specified error response is performed.

See Local alarm response (Page 176).

The other axes participating in the path motion travel to velocity 0.0 with the maximum dynamic values.
3.12  Functionality of path-synchronous motion

3.12.1  Functionality of path-synchronous motion

A path-synchronous motion on a positioning axis can be specified synchronous to the path motion with which it is specified. This causes the path-synchronous motion to start and end at the same time as the path motion. This enables a gripper to rotate in synchronism with the path motion, for example.

The path motion and the path-synchronous motion follow a common traversing profile. This also applies to the blending between two path segments.

The current path length can be output via w or w2. This is set with parameters in the path command.

3.12.2  Specification of path-synchronous motion

There are several options for path-synchronous motion, which are specified in the wMode parameter of the respective motion command:

- Movement to a defined end point of the path-synchronous axis w
  The target position of the path-synchronous motion is specified in the path command. This can be a relative (RELATIVE) or absolute (ABSOLUTE) position.
  As for the positioning command of the axis, the direction of the synchronous motion is specified using a parameter (wDirection).
  For further information, refer to the Motion Control, TO axis, electrical / hydraulic, external encoder Function Manual, "Positioning".
  The motion dynamics conform to the path, and the axis is "carried along". If the maximum dynamic values of the positioning axis are thereby violated, the dynamic parameters of the path are reduced accordingly.
  If the path length is zero and a path-synchronous motion is programmed, an error will be issued and the path-synchronous motion set to the programmed end position. The resulting setpoint jump is traversed axially with the maximum values.
  In this case, it is important to note that configured monitoring of the setpoint error of the path-synchronous axis also affects the setpoint jump.

- Movement according to the current path length via w or w2
  The current path distance is output. There are two ways of doing this:
  - Reference to the command (OUTPUT_PATH_LENGTH)
    The axis position is first set to 0.0 before the path distance is traveled.
    The reset of the axis position to zero is equivalent to a synchronized redefinePosition() command.
  - Accumulated output without reset (OUTPUT_PATH_LENGTH_ADDITIVE)
    The path distance accumulated via the command limit is output.
3.12.3 Dynamics of path-synchronous motion

The path object does not keep its own dynamic response parameters for path-synchronous motion.

The following applies when calculating the path velocity profile for simultaneous traversing of a path-synchronous motion:

- Calculation of the path velocity profile without dynamic adaptation:
  - The velocity profile for the path is determined from the dynamic response parameters of the path, see Path dynamics (Page 38).
  - The setpoints of the path interpolator for the path-synchronous motion are limited to the maximum dynamic values on the positioning axis.
  - The dynamic values (velocity, acceleration, and jerk) are adapted to the ratio of the path axis distance to the path-synchronous motion distance.

Use of this formula assumes that the unit settings for the path object and the participating axes are the same.

- Calculation of the path velocity profile with dynamic adaptation:
  The dynamics of the path-synchronous motion are incorporated into the path plan the same as an additional orthogonal coordinate, and, if necessary, the path velocity profile is adapted in such a way that the dynamic limits of the positioning axis are not violated by the path velocity profile.

3.12.4 Path blending with a path-synchronous motion

- Path blending with dynamic adaptation
  The dynamics of the path-synchronous motion are incorporated into the motion plan the same as an additional orthogonal coordinate, and, if necessary, the velocity profile in the blending range is adapted accordingly.

- Path blending without dynamic adaptation
  If the quotient of the distance length (path motion) / distance length (path-synchronous motion) is not equal over the individual path segments, the path segment transitions will be discontinuous with regard to the velocity setpoints of the path object for the path-synchronous motion.

  The setpoints resulting from the path interpolation for the path-synchronous motion are limited on the positioning axis using the axis-specific dynamic limits of that axis.

  For example, if the path object is limited over the path using just the dynamic limits available on the path object, this can result in a setpoint error on the positioning axis relative to the calculated setpoint on the path object for the path-synchronous motion. See Display and monitoring options on the axis (Page 50).
3.12.5 Output of the path distance to the positioning axis

Alternatively, the traveled path distance, i.e. the current path length, can be output to the positioning axis. This distance can be relative to an individual path segment or added up over multiple path segments.

The setting is made in the path command.

For example, this can be used to output path distance-related output cams or measuring inputs.

3.12.6 Output of Cartesian coordinates using the MotionOut Interface

The motionOut.x/y/z interfaces can be used to interconnect the Cartesian coordinates directly with other technology objects, e.g. with the MotionIn interfaces of positioning axes.

For example, this functionality can be used in the application to implement output cams and measuring inputs on Cartesian axes.
3.13 Kinematic adaptation

3.13.1 Kinematic adaptation

The kinematic transformation or the kinematic adaptation is used to convert path axis values to the Cartesian axes, and vice versa.

3.13.2 Kinematic adaptation – fundamentals

3.13.2.1 Scope of the transformation functionality

During forward calculation of the kinematics (including direct kinematics, forward kinematics or forward transformation) for position and motion conversion, the position of the end point of the kinematics is determined in the basic coordinate system from the position of the articulation angle and its spatial arrangement.

During backward calculation (including backward transformation or inverse kinematics), the position of the individual articulation angle is determined from the position of the end point of the kinematics in the basic coordinate system. For path interpolation, the position of the end point of the kinematics in the basic coordinate system is calculated over time.

The position and the dynamic values are transformed.

If the axis settings are outside the transformation range, the values in the BCS and OCS will be set to zero.

The current modulo range is retained in path axes specified as modulo axes.

See Modulo properties (Page 22).

3.13.2.2 Reference points

The following reference points are used in path interpolation:

- Cartesian zero point
- Kinematic zero point
- Kinematic end point

(because a tool is not taken into account, this is equal to the path point)

![Reference points of the coordinate systems in path interpolation](image)

Figure 3-32   Reference points of the coordinate systems in path interpolation
The path object calculates the position on the path. This is also the kinematic end point.

### 3.13.2.3 System variables for path interpolation and transformation on the path object

**Figure 3-33 Overview of system variables of the path object**

The position values and dynamic values can be accessed via a system variable:

### Path data

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path.acceleration</td>
<td>Path acceleration</td>
</tr>
<tr>
<td>path.command</td>
<td>Status of a motion command</td>
</tr>
<tr>
<td>path.dynamicAdaption</td>
<td>Display showing whether adaptation of the path dynamics to the dynamic limit values of the path axes is active.</td>
</tr>
<tr>
<td>path.length</td>
<td>Length of the current path</td>
</tr>
<tr>
<td>path.motionState</td>
<td>Motion status of path motion</td>
</tr>
<tr>
<td>path.position</td>
<td>Path position (within the path length)</td>
</tr>
<tr>
<td>path.velocity</td>
<td>Path velocity</td>
</tr>
<tr>
<td>path.csType</td>
<td>Type of coordinate system</td>
</tr>
<tr>
<td>path.csNumber</td>
<td>Number of the coordinate system (in OCS)</td>
</tr>
</tbody>
</table>

**Cartesian specifications in the basic coordinate system / path-synchronous motion**

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcs.x/y/z/w.position</td>
<td>Set positions</td>
</tr>
<tr>
<td>bcs.x/y/z/w.velocity</td>
<td>Set velocities</td>
</tr>
<tr>
<td>bcs.x/y/z/w.acceleration</td>
<td>Set accelerations</td>
</tr>
<tr>
<td>bcs.linkConstellation</td>
<td>Set link constellation</td>
</tr>
</tbody>
</table>
Basics of Path Interpolation

3.13 Kinematic adaptation

Cartesian actual values

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bcs.x/y/zActual.position</td>
<td>Actual value of the Cartesian positions of the path axes</td>
</tr>
<tr>
<td>bcs.linkConstellationActual</td>
<td>Current articulated joint positioning space</td>
</tr>
</tbody>
</table>

Defaults on path axes from path motion

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mcs.a1/a2/a3.acceleration</td>
<td>Accelerations of the path axes</td>
</tr>
<tr>
<td>mcs.a1/a2/a3.position</td>
<td>Positions of path axes in the axis coordinates</td>
</tr>
<tr>
<td>mcs.a1/a2/a3.velocity</td>
<td>Velocities of the path axes</td>
</tr>
</tbody>
</table>

Object coordinate system

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ocs[1..3].trackingIn</td>
<td>Interface for motion sequence reference value with which the OCS is to be coupled (e.g. TO external encoder)</td>
</tr>
<tr>
<td>ocs[1..3].trackingInPosition</td>
<td>Current value of the motion sequence</td>
</tr>
<tr>
<td>ocs[1..3].trackingPosition</td>
<td>Position of the OCS relative to the reference position</td>
</tr>
<tr>
<td>ocs[1..3].trackingState</td>
<td>Synchronization status</td>
</tr>
<tr>
<td>ocs[1..3].x/y/z.acceleration</td>
<td>Acceleration</td>
</tr>
<tr>
<td>ocs[1..3].x/y/z.position</td>
<td>Item</td>
</tr>
<tr>
<td>ocs[1..3].x/y/z.velocity</td>
<td>Velocity</td>
</tr>
</tbody>
</table>

Override

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>override.acceleration</td>
<td>Acceleration override</td>
</tr>
<tr>
<td>override.velocity</td>
<td>Velocity override</td>
</tr>
</tbody>
</table>

Path command statuses

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linearPathCommand.state</td>
<td>Status of linear interpolation</td>
</tr>
<tr>
<td>circularPathCommand.state</td>
<td>Status of circular interpolation</td>
</tr>
<tr>
<td>polynomialPathCommand.state</td>
<td>Status of polynomial interpolation</td>
</tr>
</tbody>
</table>
Velocity profile

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>specificVelocityProfile.state</td>
<td>Information as to whether a velocity profile is in use</td>
</tr>
<tr>
<td>specificVelocityProfile.value</td>
<td>Profile value</td>
</tr>
<tr>
<td>specificVelocityProfile.activeProfile</td>
<td>Active profile reference</td>
</tr>
<tr>
<td>specificVelocityProfile.processingState</td>
<td>Status of the profile processing</td>
</tr>
</tbody>
</table>

Command queue

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>motionBuffer.numberOfExistentEntries</td>
<td>Number of commands in the command buffer</td>
</tr>
<tr>
<td>motionBuffer.state</td>
<td>Status of the command buffer</td>
</tr>
</tbody>
</table>

Interconnections on the path object

<table>
<thead>
<tr>
<th>System variables</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection.a1</td>
<td>1. Path axis on the path object</td>
</tr>
<tr>
<td>connection.a2</td>
<td>2. Path axis on the path object</td>
</tr>
<tr>
<td>connection.a3</td>
<td>3. Path axis on the path object</td>
</tr>
<tr>
<td>connection.w</td>
<td>Path-synchronous axis on the path object</td>
</tr>
</tbody>
</table>

3.13.2.4 Transformation of the dynamic values

The `kinematicsData.transformationOfDynamics` system variable indicates whether a kinematic transformation supports the dynamics transformation functionality.

3.13.2.5 Differentiation of link constellations

If Cartesian kinematics end points can be reached via various articulation positions, articulation positioning spaces are defined for the corresponding kinematics.

All path motions take place in the same link constellation. For this reason, a change to another link constellation is not possible when a path is being executed. A change to another link constellation is possible through individual axis motions but not via a motion on the path object.

The current transformation-specific link constellation is indicated on the setpoint side in the `bcs.linkConstellation` variable and on the actual value side in the `bcs.linkConstellationActual` variable.

The link constellation is defined specifically for each transformation. See Supported kinematics (Page 62).
3.13.2.6 Information commands for the kinematic transformation

In addition to the implicit conversion in the system, the transformation calculations can also be accessed directly via user commands.

- The \_getPathCartesianPosition() command is used to calculate the Cartesian positions for the axis positions specified in the command.
- The \_getPathAxesPosition() command is used to calculate the axis positions from the Cartesian positions.
- The \_getPathCartesianData() command is used to calculate the Cartesian data for the position, velocity, and acceleration from the axis positions, axis velocities, and axis accelerations specified in the command.
- The \_getPathAxesData() command is used to calculate the axis positions, axis velocities, and axis accelerations from the Cartesian data for the position, velocity, and acceleration specified in the command.

For the calculation of axis positions, the values are specified in the axis coordinate of the path axis, and not relative to the kinematic zero point of the axis.

The modulo range is taken into account.

For the transformation of Cartesian values to path axis values, a link constellation and not a reference position of the axes has to be specified in order to ensure uniqueness.

3.13.2.7 Axis-specific zero point offset in the transformation

It is possible to set an axis-specific offset of the zero position of the axis in the axis-specific coordinate system as well as the zero definition of the axis in the transformation.

The positive direction of the axis and of the axis in the transformation must be the same. These settings are made for the axis.

The offset of the kinematic zero point relative to the axis zero point is specified in the positive direction of the axis.

When modulo axes are used for rotary links with a limited domain in kinematics, such as SCARA, the axis-specific zero point offset and the modulo property of the relevant path axis are defined such that the permissible modulo range of the path axis coincides with the domain of the relevant arm within the kinematics. Otherwise, this can cause an additional limitation in the traversing range of the kinematics.
Example: If a link is limited to \([-180°; 180°)\) and a modulo range of 0° to 360° is defined on the path axis, the zero point offset to -180° should be specified.

### 3.13.2.8 Offset of the kinematic zero point relative to the Cartesian zero point

An offset of the kinematic zero point of the transformation relative to the Cartesian zero point can be set in the `basicOffset` configuration data.

![Figure 3-35 Example of kinematic offset](image)

The above example produces negative values for the kinematic offsets.

Offset in example:
- x: -100
- y: -100
- z: -200

With SIMOTION V4.2 and higher, not only can the BCS be offset but also rotated, allowing for any rotation of the coordinate system from the kinematics zero point. This allows flexible assignment of the BCS to the handling equipment's kinematics.

![Figure 3-36 Coordinate system offset and rotation](image)
The rotations are undertaken after the offset in the following order:
1. Roll around x axis
2. Pitch around (already rotated) y axis
3. Yaw around (already twice rotated) z axis

![Diagram of kinematics offset with rotation]

Offset and rotation (around the y axis) in the example:
- x: -100
- y: -100
- z: -200
- roll: 0°
- pitch: +15°
- jaw: 0°

### 3.13.3 Supported kinematics

#### 3.13.3.1 Supported kinematics and their assignment

The following kinematics can be set using the `typeOfKinematics` configuration data:
- Cartesian 2D/3D gantries (Page 66) (CARTESIAN)
- 2D roller picker (Page 67) (ROLL_PICKER)
- 3D roller picker (Page 71) (ROLL_PICKER_3D) available as of V4.4.
- 2D delta picker (Page 73) (DELTA_2D_PICKER)
- 3D delta picker (Page 77) (DELTA_3D_PICKER)
- SCARA kinematics (Page 80) (SCARA)
- 2D articulated arm kinematics (Page 84) (ARTICULATED_ARM_2D) available as of V4.2.
- 3D articulated arm kinematics (Page 85) (ARTICULATED_ARM)
- 2D swivel arm kinematics (Page 88) (SWIVEL_ARM) available as of V4.2.
3.13.3.2 Configuration screens

With SIMOTION version 4.2 and higher you can use parameterization screens to configure the kinematics. You can access the screens via the configuration menu.

![Configuration screen](image)

Figure 3-38 Open configuration

Depending on the kinematics, the screen will contain several tabs where you can enter mechanical data and offsets and rotations.
3.13 Kinematic adaptation

Example: Cartesian kinematics 2D

Figure 3-39  Cartesian kinematics 2D - configuration
Figure 3-40  Cartesian kinematics 2D - offset
3.13.3 Cartesian 2D/3D gantries

Figure 3-41 Kinematics example: 2D/3D gantry

Table 3-1 Configuration data for the Cartesian kinematics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typeOfKinematics:</td>
<td>Cartesian gantry kinematic type</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the zero point of Cartesian coordinate x relative to the zero point of axis coordinate A1</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the zero point of Cartesian coordinate y relative to the zero point of axis coordinate A2</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the zero point of Cartesian coordinate z relative to the zero point of axis coordinate A3</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>cartesianKinematicsType</td>
<td>Select 2D or 3D (determines the number of axes involved)</td>
</tr>
<tr>
<td>config2D</td>
<td>Main plane (only for 2D gantry)</td>
</tr>
</tbody>
</table>

Table 3-2 Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linkConstellation</td>
<td>Irrelevant (always 1)</td>
</tr>
</tbody>
</table>
3.13.3.4 2D roller picker

General

The 2D roller picker has two-dimensional kinematics. You can configure 2D roller pickers in all three main planes. This description is based on a configuration in the x-y plane.
Basics of Path Interpolation

3.13 Kinematic adaptation

Figure 3-43  Kinematics of the 2D roller picker (guide pulley on the opposite side of the tool)

The guide pulley must be located on the opposite side of the tool.
3.13 Kinematic adaptation

Figure 3-44  Kinematics of the 2D roller picker (guide pulley on the tool; this case is not considered)
The alternative variant with the guide pulley on the tool can be derived by converting the coordinates:

<table>
<thead>
<tr>
<th>Guide pulley on the tool</th>
<th>Guide pulley on the opposite side of the tool</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>-x</td>
</tr>
<tr>
<td>y</td>
<td>-y</td>
</tr>
<tr>
<td>R1</td>
<td>R2</td>
</tr>
<tr>
<td>R2</td>
<td>R1</td>
</tr>
</tbody>
</table>

**Note**

The two path axes must be configured so that 360 axis-units (i.e. mm, degree, etc.) produce a disk revolution. The "modulo axis" setting should be prevented. See Units (Page 23) and Modulo properties (Page 22).

Table 3-3  Configuration data for the 2D roller picker kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics: ROLL_PICKER</th>
<th>2D roller picker kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
</tbody>
</table>

Axis 3 is not available for the 2D roller picker.

| config2D | Main plane of the path axes |

Table 3-4  Specification of the radius of the disks on the motors in:

<table>
<thead>
<tr>
<th>radius1</th>
<th>Disk radius for path axis 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius2</td>
<td>Disk radius for path axis 2</td>
</tr>
</tbody>
</table>

Table 3-5  Possible articulated joint positioning spaces

| LinkConstellation | Irrelevant (always 1) |
3.13.3.5  3D roller picker

General

Figure 3-45  Kinematics of the 3D roll picker for a roll picker in the z-x plane
The 3D roller picker is a 2D roller picker that can be moved via an additional linear axis in a further Cartesian coordinate direction. You can set the 3D roller picker in all three main planes as with the 2D roller pickers. The third path axis then extends in the third Cartesian direction.

In the example, the roll picker is set in the z-x plane.

A1 and A2 are the two axes of the roll picker in the example figure. The linear axis A3 is indicated by the arrow in the y direction.

Further information can be found in the documentation of the 2D roller picker (Page 67).

Table 3-6  Configuration data for the 3D roller picker kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics: ROLL_PICKER_3D</th>
<th>3D roller picker kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>config2D</td>
<td>Main plane of the path axes</td>
</tr>
</tbody>
</table>

Specification of the radius of the disks on the motors in:

| radius1                          | Disk radius for path axis 1     |
| radius2                          | Disk radius for path axis 2     |

Table 3-7  Possible articulated joint positioning spaces

| LinkConstellation                | Irrelevant (always 1)            |
3.13.3.6 2D delta picker

General

Figure 3-46  Kinematics of the 2D delta picker (x-y plane example)
Definitions

- The complete structure is contained in one of the two-dimensional main planes. The x-y plane is used as an example in the following description.

- A1 and A2 denote the two active drive axes of the kinematic structure. They lie on the straight line \(y = 0\) and are separated from each other by the distance \(2x\ distanceD1\). Their zero position within the kinematic structure corresponds to the orientation of the upper arm segments (\(length1\)) in the direction of the negative y axis. Positive displacements occur as shown in the figure.

- It is assumed that the lower connection plate between G3 and G4 is always oriented horizontally. This results in \(y_{G3} = y_{G4}\) and a horizontal distance of \(2x\ distanceD2\).

- If \(x_{ZP} = y_{ZP} = 0\), the zero position of the kinematics is located midway between the drive axes A1 and A2.

- The target point of the direct transformation, with its coordinates \(x_{EP}\) and \(y_{EP}\) is defined as being midway between G3 and G4. This results in the position \(G3 = (x_{EP} - distanceD2; y_{EP})\) and \(G4 = (x_{EP} + distanceD2; y_{EP})\).

Table 3-8 Configuration data for the 2D delta picker kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics: DELTA_2D_PICKER</th>
<th>2D delta picker kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset</td>
<td>Offset of the kinematic zero point (ZP) relative to the Cartesian zero point</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
</tbody>
</table>

Axis 3 is not available for the 2D delta picker.

<table>
<thead>
<tr>
<th>config2D</th>
<th>Main plane of the path axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>length1</td>
<td>Length of the upper arm segment</td>
</tr>
<tr>
<td>length2</td>
<td>Length of the lower arm segment</td>
</tr>
<tr>
<td>distanceD1</td>
<td>Distance of the drive axes A1 and A2 from the kinematic zero point (ZP)</td>
</tr>
<tr>
<td>distanceD2</td>
<td>Distance of the joints G3 and G4 from the end point (EP)</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Offset of the A1 drive axis</td>
</tr>
<tr>
<td>offsetA2</td>
<td>Offset of the A2 drive axis</td>
</tr>
</tbody>
</table>

Table 3-9 Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>LinkConstellation</th>
<th>Angle of axis A1 in the range [-180°, 0°]. Angle of axis A2 in the range [0°, 180°].</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Angle of axis A1 in the range [-180°, 0°]. Angle of axis A2 in the range [0°, 180°].</td>
</tr>
<tr>
<td>2</td>
<td>Angle of axis A1 in the range [0°, 180°]. Angle of axis A2 in the range [0°, 180°].</td>
</tr>
</tbody>
</table>
### 3.13 Kinematic adaptation

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Angle of axis A1 in the range (0°, 180°). Angle of axis A2 in the range (-180°, 0°).</td>
</tr>
<tr>
<td>4</td>
<td>Angle of axis A1 in the range [-180°, 0°]. Angle of axis A2 in the range (-180°, 0°).</td>
</tr>
</tbody>
</table>
Figure 3-47  Illustration of the articulated joint positioning as exemplified by a single axis
3.13.3.7 3D delta picker

General

Figure 3-48 Kinematics of the 3D delta picker (top view)

Figure 3-49 Kinematics of the 3D delta picker (bottom view)
Definitions

- A1, A2 and A3 denote the three active drive axes of the kinematic structure. They are in the x-y plane where \( z = 0 \), and each is at distance \( \text{distanceD1} \) from the kinematic zero point (ZP). Their zero position within the kinematic structure corresponds to direct orientation of the upper arm segments (\text{length1}) in the direction of the negative Z axis. Positive displacements occur counterclockwise, as shown in the previous figure.

- G1 to G6 denote the freely movable joints.

- It is assumed that the connection of the joints at the end point (EP) has a horizontal orientation based on the parallel struts. This results in \( y_{G4} = y_{G5} = y_{G6} \). G4 to G6 are all at the horizontal distance \( \text{distanceD2} \) from the end point (EP).

- If \( x_{ZP} = y_{ZP} = z_{ZP} = 0 \), the zero position of the kinematics is located midway between the drive axes A1 to A3.

- The target point of the transformation, with its coordinates \( x_{\text{EP}}, y_{\text{EP}} \) and \( z_{\text{EP}} \) is defined as being midway between G4 and G6.

Table 3-10  Configuration data for the 3D delta picker kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics: DELTA_3D_PICKER</th>
<th>3D delta picker kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset</td>
<td>Offset of the kinematic zero point (ZP) relative to the Cartesian zero point</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
</tbody>
</table>
### Table 3-11 Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>LinkConstellation</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>length1</td>
<td>Length of the upper arm segment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>length2</td>
<td>Length of the lower arm segment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>distanceD1</td>
<td>Distance of the drive axes A1 to A3 from the kinematic zero point (ZP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>distanceD2</td>
<td>Distance of the articulations G4 to G6 from the end point (EP)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>offsetA1</td>
<td>Offset of the A1 drive axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>offsetA2</td>
<td>Offset of the A2 drive axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>offsetA3</td>
<td>Offset of the A3 drive axis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>angleArm1ToX</td>
<td>Angular offset of arm A1-G1-G4 with respect to the x axis for rotation around the positive z axis.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>angleArm2ToArm1</td>
<td>Angular offset of arm A2-G2-G5 with respect to arm A1-G1-G4 for rotation around the positive z axis.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>angleArm3ToArm1</td>
<td>Angular offset of arm A3-G3-G6 with respect to arm A1-G1-G4 for rotation around the positive z axis.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following angles are always in relation to the direct orientation $\varphi_0$ of the active axis to the end point.

You will find an illustration of the articulated joint positioning options for a single axis in 2D delta picker (Page 73).
3.13.3.8 SCARA kinematics

General

The kinematic zero point is at point A1.

The zero positions of the A1 axis and A2 axis are as follows:
The definition range of the single A1 and A2 axes is limited to [-180°; 180°).
Link compensations

Coupled axes

Mechanical couplings are possible:
- between A1 and A2
- between A1, A2, and A_{synchronous} (A4)
- between A_{synchronous} (A4) and A3

The following axis couplings can be compensated via the system:
- A coupling from axis A1 to axis A2
- A coupling from axis A1 and axis A2 to the path-synchronous controlled axis A4
  That is, the setpoint of axis A4 is changed to the position axis in accordance with the changes of A1 and A2.
  If axis A1 and/or axis A2 is traversed and a path-synchronous motion in w is specified in parallel, the system superimposes/adds a path-synchronous motion specification and compensation onto the position axis.
- A coupling from axis A4 to axis A3 (lifting axis)
  If axis A3 is traversed via the path motion and a compensation from w to axis A3 is required simultaneously, the specifications are superimposed.

The compensation functionality and the specifications to the path-synchronous axis w via the path-synchronous motion are independent of one another and are executed simultaneously by the system.

Effective direction of the coupled axes

For the coupling of axis A1 to axis A2 and of axis A4 to axis A3, the following applies:
- For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a negative motion on the second axis.
- For the coupling of axis A1 to axis A4 and of axis A2 to axis A4, the following applies:
  For a coupling factor of > 0.0, the transformation is based on the assumption that a positive motion on the first axes results in a positive motion on the second axis.
  The coupling between axis A4 and axis A3 is implemented as a spindle pitch, i.e. for a coupling factor of +1.0, 360.0 degrees on axis A4 correspond to a path of -1.0 mm on axis A3.
### Table 3-12 Configuration data for the SCARA kinematics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typeOfKinematics: SCARA</td>
<td>SCARA kinematics type</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Offset of the axis zero point of axis A1 relative to zero position of axis A1 in the transformation</td>
</tr>
<tr>
<td>distanceA1A2</td>
<td>Distance A1 - A2</td>
</tr>
<tr>
<td>offsetA2</td>
<td>Offset of the axis zero point of axis A2 relative to zero position of axis A2 in the transformation</td>
</tr>
<tr>
<td>distanceA2Endpoint</td>
<td>A2 - end point distance</td>
</tr>
<tr>
<td>linkCompensationA2.enableA1A2</td>
<td>Compensate articulated joint dependency of A1 to A2</td>
</tr>
<tr>
<td>linkCompensationA2.factorA1A2</td>
<td>Coupling factor for the compensation on axis A2</td>
</tr>
<tr>
<td>linkCompensationA4.enableA1A4</td>
<td>Compensate articulated joint dependency of A1 to A4</td>
</tr>
<tr>
<td>linkCompensationA4.factorA1A4</td>
<td>Coupling factor for the compensation on axis A4</td>
</tr>
<tr>
<td>linkCompensationA4.enableA2A4</td>
<td>Compensate articulated joint dependency of A2 to A4</td>
</tr>
<tr>
<td>linkCompensationA4.factorA2A4</td>
<td>Coupling factor for the compensation on axis A4</td>
</tr>
<tr>
<td>linkCompensationA3.enableA4A3</td>
<td>Compensate articulated joint dependency of A4 to A3</td>
</tr>
<tr>
<td>linkCompensationA3.factorA4A3</td>
<td>Coupling factor for the compensation on axis A3</td>
</tr>
</tbody>
</table>

### Table 3-13 Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>LinkConstellation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Positive articulated joint positioning: Angle of axis A2 in the range [0°, 180°) relative to the kinematic zero point</td>
</tr>
<tr>
<td>2</td>
<td>Negative articulated joint positioning: Angle of axis A2 in the range [-180°, 0°) relative to the kinematic zero point</td>
</tr>
</tbody>
</table>
Basics of Path Interpolation

3.13 Kinematic adaptation

3.13.3.9 2D articulated arm kinematics

General

![Diagram of 2D articulated arm kinematics]

Figure 3-56 2D articulated arm: Display of the axes

Table 3-14 Configuration data for the 2D articulated arm kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics:ARTICULATED_ARM_2D</th>
<th>2D articulated arm kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>config2D</td>
<td>Main plane of the path axes</td>
</tr>
<tr>
<td>linkCompensationA2.enableA1A2</td>
<td>Compensate articulated joint dependency of A1 to A2</td>
</tr>
<tr>
<td>distanceA1A2</td>
<td>Distance A1 - A2</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Angle offset</td>
</tr>
<tr>
<td>offsetA2</td>
<td>Angle offset</td>
</tr>
</tbody>
</table>

Table 3-15  Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>LinkConstellation</th>
<th>1</th>
<th>Angle of axis A2 in the range [0°, 180°) relative to the kinematic zero point</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>Angle of axis A2 in the range [-180°, 0°) relative to the kinematic zero point</td>
</tr>
</tbody>
</table>

3.13.3.10 3D articulated arm kinematics

General

Figure 3-57  3D articulated arm: Display of the axes
The kinematic zero point is at point A1. The point is the zero position of the kinematics if distanceA1A2, distanceA2A3, and distanceA3EP are pointing in the Cartesian x-direction.

Figure 3-58 3D articulated arm: Kinematics

Figure 3-59 3D articulated arm: Zero positions of axes A2 and A3
The definition range of the single A1 to A3 axes is limited to [-180°; 180°).

**Coupled axes**

If a positive coupling factor between two axes is specified, the transformation assumes that a positive motion on the first axes leads to a negative motion on the second axis.

### Table 3-16 Configuration data for the 3D articulated arm kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics:</th>
<th>3D articulated arm kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARTICULATED_ARM</td>
<td>3D articulated arm kinematics</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Offset of axis zero point axis A1 relative to zero position of axis A1 in the transformation</td>
</tr>
<tr>
<td>distanceA1A2</td>
<td>Distance A1 - A2</td>
</tr>
<tr>
<td>offsetA2</td>
<td>Offset of axis zero point axis A2 relative to zero position of axis A2 in the transformation</td>
</tr>
<tr>
<td>distanceA2A3</td>
<td>A2 - A3 distance</td>
</tr>
<tr>
<td>offsetA3</td>
<td>Offset of axis zero point axis A3 relative to zero position of axis A3 in the transformation</td>
</tr>
<tr>
<td>distanceA3Endpoint</td>
<td>A3 - end point distance</td>
</tr>
<tr>
<td>linkCompensation.enableA2A3</td>
<td>Compensate articulated joint dependency of A3 to A2</td>
</tr>
<tr>
<td>linkCompensation.factorA2A3</td>
<td>Coupling factor for the compensation on axis A2</td>
</tr>
</tbody>
</table>
Table 3-17 Possible articulated joint positioning spaces

<table>
<thead>
<tr>
<th>LinkConstellation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Angle of axis A3 in the range [0°, 180°) relative to the kinematic zero point. Angle of axis A1 corresponds to ( \text{atan}(EP_y/EP_x) )</td>
</tr>
<tr>
<td>2</td>
<td>Angle of axis A3 in the range [-180°, 0°) relative to the kinematic zero point. Angle of axis A1 corresponds to ( \text{atan}(EP_y/EP_x) )</td>
</tr>
<tr>
<td>3</td>
<td>Angle of axis A3 in the range [0°, 180°) relative to the kinematic zero point. Angle of axis A1 corresponds to ( -\text{atan}(EP_y/EP_x) )</td>
</tr>
<tr>
<td>4</td>
<td>Angle of axis A3 in the range [-180°, 0°) relative to the kinematic zero point. Angle of axis A1 corresponds to ( -\text{atan}(EP_y/EP_x) )</td>
</tr>
</tbody>
</table>

3.13.3.11 2D swivel arm kinematics

2D swivel arm kinematics

In swivel arm kinetics, programming settings are made on the lateral surface that can be accessed by the kinematics.
Unraveling the lateral surface results in a 2D plane, for which coordinate-plane and offset parameters can be assigned in the same way as with Cartesian 2D (Page 66) kinematics. The offsets are applied to the set coordinate plane and rotation is about the axis that is perpendicular to the plane.

Depending on the plane used, the following parameters are effective:

**X_Y plane:**
Offset in x and y directions and rotation about the z axis

**Y_Z plane:**
Offset in y and z directions and rotation about the x axis

**Z_X plane:**
Offset in z and x directions and rotation about the y axis

With this type of kinematics, rotation does not serve any useful purpose.
The link compensations LinkCompensationA1 and LinkCompensationA2 and angular offset offsetA1 at rotary joint A1 work in the same way as with SCARA kinematics (Page 80).

With this type of kinematics, the conveyor tracking function (see Motion sequence at path object (Page 100)) does not serve any useful purpose.

Table 3-18  Configuration data for the 2D swivel arm kinematics

<table>
<thead>
<tr>
<th>parameter</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>typeOfKinematics:SWIVEL ARM</td>
<td>2D swivel arm kinematics type</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>config2D</td>
<td>Main plane of the path axes</td>
</tr>
<tr>
<td>linkCompensationA4.enableA1A4</td>
<td>Compensate articulated joint dependency of A1 to A4</td>
</tr>
<tr>
<td>linkCompensationA4.factorA1A4</td>
<td>Coupling factor for the compensation on axis A4</td>
</tr>
<tr>
<td>linkCompensationA2.enableA4A2</td>
<td>Compensate A4 articulated joint positioning dependence to A2</td>
</tr>
<tr>
<td>linkCompensationA2.factorA4A2</td>
<td>Coupling factor for the compensation on axis A2</td>
</tr>
<tr>
<td>distanceA1Endpoint</td>
<td>A1 - end point distance</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Angle offset at rotary joint A1</td>
</tr>
</tbody>
</table>
3.13.3.12 3D cylindrical robot

General

In the 3D cylindrical robot, one of the Cartesian axes is replaced by a rotary axis. The working area is cylindrical.
Definitions

- A1, A2, A3 and A4 designate the four active drive axes of the kinematic structure. A1 and A4 are rotary axes and A2 and A3 linear axes.

- L1 designates the distance between axes A2 and A3. In the zero position, this corresponds to the distance between axis A3 to the x axis. This distance therefore also has an effect on the calculation of the end point of the transformation. Positive run in the direction of the y axis. In the above plan view, the value of L1 is therefore negative.

- The definition range of axis A1 is limited to $[-180^\circ; 180^\circ)$.

**Coupled axes**

Mechanical couplings are possible:

- between A1 and A4
- between A4 and A2

The following axis couplings can be compensated via the system:

- a coupling from axis A1 to axis A4
- a coupling from axis A4 to axis A2

**Effective direction of the coupled axes**

For the coupling of axis A1 to axis A4, the following applies:

For a coupling factor of $> 0.0$, the transformation is based on the assumption that a positive motion on the first axes results in a positive motion on the second axis.

For the coupling of axis A4 to axis A2, the following applies:

For a coupling factor of $> 0.0$, the transformation is based on the assumption that a positive motion on the first axes results in a negative motion on the second axis.

The coupling between axis A4 and axis A2 is implemented as a spindle pitch, i.e. for a coupling factor of $+1.0$, 360.0 degrees on axis A4 correspond to a path of -1.0 mm on axis A2.

Table 3-19  Configuration data for the 3D cylindrical robot kinematics

<table>
<thead>
<tr>
<th>typeOfKinematics: CYLINDRICAL</th>
<th>3D cylindrical robot kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>basicOffset</td>
<td>Offset of the kinematic zero point (ZP) relative to the Cartesian zero point</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>distanceA2A3</td>
<td>Distance of axis A2 to axis A3</td>
</tr>
<tr>
<td>offsetA1</td>
<td>Angular offset of rotary axis A1</td>
</tr>
</tbody>
</table>
3.13 Kinematic adaptation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>linkCompensation.enableA1A4</td>
<td></td>
<td>Compensate articulated joint dependency of axis A1 to axis A4</td>
</tr>
<tr>
<td>linkCompensation.factorA1A4</td>
<td></td>
<td>Coupling factor for the compensation on axis A4</td>
</tr>
<tr>
<td>linkCompensation.enableA4A2</td>
<td></td>
<td>Compensate articulated joint dependency of axis A4 to axis A2</td>
</tr>
<tr>
<td>linkCompensation.factorA4A2</td>
<td></td>
<td>Coupling factor for the compensation on axis A2</td>
</tr>
</tbody>
</table>

3.13.3.13 2D/3D user function

General

The kinematics 2D/3D user function allows the use of a transformation function provided by the user. This user function is created in ST notation and executed via a hook interface in the TO execution context. Compiled ST code in the TO execution context can be executed via the hook interface provided on the TO.

The function interface for the 2D/3D user function is permanently defined in ST. The input parameters are written by the TO and the calculated output values by the ST function.

Hook interface

Two different interfaces are defined for the forward and backward transformations. Each of the two transformation directions is therefore implemented by a function.

Table 3-20 FC input parameters for the forward transformation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a1Position</td>
<td>LREAL</td>
<td>Position, axis 1</td>
</tr>
<tr>
<td>a1Velocity</td>
<td>LREAL</td>
<td>Velocity, axis 1</td>
</tr>
<tr>
<td>a1Acceleration</td>
<td>LREAL</td>
<td>Acceleration, axis 1</td>
</tr>
<tr>
<td>a2Position</td>
<td>LREAL</td>
<td>Position, axis 2</td>
</tr>
<tr>
<td>a2Velocity</td>
<td>LREAL</td>
<td>Velocity, axis 2</td>
</tr>
<tr>
<td>a2Acceleration</td>
<td>LREAL</td>
<td>Acceleration, axis 2</td>
</tr>
<tr>
<td>a3Position</td>
<td>LREAL</td>
<td>Position, axis 3</td>
</tr>
<tr>
<td>a3Velocity</td>
<td>LREAL</td>
<td>Velocity, axis 3</td>
</tr>
<tr>
<td>a3Acceleration</td>
<td>LREAL</td>
<td>Acceleration, axis 3</td>
</tr>
<tr>
<td>a4Position</td>
<td>LREAL</td>
<td>Position, axis 4</td>
</tr>
<tr>
<td>a4Velocity</td>
<td>LREAL</td>
<td>Velocity, axis 4</td>
</tr>
<tr>
<td>a4Acceleration</td>
<td>LREAL</td>
<td>Acceleration, axis 4</td>
</tr>
<tr>
<td>pathObject</td>
<td>PathObjectType</td>
<td>Instance of the path object</td>
</tr>
<tr>
<td>executionContext</td>
<td>EnumPathObjectTransformationContext</td>
<td>Execution context &lt;br&gt; <strong>TO_INTERPOLATION_CYCLE</strong>: FC call for calculation of the transformation in interpolation cycles, e.g. for path interpolation &lt;br&gt; <strong>NON_CYCLIC</strong>: FC call for non-cyclic calculation of the transformation, e.g. for a query function</td>
</tr>
</tbody>
</table>
Basics of Path Interpolation

3.13 Kinematic adaptation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinematicsConfigData</td>
<td>Array [1..32]</td>
<td>User-defined configuration data (corresponds to the configuration data item <code>userTrafoID</code>)</td>
</tr>
<tr>
<td>userTrafoID</td>
<td>UDINT</td>
<td>Identifier of the user-defined coordinate transformation (corresponds to the configuration data item <code>userTrafoID</code>)</td>
</tr>
<tr>
<td>kinematicsType</td>
<td>EnumPathUserKinematicsType</td>
<td>Setting of the kinematics for the user function (_2D or _3D)</td>
</tr>
<tr>
<td>config2D</td>
<td>EnumPathKinematicsConfig2D</td>
<td>Main plane of the path axes (is ignored for the 3D user function). Possible values are X, Y, Z and Z, X</td>
</tr>
</tbody>
</table>

Table 3-21  FC output parameters for the forward transformation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>functionResult</td>
<td>DINT</td>
<td>Function result</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: OK, velocity and acceleration have not been calculated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: OK, velocity and acceleration have been calculated</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Other values: Error</td>
</tr>
<tr>
<td>xPosition</td>
<td>LREAL</td>
<td>X component of the position</td>
</tr>
<tr>
<td>xVelocity</td>
<td>LREAL</td>
<td>X component of the velocity</td>
</tr>
<tr>
<td>xAcceleration</td>
<td>LREAL</td>
<td>X component of the acceleration</td>
</tr>
<tr>
<td>yPosition</td>
<td>LREAL</td>
<td>Y component of the position</td>
</tr>
<tr>
<td>yVelocity</td>
<td>LREAL</td>
<td>Y component of the velocity</td>
</tr>
<tr>
<td>yAcceleration</td>
<td>LREAL</td>
<td>Y component of the acceleration</td>
</tr>
<tr>
<td>zPosition</td>
<td>LREAL</td>
<td>Z component of the position</td>
</tr>
<tr>
<td>zVelocity</td>
<td>LREAL</td>
<td>Z component of the velocity</td>
</tr>
<tr>
<td>zAcceleration</td>
<td>LREAL</td>
<td>Z component of the acceleration</td>
</tr>
<tr>
<td>wPosition</td>
<td>LREAL</td>
<td>W component of the position</td>
</tr>
<tr>
<td>wVelocity</td>
<td>LREAL</td>
<td>W component of the velocity</td>
</tr>
<tr>
<td>wAcceleration</td>
<td>LREAL</td>
<td>W component of the acceleration</td>
</tr>
<tr>
<td>linkConstellation</td>
<td>DINT</td>
<td>Articulated joint positioning</td>
</tr>
</tbody>
</table>

Table 3-22  FC input parameters for the backward transformation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>xPosition</td>
<td>LREAL</td>
<td>X component of the position</td>
</tr>
<tr>
<td>xVelocity</td>
<td>LREAL</td>
<td>X component of the velocity</td>
</tr>
<tr>
<td>xAcceleration</td>
<td>LREAL</td>
<td>X component of the acceleration</td>
</tr>
<tr>
<td>yPosition</td>
<td>LREAL</td>
<td>Y component of the position</td>
</tr>
<tr>
<td>yVelocity</td>
<td>LREAL</td>
<td>Y component of the velocity</td>
</tr>
<tr>
<td>yAcceleration</td>
<td>LREAL</td>
<td>Y component of the acceleration</td>
</tr>
<tr>
<td>zPosition</td>
<td>LREAL</td>
<td>Z component of the position</td>
</tr>
<tr>
<td>zVelocity</td>
<td>LREAL</td>
<td>Z component of the velocity</td>
</tr>
</tbody>
</table>
### 3.13 Kinematic adaptation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>zAcceleration</td>
<td>LREAL</td>
<td>Z component of the acceleration</td>
</tr>
<tr>
<td>wPosition</td>
<td>LREAL</td>
<td>W component of the position</td>
</tr>
<tr>
<td>wVelocity</td>
<td>LREAL</td>
<td>W component of the velocity</td>
</tr>
<tr>
<td>wAcceleration</td>
<td>LREAL</td>
<td>W component of the acceleration</td>
</tr>
<tr>
<td>w2Position</td>
<td>LREAL</td>
<td>W2 component of the position (path length)</td>
</tr>
<tr>
<td>w2Velocity</td>
<td>LREAL</td>
<td>W2 component of the velocity</td>
</tr>
<tr>
<td>w2Acceleration</td>
<td>LREAL</td>
<td>W2 component of the acceleration</td>
</tr>
<tr>
<td>linkConstellation</td>
<td>DINT</td>
<td>Articulated joint positioning</td>
</tr>
<tr>
<td>pathObject</td>
<td>_PathObjectType</td>
<td>Instance of the path object</td>
</tr>
</tbody>
</table>
| executionContext   | EnumPathObjectTransformationContext | Execution context  
|                    |                 | TO_INTERPOLATION_CYCLE: FC call for calculation of the transformation in interpolation cycles, e.g. for path interpolation  
|                    |                 | NON_CYCLIC: FC call for non-cyclic calculation of the transformation, e.g. for a query function  |
| kinematicsConfigData | Array [1..32]  | User-defined configuration data (corresponds to the configuration data item parameter) |
| userTrafoID        | UDINT           | Identifier of the user-defined coordinate transformation (corresponds to the configuration data item userTrafoId) |
| kinematicsType     | EnumPathUserKinematicsType | Setting of the kinematics for the user function (_2D or _3D) |
| config2D           | EnumPathKinematicsConfig2D | Main plane of the path axes (is ignored for the 3D user function). Possible values are X_Y, Y_Z and Z_X |

Table 3-23 FC output parameters for the backward transformation

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
</table>
| functionResult | DINT         | Function result  
|              |              | 0: OK, velocity and acceleration have not been calculated  
|              |              | 1: OK, velocity and acceleration have been calculated  
|              |              | Other values: Error  |
| a1Position   | LREAL        | Position, axis 1                                                |
| a1Velocity   | LREAL        | Velocity, axis 1                                                |
| a1Acceleration | LREAL     | Acceleration, axis 1                                            |
| a2Position   | LREAL        | Position, axis 2                                                |
| a2Velocity   | LREAL        | Velocity, axis 2                                                |
| a2Acceleration | LREAL     | Acceleration, axis 2                                            |
| a3Position   | LREAL        | Position, axis 3                                                |
| a3Velocity   | LREAL        | Velocity, axis 3                                                |
Name | Type | Description
--- | --- | ---
a3Acceleration | LREAL | Acceleration, axis 3
a4Position | LREAL | Position, axis 4
a4Velocity | LREAL | Velocity, axis 4
a4Acceleration | LREAL | Acceleration, axis 4

**Note**

The functionResult output parameter

The user must set the functionResult output parameter to 1 if the velocities and the accelerations are to be calculated in the FCs in addition to the positions. If the parameter is not set explicitly, then it corresponds to 0. In this case, only the calculated position values of the FC are taken into account.

**Note**

Units for the interface variables

All variables at the interface are available in the set user units.

**Function creation**

The transformation functions can be created in the ST programming language. In order to be able to use the functions for the transformation, a pragma with attribute must be set.

ToHookApplicable := YES | NO

If the pragma is set, only certain operations are permitted.

**Permitted operations**

- Numeric and logic standard functions
- Access to bits in bit strings
- Standard functions for data type conversion
- Converting between any data types and byte arrays
- Combination of bit-string data types
- Functions for verification of floating-point numbers
- Functions for selection
- Access to global variables
- Consistent access to global variables of derived data types (UDT)
- Determination of the memory size of a variable or of a data type

**Operations that are not permitted**

- System functions on the TO
- Access functions to the TO (configuration data, system variables)
- Access functions to the I/O data and the I/O container
- System functions on devices
- Standard functions for controlling I/O modules and drive components
- System functions for controlling axes according to PLCopen
- Data backup and initialization from user program
- Communication functions
- System function blocks (for example counters, edge encoders or timers)
- Marshalling
- String machining
- Conversion of technology object data types

**Example**

Here is a minimal example of the FCs of the forward and backward transformation.

**Forward transformation**

```plaintext
FUNCTION FC_trans_Direct_Real :StructPathDirectUserTransformationOut
{ToHookApplicable:= YES}
VAR_INPUT
  ParIn : StructPathDirectUserTransformationIn;
END_VAR
FC_trans_Direct_Real.xPosition      := ParIn.a1Position;
FC_trans_Direct_Real.yPosition      := ParIn.a2Position;
FC_trans_Direct_Real.zPosition      := ParIn.a3Position;
FC_trans_Direct_Real.wPosition      := ParIn.a4Position;
FC_trans_Direct_Real.functionResult := 0;
END_FUNCTION
```

**Backward transformation**

```plaintext
FUNCTION FC_trans_Inverse_Real :StructPathInverseUserTransformationOut
{ToHookApplicable:= YES}
VAR_INPUT
  ParIn : StructPathInverseUserTransformationIn;
END_VAR
FC_trans_Inverse_Real.a1Position     := ParIn.xPosition;
FC_trans_Inverse_Real.a2Position     := ParIn.yPosition;
FC_trans_Inverse_Real.a3Position     := ParIn.zPosition;
FC_trans_Inverse_Real.a4Position     := ParIn.wPosition;
FC_trans_Inverse_Real.functionResult := 0;
END_FUNCTION
```

**Execution time**

The time for execution of the user transformation function is included in the execution time of the IPO system task. You will find more information on monitoring functions and settings of the IPO system task in the Basic functions Manual in Section Execution System, Tasks, and System Cycle Clocks.
3.13 Kinematic adaptation

Breakpoints

For the FCs of the user transformation function, breakpoints are ignored.

Definitions

Table 3-24 Configuration data for the user-specific coordinate transformation

<table>
<thead>
<tr>
<th>typeOfKinematics: USER_FUNCTION</th>
<th>2D/3D user function kinematics type</th>
</tr>
</thead>
<tbody>
<tr>
<td>kinematicsType</td>
<td>Setting of the kinematics for the user function</td>
</tr>
<tr>
<td></td>
<td>• 2D: 2D programming in a plane</td>
</tr>
<tr>
<td></td>
<td>• 3D: 3D programming in space</td>
</tr>
<tr>
<td>basicOffset</td>
<td>Offset of the kinematic zero point (ZP) relative to the Cartesian zero point</td>
</tr>
<tr>
<td>basicOffset.x</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the x coordinate</td>
</tr>
<tr>
<td>basicOffset.y</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the y coordinate</td>
</tr>
<tr>
<td>basicOffset.z</td>
<td>Offset of the kinematic zero point relative to the Cartesian zero point, the z coordinate</td>
</tr>
<tr>
<td>basicOffset.roll</td>
<td>Rotation about x (roll)</td>
</tr>
<tr>
<td>basicOffset.pitch</td>
<td>Rotation about y (pitch)</td>
</tr>
<tr>
<td>basicOffset.yaw</td>
<td>Rotation about z (yaw)</td>
</tr>
<tr>
<td>config2D</td>
<td>Main plane of the path axes</td>
</tr>
<tr>
<td>parameter</td>
<td>Up to 32 user-defined parameters for the transformation</td>
</tr>
<tr>
<td>userTrafoID</td>
<td>Identifier of the user-defined coordinate transformation</td>
</tr>
</tbody>
</table>

3.13.3.14 Use of virtual axes

If you want to set up kinematics that cannot be mapped onto any existing kinematics but not all axes are used as real axes for these kinematics, you must create and interconnect the missing axes as virtual path axes.

As an example: The articulated arm kinematics provides three axes.
3.13 Kinematic adaptation

3.13.3.15 Specific kinematics

SPECIFIC kinematics type

The SPECIFIC kinematics type can be set using the typeOfKinematics configuration datum.

Example of settings on path object

```plaintext
Kinematics.typeOfKinematics = SPECIFIC (6)
```

The kinematics and parameters required can then be specified by the TrafoID and a parameter list.

See also

Appendix A (Page 177)
### 3.14 Object coordinate systems and motion sequences on the path object

#### 3.14.1 Object coordinate system (OCS) on the path object

As of SIMOTION V4.1.2, position details in the motion commands can be related optionally to the basic coordinate system (BCS, previously present functionality) or to an object coordinate system (OCS).

The motion sequence is available as from V4.2

An OCS can be permanent (static OCS) or coupled with a motion value supplied to the trackingIn interface of the path object.

A technology object that provides motion information with a position (the motion sequence reference value) can use the TrackingIn interface to interconnect with the path object. This can be, for example, an external encoder or a positioning axis.

If path motions relate to an OCS, then, for example, products can be taken from a moving belt or placed there.

---

**Note**

For an active motion sequence, a blending with dynamic response adaptation (blendingMode:= ACTIVE_WITH_DYNAMIC_ADAPTATION) is not supported. Motions programmed with blending will then be performed without blending.

---

Three object coordinate systems are available on the path object and the positions of the working point are displayed in the relevant OCS (see also system variable ocs).

**Activating OCS**

The reference position of the OCS relative to the BCS is set again via the system function _setPathObjectOcs. The default values can be set in the SCOUT form Object coordinate systems. However, these values are only activated after _setPathObjectOcs has been called. The corresponding system variable for the default values is userDefaultOcs.

The default values can also be set by calibration (Page 132). In this case, the new values are first written into userDefaultOcs. The reference position of the object coordinate system is itself not overwritten. For this, it is not necessary to call _setPathObjectOcs.

If the parameter ocsSettingType of the system function _setPathObjectOcs is set to the value DIRECT, values for offset and rotation can be entered directly.

**Coupled OCS**

A coupled OCS is an OCS coupled with a motion value of a technology object interconnected to the trackingIn interface.

**Static OCS**

A static OCS is an OCS not coupled with a motion value. The position of a static OCS is always the OCS reference position.

A static OCS can be used to perform motions in a coordinate system displaced relative to the BCS and has been rotated.
Motion sequence

The motion sequence functionality permits the synchronous coupling of a kinematic end point with a coupled OCS. It contains the functions for the synchronization and coupling with a moving product on a conveyor.

3.14.2 Motion sequence – fundamentals

3.14.2.1 Defining an OCS reference position

The reference position of the OCS is defined compared to the BCS in the OCS basic frame. The OCS basic frame contains the translation of the Cartesian X-, Y-, and Z-axes and the subsequent rotation at the individual axes.

To define the reference position of the OCS, the translation is performed first:

![Diagram of OCS reference position](image)

Figure 3-67 Translation of the OCS compared to the BCS

The rotations are then performed in the following sequence:
1. Roll at the X axis
2. Pitch at the (already turned) Y axis
3. Yaw at the (already twice-turned) Z axis

Figure 3-68  Rotations of the OCS

The `_setPathObjectOcs()` command can be used to set the basic frame for each OCS on the path object. Either default values in the system variables can be used or other values specified directly.

Definition of the terminology

Figure 3-69  Schematic drawing of the motion sequence

OCS  Object coordinate system
BCS  Basic coordinate system
Frame  Translation and rotation of the OCS for the BCS
3.14.2.2 Assigning an OCS to a motion sequence reference value

The trackingIn input interconnection interface of the path object can be interconnected with another TO that provides an output interface with motion information. This can be, for example, the motion setpoint or actual value of an axis or the actual value of an external encoder.

The motion sequence value and the motion sequence reference value are assigned to the X-direction of the OCS. The OCS is coupled to the motion sequence reference value, the OCS coupling position is translated by the motion sequence value with regard to the OCS reference position in the X-direction of the OCS.

If the OCS is not interconnected or no TO is specified in the _setPathObjectOcs() command (trackingIn:=TO#NIL), the OCS then acts in its reference position.

If the kinematic end point is synchronized to a coupled OCS or is already synchronous with it (trackingIn <> TO#NIL and trackingState <> INACTIVE), the _setPathObjectOcs() command is not performed on this OCS and an error message issued. Before executing the command, the synchronized state (‘SYNCHRONIZED’ status) on this OCS must be ended.

See Terminate the coupling of the kinematic end point to a controlled OCS (‘desynchronize’) (Page 107) for further information.

3.14.2.3 Defining the translation of the position of the coupled OCS

Because normally a product-based programming is performed, the position of the OCS on the conveyor must be modified appropriately for the product position, i.e. translated.

The _redefinePathObjectOcs() command is used to translate the position of the OCS in the X-direction and so in the direction of the value on the motion sequence input.

If the kinematic end point is synchronized to a coupled OCS or is already synchronous with it (trackingIn <> TO#NIL and trackingState <> INACTIVE), the _redefinePathObjectOcs() command is not performed on this OCS and the 30002 error message issued. Before executing the command, the synchronized state (SYNCHRONIZED status) on this OCS must be ended.

The current position values of the coupled OCS and the motion sequence input will be displayed in the following system variables:
Motion sequence reference value
The `trackingInPosition` system variable contains the position value present at the motion sequence input of the OCS, the motion sequence reference value (the conveyor position).

Motion sequence value
The `trackingPosition` system variable contains the position of the coupled OCS, the motion sequence value.

\[ \text{trackingPosition} = \text{trackingInPosition} + \text{translation} \]

Figure 3-70  Current position of the OCS

Behavior of the OCS for modulo encoders
The motion sequence value indicates the continuing value on the motion sequence input without considering the modulo properties, i.e. the motion sequence value will not be reset when the motion sequence reference value on the modulo range end is reset, refer to the following figure.

Resetting trackingPosition
The `_redefinePathObjectOCS()` function can be used to set or translate `trackingPosition` only when the kinematic end point is not synchronous to this OCS or currently being synchronized (indicated using the `trackingState:=INACTIVE` variable).
There are 2 modes for translating the trackingPosition with the function \_redefinePathObjectOcs() - absolute or relative.

For the respective mode the value for trackingPosition can be calculated as follows:

- For \( \text{mode} := \text{RELATIVE} \)
  \[
  \text{trackingPosition} := \text{trackingPosition} + \text{value}
  \]

- For \( \text{mode} := \text{ABSOLUTE} \)
  \[
  \text{trackingPosition} := \text{trackingPosition} + \text{value}
  \]

The use of \( \text{mode} := \text{ABSOLUTE} \) has the advantage that translating always refers to trackingInPosition, which means that previous translations do not have to be buffered in the application. For \( \text{mode} := \text{RELATIVE} \) it is necessary to buffer the translation in the application.

The \_redefinePathObjectOCS() and \_setPathObjectOCS() functions are not executed for the associated OCS when it is in the 'SYNCHRONIZED' status.

See Terminate the coupling of the kinematic end point to a controlled OCS ('desynchronize') (Page 107) for further information.

### 3.14.2.4 Synchronizing motion on the path object to the coupled OCS

A handling application for which, for example, a product is to be fetched from a moving conveyor, is realized with an OCS coupled with the conveyor. The motion commands are configured here so that they act directly in the OCS.

This requires the motion calculated for the path object for the kinematic end point to be synchronized to the coupled OCS.

In the simplest case, after the synchronization, the kinematic end point moves with a defined point in the OCS and so with a point located on the conveyor. Furthermore, after the synchronization in the coupled OCS, linear, circular or polynomial paths can also be followed.

The \_enablePathObjectTrackingSuperimposed() command is used to synchronize the kinematic end point with an OCS coupled with the motion sequence reference value (e.g. position of a conveyor). Some of the arguments specified with the command:

- **Synchronization mode (synchronizingMode)**
  The following synchronization modes are available:
  - Other coupling with the position in the OCS specified in the command (setting: \( \text{synchronizingMode} := \text{IMMEDIATELY} \))
    Synchronization is executed immediately and coupled with the OCS.
  - Synchronization and coupling in the OCS at the position of the OCS specified in the command, i.e. as soon as the \( \text{ocsTrackingPosition} \) (e.g. of a conveyor belt) has reached a specified value (the synchronization position) (setting: \( \text{synchronizingMode} := \text{ON_POSITION} \)).
    For this synchronization mode, a preliminary synchronization is made to the specified synchronization position in the OCS.

- **The synchronization position (position)**
  The specification of a motion sequence value, above which travel synchronous with the OCS is to take place. This value is used only for \( \text{synchronizingMode} := \text{ON_POSITION} \).

The following applies for both synchronization modes:
The synchronization on the conveyor belt occurs where necessary superjacent to a motion that is still active in the BCS.

- No further motion commands are possible in the BCS during synchronization.
- Motion commands in the OCS are only possible once the status SYNCHRONIZED has been reached.

The desired position of the product in the OCS can be approached after the synchronization via a path command in the OCS.

The following applies for the synchronization mode ON_POSITION:

- The synchronization process will be aborted when the direction of the encoder value reverses during the synchronization.
- This can occur when the external encoder of a conveyor belt delivers a fluctuating position value during standstill due to missing filters or an insufficient tolerance window (with Extrapolation) which results in a direction reversal of the actual position.

**Synchronization status**

The synchronization status of the kinematic end point to a coupled OCS is indicated in the ocs[i].trackingState system variables on the OCS.

The synchronization status is SYNCHRONIZED when

- there is no motion active in the BCS, the speed of the kinematic end point is equal to the speed of the conveyor belt and the position misalignment of the synchronization motion resulting from the synchronization has been rectified.
- a motion is active in the BCS, the speed of the overlying synchronization motion is equal to the speed of the conveyor belt and the position misalignment of the synchronization motion resulting from the synchronization has been rectified.

A static OCS interconnected with TO#NIL always has the SYNCHRONIZED status. In addition to the static OCS, not more than one coupled OCS can have the SYNCHRONIZED status.

**Dynamic values for the synchronization action**

Dynamic values for the synchronization action can be specified in the _enablePathObjectTrackingSuperimposed () command.

The default dynamic values of the path object can be used or the values specified explicitly.

- The overall dynamics during the synchronization process result from the active path motion in the BCS (where necessary) and the overlying synchronization motion. This must be taken into consideration when specifying the dynamics, as otherwise this can cause the programmed or maximum dynamic values on the path object to be exceeded.

### 3.14.2.5 Performing path motions in the coupled OCS

Path motions can be related using the csType command parameter optionally to the BCS or an OCS.

Prerequisite for path commands in the OCS acting is that the OCS has the SYNCHRONIZED status. Otherwise the path motion command for the OCS will not be performed. Path motion commands for the OCS can only be issued after synchronization.
3.14.2.6 Terminate the coupling of the kinematic end point to a controlled OCS ('desynchronize')

The coupling of the kinematic end point to a controlled OCS is terminated by the coming into force of a path motion command related to the BCS or a static OCS.

Any existing path motion commands are discarded; the new path motion command will be executed immediately.

The system variables for PATH on the path object indicate the state of the path in the BCS or OCS coordinates system selected.

When revoking a synchronous motion sequence (conveyor synchronization) using _stopPath in the BCS, no path active and/or zero values are displayed for removal of the motion from the motion sequence.

3.14.2.7 Stopping in the OCS

The _stopPath() command can be stopped relative to the OCS. The SYNCHRONIZED status with the coupled OCS is retained. This means the motion can be continued using _continuePath() relative to the coupled OCS.

3.14.3 Motion sequence – sample application

3.14.3.1 Sample application of an OCS

The use of an OCS for the motion sequence is explained using a short example.

Figure 3-72 Overview of the sample application
In this example, products are placed on a conveyor. A sensor records the exact position of the products. The handling device should fetch products from the conveyor and place them at another location.

3.14.3.2 Defining the reference position of the OCS

The reference position of the OCS is defined in the system variables.

In this example, the OCS1 is used. The settings for this coordinate system are made in the `userdefaultocs[1]` structure.

Consequently, the OCS is displaced by 100 mm in the X-direction and 15 mm in the Z-direction:

The OCS is rotated by -15° at the Y-axis.
3.14.3.3 Determining the motion sequence reference value of the OCS

The position and motion data of the conveyor are acquired using the CONVEYOR_BELT external encoder. For the OCS, the base frame is set and activated with the CONVEYOR_BELT motion sequence reference value. The OCS is then coupled with the conveyor, in particular, at the position supplied by the CONVEYOR_BELT external encoder.

```plaintext
// Set OCS_1 to CONVEYOR_BELT,  
// for the BCS base frame for the OCS,  
// use the default settings.  
myRetDINT :=  
  _setPathObjectOcs(  
    pathObject:=Portal_3D,  
    ocsNumber:=1,  
    trackingIn:=CONVEYOR_BELT,  
    ocsSettingType:=USER_DEFAULT
  );
```

3.14.3.4 Defining the position of the OCS relative to the motion sequence reference value

The sensor, for example, a light barrier, is triggered by the passing product. The current value of the CONVEYOR_BELT external encoder is stored in the belt_position variable.

Because the position of the sensor related to the reference position of the OCS is known, the position of the product with regard to the motion sequence reference value is known.
3.14 Object coordinate systems and motion sequences on the path object

Figure 3-76  Defining the position of the OCS

```
myRetDINT :=
    _redefinePathObjectOcs(
        pathObject:=Portal_3D,
        ocsNumber:=1,
        mode:=RELATIVE,
        value:=belt_position - sensor_position
    );
```

3.14.3.5 Synchronizing motion on the path object to the coupled OCS

The handling device should be coupled synchronous with the product after a `synch_space` travel length after the sensor.

The acting point position of the grabber is specified in the OCS. In the example, this is done using the `offset_x`, `offset_y` and `offset_z` variables. This displacement is then used for positioning after synchronization by means of a command in the OCS so that the gripper can hold the product above its center of gravity.
3.14 Object coordinate systems and motion sequences on the path object

Figure 3-77  Synchronizing the handling device

myRetDINT :=
 ENABLEPATHOBJECTTRACKINGSUPERIMPOSED(
   pathObject:=Portal_3D,
   ocsNumber:=1,
   synchronizingMode:=ON_POSITION,
   position:=sensor_position + synch_space
);

When the status "synchronous" has been reached (ocs[1].trackingState = SYNCHRONIZED),
the command for positioning the gripper at the acting point of the product (offset_x, offset_y,
offset_z) can be issued in the OCS.
myRetDINT :=
 MOVEPATHLINEAR(
   pathObject:=Portal_3D,
   pathMode:=ABSOLUTE,
   x:=offset_x,
   y:=offset_y,
   z:=offset_z,
   csType:=OCS,
   csNumber:=1
);

3.14.3.6 Performing path motions in the coupled OCS

The handling device now moves itself 15 mm away from the conveyor and brings the product
to the placement position (dispose_x, dispose_y, dispose_z). The first motion occurs in the
OCS, the second in the BCS. The synchronization is terminated by calling the second
command.
myRetDINT :=
 MOVEPATHLINEAR(
   pathObject:=Portal_3D,
   pathMode:=RELATIVE,
   z:=15.0,
   csType:=OCS,
```plaintext
  csNumber:=1

  myRetDINT :=
    _movePathLinear(
      pathObject:=Portal_3D,
      pathMode:=ABSOLUTE,
      x:=dispose_x,
      y:=dispose_y,
      z:=dispose_z,
      cstype:=BCS
    );
```
3.15 Interconnection, interconnection rules

- Path axis interface 1 and path axis interface 2 of a path object must be interconnected with path axes.
- Path axis interface 3 of the path object can optionally be interconnected with a path axis, irrespective of the kinematic settings.
- The positioning axis interface for the path-synchronous motion can optionally be interconnected with a positioning axis.
- The MotionOut.x, MotionOut.y or MotionOut.z interface can optionally be interconnected with a positioning axis.
- The path object can be interconnected with a cam for specifying a velocity profile.
- To specify a motion sequence reference value, the TrackingIn interface can be interconnected with a TO that provides an output interface with position value.

For additional information, see Motion Control Basic Functions Function Manual, "Available technology objects".

Notes:
- The path interfaces cannot be distributed, i.e. all objects involved in the path group (path object, path axes, and positioning axes) must be on the same device.
- The objects involved in a path interpolation group (path object, path axes and, if applicable, a positioning axis) must be assigned to the same IPO or IPO_2 execution level. The SERVO setting is not possible.
- The currently effective interconnections are shown in the system variables specificVelocityProfile, motionBuffer, connections.
### 3.16 Simulation operation

A path interpolation can be switched to simulation, i.e. values are calculated on the path object but are not output to the slave axes/positioning axis.

It is possible to enable and disable the path simulation at any time, including while the relevant axes are in motion, provided there is no error response.

The simulation [ACTIVE/INACTIVE] system variable provides information about the simulation status of the path object.

**Commands for the simulation operation**

- The `enablePathObjectSimulation()` command sets the path interpolation to simulation mode.
  Values are calculated but are not output to the path axes/positioning axis. This can be done at any time.

- The `disablePathObjectSimulation()` command resets the path interpolation from simulation mode.
  Values are output to the path axes/positioning axis again.

  If there is a discrepancy between the axis setpoint calculated from the path interpolation and the current setpoint on the axis, the change in the axis setpoint on the axis is limited due to the maximum dynamic limits of the axis.

**Maintaining the setpoint calculation on the path object even when the axis enables are canceled**

The configuration data `decodingConfig.disablePathOperation` can be used to specify whether the setpoints on the path object will continue to be calculated even when the axis enables are canceled.

- If **NO** (default), the path interpolation is also canceled in simulation mode if the enables on the path axis/positioning axis have been canceled.

- If **YES**, the path interpolation is not canceled in simulation mode if the enables on the path axis/positioning axis have been canceled while the path object is in simulation mode.
  Any path commands that are undergoing execution are retained.
4.1 Selecting the path interpolation technology package

1. Select the device in the project navigator and select **Select technology packages** in the shortcut menu (right-click).

2. Select the **PATH** option and confirm with **OK**.

![Figure 4-1 Selection of technology packages](image)

The following technology objects could not be integrated due to:

- Incorrect version
- Missing technology package
4.2 Creating axes with path interpolation

- When creating an axis in SCOUT, enable the path interpolation technology.

![Insert Axis](image)

**Figure 4-2** Inserting an axis with path interpolation

**Notes**

When you specify the path interpolation technology for an Axis technology object, a path object is not inserted automatically.

A positioning axis, for example, cannot be changed into a path axis at a later point.

Interconnections are made on the path object.

See Interconnecting a path object (Page 127).
4.3 Creating a path object

In SIMOTION SCOUT, path objects are created at the same level as an Axis and a Cam technology object. These path objects can be assigned to all applicable axes of the device.

1. To create a new path object, double-click **Insert path object** under **PATH OBJECTS** in the project navigator. You can also copy an existing path object to the clipboard and then insert it under another name.

![Insert Path object dialog box](image)

Figure 4-3 Inserting a path object

2. Enter a name and, if necessary, the author, version, and comments, and click **OK** to confirm. The new path object will be inserted under **TECHNOLOGY**.
4.4 Representation in the project navigator

The path object appears in the project navigator at the same level as the Axis and Cam technology objects. Links symbolize the connection to path axes or a positioning axis for path-synchronous motion.

Figure 4-4  Project with path interpolation in the project navigator
4.5 Assigning path object parameters/default values

- In the project navigator, double-click Default under the path object.
  In this window, you define the substitute values (default) for calling the path object (_movePath..., _stopPath(), _setPathObjectOcs(), etc.).

**Default for object coordinate systems**

On the **Object coordinate systems**, you can define the default values for the three available OCS.

![Path object: Default - Object coordinate systems tab](image)

In this window, you can define the values for the object coordinate systems that are used by default.

For more information, refer to Object coordinate system (OCS) on the path object (Page 100).

**Dynamic response parameters**

You specify the path velocity, the velocity profile, and the acceleration/deceleration and jerk on the **Dynamic response** tab.
You define the substitute values (default settings) for the dynamic response in this window.

You can set the following parameters:

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Enter the substitute value for the path velocity here.</td>
</tr>
<tr>
<td>Velocity profile</td>
<td>Select the velocity profile here.</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Enter the substitute value for the path acceleration here.</td>
</tr>
<tr>
<td>Deceleration</td>
<td>Enter the substitute value for the path deceleration here.</td>
</tr>
<tr>
<td>Jerk</td>
<td>Enter the substitute value for the path jerk here.</td>
</tr>
</tbody>
</table>

For more information, refer to Path dynamics (Page 38).

The meaning of the parameters in the dialog box and their permissible value ranges can be found in the SIMOTION reference lists.

**Path parameters**

You specify the default settings for the path on the *Path* tab.
You define the substitute values (default settings) for the path in this window.

You can set the following parameters:

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path plane</td>
<td>Here, you specify the path plane: X_Y_Z / X_Y / Y_Z / Z_X (default: X_Y_Z, for 3D; for 2D kinematics, this is implicitly X_Y) (userDefault.path.plane)</td>
</tr>
<tr>
<td>Path mode</td>
<td>Specify the path mode: absolute or relative (userDefault.path.mode)</td>
</tr>
<tr>
<td>Limit dynamic path response to</td>
<td>Here, you specify whether the dynamic path response should be limited to the</td>
</tr>
<tr>
<td>transformed axis limit values</td>
<td>transformed axis limit values.</td>
</tr>
<tr>
<td></td>
<td>(userDefault.path.dynamicAdaption)</td>
</tr>
<tr>
<td></td>
<td>See Limiting the path dynamics (Page 39).</td>
</tr>
<tr>
<td>Polynomial default</td>
<td>Here you can define the type of polynomial interpolation.</td>
</tr>
<tr>
<td></td>
<td>(userDefault.path.polynomialMode)</td>
</tr>
<tr>
<td></td>
<td>See Polynomial paths (Page 32).</td>
</tr>
<tr>
<td>Circle default</td>
<td>Here you can define the type of circular interpolation.</td>
</tr>
<tr>
<td></td>
<td>(userDefault.path.circularType)</td>
</tr>
<tr>
<td></td>
<td>See Circular paths (Page 27).</td>
</tr>
</tbody>
</table>
| Direction of synchronous path      | Here, you specify the direction of the positioning axis for path-synchronous motion.
| motion                             | (userDefault.w.direction)                                                          |
|                                    | See Functionality of path-synchronous motion (Page 53).                            |
### Configuring the Path Object

#### 4.5 Assigning path object parameters/default values

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/Instruction</th>
</tr>
</thead>
</table>
| Mode of path-synchronous motion    | Here, you specify the synchronous axis mode:  
  - Absolute  
  - Relative  
  - Output path lengths  
  - Additive output of path lengths  
  (userDefault.w.mode)  
  See Functionality of path-synchronous motion (Page 53). |
| Blending                           | Here, you specify whether and how blending is performed.  
  (userDefault.blendingMode)  
  See Path behavior at motion end (Page 43). |
| Construction point mode            | Center point or intermediate point:  
  - Absolute  
  - Relative  
  - Same as target position mode  
  (userDefault.path.ijkMode)  
  See Circular paths (Page 27). |

**Note:**

Transformations are set via the Configuration (Page 123) form or via the expert list.  
For more information, refer to Basics of Path Interpolation (Page 19).  
The meaning of the parameters in the dialog box and their permissible value ranges can be found in the SIMOTION reference lists.
4.6 Configuring a path object

- In the project navigator, double-click Configuration under the path object.

**Configuration of the path object**

On the Configuration tab, you define the type of kinematics, the processing cycles, and, depending on the kinematics, the coordinate plane.

![Path object: Configuration - Configuration tab](image)

In this window, you can define the following parameters:

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing cycle clock</td>
<td>You define here whether the path object is processed in the IPO or in the IPO_2. All TOs (path object, path axes, positioning axis for path-synchronous motion) involved in a path interpolation grouping must be assigned to the same interpolation cycle! (Execution.executionlevel)</td>
</tr>
<tr>
<td>Kinematics</td>
<td>This is where you set the desired kinematics. (Kinematics.typeOfKinematics)</td>
</tr>
<tr>
<td>Coordinate plane</td>
<td>This is where you set the coordinate plane. This setting is not available for all kinematics. (config2D)</td>
</tr>
</tbody>
</table>

**Offset and rotation on the path object**

On the Offset tab, you define the translation or rotation of the kinematic zero point relative to the BCS zero point.
In this window, you can define the following parameters:

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset x</td>
<td>Here, you set the offset of the kinematic zero point. Depending on the kinematic type and/or set coordinate level, only certain fields exist. In this example, it is 2D kinematics or coordinate level XY. For this reason, the offset can only be set in the x and y direction. (basicOffset.x, basicOffset.y, basicOffset.z)</td>
</tr>
<tr>
<td>Offset y</td>
<td></td>
</tr>
<tr>
<td>Offset z</td>
<td></td>
</tr>
<tr>
<td>roll - rotation about x</td>
<td></td>
</tr>
<tr>
<td>pitch - rotation about y</td>
<td></td>
</tr>
<tr>
<td>yaw - rotation about z</td>
<td></td>
</tr>
</tbody>
</table>

Units used for the path object

On the Units tab, you define the units to be used on the path object.
Further information
For an overview of functions, refer to Overview of Path Interpolation (Page 13).
For a description of functions, refer to Basics of Path Interpolation (Page 19).
The meaning of the configuration data and the permissible value ranges can be found in the SIMOTION reference lists.
4.7 Defining limits

- In the project navigator, double-click **Limits** under the object.

![Limits on the path object](image)

In this window, you specify the maximum dynamic path limit values:

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Velocity</td>
<td>Enter the maximum velocity here. (<a href="#">limitsOfPathDynamics.velocity</a>)</td>
</tr>
<tr>
<td>Acceleration</td>
<td>Enter the maximum acceleration here. (<a href="#">limitsOfPathDynamics.positiveAccel</a>)</td>
</tr>
<tr>
<td>Deceleration</td>
<td>Enter the maximum deceleration here. (<a href="#">limitsOfPathDynamics.negativeAccel</a>)</td>
</tr>
<tr>
<td>Positive jerk</td>
<td>Here, you enter the maximum jerk during acceleration build-up / deceleration reduction. (<a href="#">limitsOfPathDynamics.positiveJerk</a>)</td>
</tr>
<tr>
<td>Negative jerk</td>
<td>Here, you enter the maximum jerk during acceleration reduction / deceleration build-up. (<a href="#">limitsOfPathDynamics.negativeJerk</a>)</td>
</tr>
</tbody>
</table>

For more information, refer to Path dynamics (Page 38).

The meaning of the configuration data and the permissible value ranges can be found in the SIMOTION reference lists.
4.8 Interconnecting a path object

- In the project navigator, double-click Interconnections under the object.

![Image of Interconnections window]

In this window, you interconnect the outputs of the path object with the path axes or with a position axis. (The objects must have already been created.)

Place a check mark beside the required objects.

A path object must be interconnected with at least two path axes.

The following connectors of the path object must be interconnected:
- **Path axis 1**: With a path axis
- **Path axis 2**: With a path axis

The following connectors of the path object can be interconnected:
- **Path axis 3**: With a path axis
- **Axis for path-synchronous motion**: With a position axis, synchronous axis, or path axis
• **Velocity profile**: With a cam
• **Output of the path length in w2** (as of V4.4): with a positioning axis

For more information, refer to Interconnection, interconnection rules (Page 113).

**Following the motion of**

As from V4.2, an axis or an external encoder can be interconnected for the motion sequence. For more information, refer to Object coordinate systems and motion sequences on the path object (Page 100).

**Specification of the dynamics**

As of V4.3, the path dynamics can be specified via DynamicsIn. For more information, refer to Preset path dynamics (Page 38).
4.9 Path control panel

The path control panel is used to commission kinematics. It is therefore possible to control and monitor path objects without a user program. Applications include the following:

- Enabling and disabling all connected axes
- Function test of path objects and the connected axes
- Commissioning and setting up the kinematics
- Cartesian traversing in the BCS or in an OCS
- Traversing in machine axes
- Monitoring axis positions and Cartesian coordinates

control priority

The control priority for the path object and the connected axes can either be with the path control panel or with the user program. If the control priority is with the path control panel, motion commands on the path object and the axes involved are refused.

⚠️ WARNING

Danger to life from unexpected machine movement

Failure to observe the safety instructions can result in personal injury and material damage.
- Note the safety instructions in the Assume Control Priority SCOUT dialog box.

Requirements

- An online connection to the SIMOTION device must be established.
- The current configurations of the axes and the path object must be on the target device. If necessary, download the project or upload the configuration data for alignment purposes (Target system > Load > Load configuration data to PG).

You can find additional information in the SCOUT online help (index: Path control panel).
4.10 Configuring kinematic adaptation in the expert list

All configuration data and system variables for the Path object technology object can be displayed and changed in the expert list.

Here, you define the kinematic type and adapt it for your requirements (see Kinematic adaptation (Page 56)).

For additional information, see Motion Control Basic Functions Function Manual, "Expert list".
4.11 Configuring path monitoring

Path monitoring can be configured on the axis.

In the project navigator, double-click **Monitoring** under the axis object.

Set the required parameters on the **Path motion** or **Synchronous path motion** tab.

---

### Path motions

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setpoint monitoring</td>
<td>Here, you activate the setpoint monitoring for the path axis.</td>
</tr>
<tr>
<td></td>
<td>(pathAxisPosTolerance.enableCommandValue)</td>
</tr>
<tr>
<td>Setpoint tolerance</td>
<td>Here, you specify the permissible deviation of the setpoint on the axis calculated by the path object for the path axis, taking into account the limits of the executable setpoint.</td>
</tr>
<tr>
<td></td>
<td>(pathAxisPosTolerance.commandValueTolerance)</td>
</tr>
<tr>
<td></td>
<td>An alarm will be initiated if exceeded.</td>
</tr>
</tbody>
</table>

### Path-synchronous axis - monitoring functions - path-synchronous movement

<table>
<thead>
<tr>
<th>Field/button</th>
<th>Meaning/instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setpoint monitoring</td>
<td>Here, you activate the setpoint monitoring for the positioning axis.</td>
</tr>
<tr>
<td></td>
<td>(pathSyncAxisPosTolerance.enableCommandValue)</td>
</tr>
<tr>
<td>Setpoint tolerance</td>
<td>Here, you specify the permissible deviation of the setpoint on the axis calculated by the path object for the positioning axis, taking into account the limits of the executable setpoint.</td>
</tr>
<tr>
<td></td>
<td>(pathSyncAxisPosTolerance.commandValueTolerance)</td>
</tr>
<tr>
<td></td>
<td>An alarm will be initiated if exceeded.</td>
</tr>
</tbody>
</table>

For additional information, see Display and monitoring options on the axis (Page 50).
4.12 Calibrate path object

You calibrate the path object in this window by re-aligning the coordinate systems. The basic coordinate system can be aligned with the kinematic zero point or an object coordinate system with the basic coordinate system.

The following five steps are required for the example:

1. Select an OCS
2. If required, accept the current position of the kinematics
3. Specify the actual positions and the position setpoints of the kinematics
4. Start the calculation of the offset and the rotation
5. If required, accept the calculated values as the new values for the kinematics

Step 1 is omitted when calibrating the basic coordinate system.

Calibrating the basic coordinate system (BCS)

You calibrate the BCS to the kinematic zero point on the Basic coordinate system tab. Depending on the dimensions of the selected kinematics, you must specify two or three points in the BCS and two or three corresponding new points from which the offset and the rotation then result. Input fields for at least two coordinates are available for each point. With 3D kinematics, there is also an additional input field for the third coordinate. Individual points in the BCS can also be taken over directly from the current position of the path object using the button to the left of the input fields.
After entering valid coordinates, a button appears to calculate the offset and rotation for the BCS. The calculated values are displayed and can be transferred to the offline data management via a further button for the path object. If the checkbox for manual adaptation is activated, the calculated values can be subsequently edited.

You can open the path control panel via the black and yellow button. This button is only available when the controller is in online mode.

**Calibrating the object coordinate system (OCS)**

You calibrate the OCS_1, OCS_2 and OCS_3 object coordinate systems on the *Object coordinate systems* tab. The respective OCS is selected in the area on the left. The calibration is similar to that of the BCS to the kinematics zero point. The difference is that the inputs refer to reference positions in the BCS and positions in the respective OCS.

It is also possible to take over the points of a previously calibrated OCS for another OCS. The appropriate buttons are on the left.
## 4.13 Path interpolation - context menu

You can select the following functions:

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration</td>
<td>This function opens the configuration for the path object selected in the project navigator. Define the processing cycle clock in this window.</td>
</tr>
<tr>
<td>Expert list</td>
<td>This function opens the expert list for the path object selected in the project navigator. The configuration data and system variables can be displayed and changed in this list.</td>
</tr>
<tr>
<td>Default</td>
<td>This function opens the defaults for the path object selected in the project navigator. You define the substitute values for the object coordinate systems, the dynamic response, and the path in this window.</td>
</tr>
<tr>
<td>Calibration</td>
<td>With this function, you open a window for calibrating the path object. In this window you can calibrate the values for the BCS or the three OCS.</td>
</tr>
<tr>
<td>Limits</td>
<td>This function opens the limits for the path object selected in the project navigator. You define the dynamic limits for the path object in this window.</td>
</tr>
<tr>
<td>Interconnections</td>
<td>This function opens the interconnections for the path object selected in the project navigator. You can see the inputs of the axes in this window.</td>
</tr>
<tr>
<td>Expert</td>
<td>This function opens the submenu for the expert settings.</td>
</tr>
<tr>
<td>Import object</td>
<td>Use Import object to open a window for the XML import. You can define the parameters for the XML import in this window.</td>
</tr>
<tr>
<td>Save project and export object</td>
<td>Use Save project and export object to open a window for the XML export. You can define the parameters for the XML export in this window.</td>
</tr>
</tbody>
</table>
5.1 Overview of the example

To illustrate how the path interpolation is configured, the following sections describe a sample project step-by-step.

The following descriptions assume that the project with the SIMOTION device and the drives have already been created in the HW Config.

In this example, the following 2D gantry is created:

This 2D gantry comprises the following axes:

- Vertical axis: 1400 cm traversing length, axis zero point at the lowest position
- Horizontal axis: 2000 cm traversing length, axis zero point at the left-hand side

This means the zero point of the path object is at the lower left.

How to create and use the 2D gantry:

- Select technology package (Page 137)
- Create axes. (Page 138)
- Creating a path object (Page 140)
- Defining the kinematics (Page 141)
- Interconnecting a path object (Page 142)
- Programming a path in MCC (Page 145)
- Checking a motion with trace (Page 163)
To show how a path-synchronous axis (Page 164) operates, a gripper will also be created at the end of the example.
5.2 Select technology package

The PATH and CAM_EXT technology packages support path interpolation.

Right-click the device in the project navigator, select Select technology package in the context menu and use the PATH technology package.
5.3 Create axes.

The example project requires two linear axes: \texttt{Axis\_X} and \texttt{Axis\_Z}. These axes are created with the \textit{path interpolation} technology. Both axes are created as linear, virtual, non-modular axes.

Click \texttt{AXES -> Insert axis} to add an axis to the device. Name the first axis \texttt{Axis\_X} and the second axis \texttt{Axis\_Z}, and set up the two as follows:

1. \textit{Path interpolation} technology selected

![Figure 5-3 Creating an axis](image)

2. Linear, virtual, non-modular axis
   
   The configuration of the two axes is as follows:

   ![Figure 5-4 Sample summary of the configuration (Axis\_X axis)](image)

   The project navigator should now look like this:
5.3 Create axes.

Figure 5-5  Project navigator with created axes
5.4 Creating a path object

Insert a new path object for the device under PATH OBJECTS. Name the path object Portal_2D.
5.5 Defining the kinematics

For the definition of the kinematics, the kinematics type with its mechanical data and the displacement of the coordinate system at the zero point of the base coordinate system are specified.

Open the Configuration window for the path object. In this form, you can set the type of kinematics and the coordinate plane as follows:

- Kinematics: Cartesian 2D
- Coordinate plane: ZX plane

![Configuration of the kinematics](image)

Figure 5-7 Configuration of the kinematics
5.6 Interconnecting a path object

Open the Interconnections window for the path object. In this screen form, assign the axes to the path object.

Because the kinematics operate in the Z-X plane, the Z-axis must be used as first path axis and the X-axis as second path axis.

Parameterize the interconnections of the path object as follows:

- 1. Path axis: Axis_Z
- 2. Path axis: Axis_X

Figure 5-8 Interconnecting a path object
5.7 Setting the default settings of the path object

The settings described below must be made for the default of the path object.

How to set the defaults for the path object:

1. For the path object, click Default.
2. In the Default window, on the Object coordinate systems tab, you can set the object coordinate system:

![Object coordinate systems](image)

If you want to activate an object coordinate system, you must call the command `setPathObjectOcs` (see Object coordinate system (OCS) on the path object (Page 100)).
3. On the **Dynamic response** tab, you can set the path object, for example, as follows:

<table>
<thead>
<tr>
<th>Object coordinate systems</th>
<th>Dynamic response</th>
<th>Path tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path plane</td>
<td>$X$ plane</td>
<td></td>
</tr>
<tr>
<td>Path mode</td>
<td>Absolute value specification</td>
<td></td>
</tr>
<tr>
<td>Limitation of the dynamic path response to the transformed axis limits:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Polynomial specification:</td>
<td>Explicit initial values</td>
<td></td>
</tr>
<tr>
<td>Circle specification:</td>
<td>Circle interpolation with intermediate and end point</td>
<td></td>
</tr>
<tr>
<td>Direction of path-synchronous motion:</td>
<td>From sign of the target position</td>
<td></td>
</tr>
<tr>
<td>Mode of path-synchronous motion:</td>
<td>Relative target position</td>
<td></td>
</tr>
<tr>
<td>Blending:</td>
<td>Blending with consideration of the dynamic axis response</td>
<td></td>
</tr>
<tr>
<td>Construct, point mode:</td>
<td>Value specification like target position</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-10  Default - dynamic response

4. Make the following settings on the **Path** tab:

<table>
<thead>
<tr>
<th>Object coordinate systems</th>
<th>Dynamic response</th>
<th>Path tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path plane</td>
<td>$X$ plane</td>
<td></td>
</tr>
<tr>
<td>Path mode</td>
<td>Absolute value specification</td>
<td></td>
</tr>
<tr>
<td>Limitation of the dynamic path response to the transformed axis limits:</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Polynomial specification:</td>
<td>Explicit initial values</td>
<td></td>
</tr>
<tr>
<td>Circle specification:</td>
<td>Circle interpolation with intermediate and end point</td>
<td></td>
</tr>
<tr>
<td>Direction of path-synchronous motion:</td>
<td>From sign of the target position</td>
<td></td>
</tr>
<tr>
<td>Mode of path-synchronous motion:</td>
<td>Relative target position</td>
<td></td>
</tr>
<tr>
<td>Blending:</td>
<td>Blending with consideration of the dynamic axis response</td>
<td></td>
</tr>
<tr>
<td>Construct, point mode:</td>
<td>Value specification like target position</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-11  Default - path
5.8 Programming the path interpolation in MCC

5.8.1 Programming the travel commands in MCC

The following path should be created for this example:

This path consists of the following path segments:

<table>
<thead>
<tr>
<th>Segment</th>
<th>Path type</th>
<th>Start point (x,z)</th>
<th>End point (x,z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A - B</td>
<td>Linear path</td>
<td>(0,0)</td>
<td>(0,1200)</td>
</tr>
<tr>
<td>B - C</td>
<td>Polynomial path</td>
<td>(0,1200)</td>
<td>(200,1400)</td>
</tr>
<tr>
<td>C - D</td>
<td>Linear path</td>
<td>(200,1400)</td>
<td>(1800,1400)</td>
</tr>
<tr>
<td>D - E</td>
<td>Polynomial path</td>
<td>(1800,1400)</td>
<td>(2000,900)</td>
</tr>
<tr>
<td>E - F</td>
<td>Linear path</td>
<td>(2000,900)</td>
<td>(2000,0)</td>
</tr>
<tr>
<td>F - A</td>
<td>Circular path</td>
<td>(2000,0)</td>
<td>(0,0)</td>
</tr>
</tbody>
</table>
5.8.2 Creating the program

1. In the project navigator, click Insert MCC source file. Name the MCC source file MCC_Example.

![Figure 5-13 Creating an MCC source file](image)

2. Define the following variables in the Interface of the MCC source file:
   - `start_move` (BOOL, true): The gantry should perform the motion when `start_move=true` is set and stop when `start_move=false`.
   - `forw_back` (BOOL, false): The `forw_back` variable indicates whether the gantry moves forwards (true, A->F) or backwards (false, F->A).
5.8 Programming the path interpolation in MCC

Figure 5-14  Defining variables in the MCC source file
3. In the project navigator, click **Insert MCC chart** for the new MCC source file. Name this chart **TopLoader**.

![Inserting an MCC chart](image)

**Figure 5-15** Inserting an MCC chart

4. Open the MCC chart and define the following variables.

   - `endPoly, startPoly`: `structRetGetLinearPathGeometricData`
   - `x_start, z_start, x_end, z_end`: `LREAL`

   These variables are used for calculating the data for the polynomial interpolation.

![Creating variables in the MCC source file](image)

**Figure 5-16** Creating variables in the MCC source file
5.8.3 Programming the traversing program

5.8.3.1 Programming the traversing program

The following traversing program should be programmed:

The traversing program should be performed within a While loop. It will be performed while start_move is set to true.
5.8.3.2 Creating a WHILE loop

For the example, a While loop is created that is run while `start_move` is set to true.

![WHILE loop](image)

Figure 5-18  WHILE loop

5.8.3.3 Programming the A - B linear path

Before starting the forwards motion, the `forw_back` direction flag is set to true. This is performed using a variable assignment.

To do this, place an ST zoom-in command within the While loop and program it as follows:
Figure 5-19  Set forw_back to true

Then add a **travel linear path** command to the While loop. Define the **A-B** linear path as follows:
5.8.3.4 Programming the B-C polynomial path

To program the B-C polynomial path, the geometric derivatives for the start and end points must be calculated in an ST zoom-in.

Add an ST zoom-in command for both the start point and the end point in the While loop, and program it as shown:
Figure 5-21 Calculating the derivatives at the start point of the first polynomial path

Figure 5-22 Calculating the derivatives at the end point of the first polynomial path

Now add a **travel polynomial path** command to the While loop and define the polynomial path as follows:
The previously calculated derivatives are specified as follows in the command:

- First derivative at the start point: `startpoly.firstGeometricDerivative.x / .y / .z`
- Second derivative at the start point: `startpoly.secondGeometricDerivative.x / .y / .z`
- First derivative at the end point: `endpoly.firstGeometricDerivative.x / .y / .z`
- Second derivative at the end point: `endpoly.secondGeometricDerivative.x / .y / .z`

**Note**

An alternative polynomial assignment is described for the programming of the D-E polynomial path.
5.8.3.5 Programming the C-D linear path

For the C-D linear path, add the **travel linear path** command to the While loop and make the following settings:

![Programming the C-D linear path](image)

Figure 5-24  Programming the C-D linear path
5.8.3.6 Programming the D-E polynomial path

The attach continuously type of polynomial specification is used for the D-E polynomial path. The geometric deviations of the start point are calculated using the previous path segment. Only the deviations for the end point need to be calculated using an ST zoom-in.

Both the ST zoom-in command and the travel polynomial path command must be added successively to the While loop.

Parameterize the polynomial path as shown.
Figure 5-26  Programming the D-E polynomial path
5.8.3.7 Programming the E-F linear path

For the E-F linear path, add the travel linear path command and make the following settings:

![Image of programming interface]

Figure 5-27  Programming the E-F linear path

5.8.3.8 Programming the F-A return travel

The gantry grabber should return to the initial position taking a circular path. The start point of the circular path is (2000, 0), the end point is (0, 0).

There are several ways of defining the circular path. For this example, the circular path should be defined using an intermediate point and the end point. The point (1000, 1000) is chosen as intermediate point.
To indicate the reverse motion, the `forw_back` variable is set to `false` in an ST zoom-in command.

![Figure 5-28 Defining the return circular path](image)

To indicate the reverse motion, the `forw_back` variable is set to `false` in an ST zoom-in command.

![Figure 5-29 Set forw_back to false](image)

The `travel circular path` command is then added and the following settings made:
5.8.4 Activating the axis enables and homing the axes

To move the axes, an enable must be made for each of them. The axes must also be homed. This requires that the the **enable axis** and **home axis** commands are added for each axis before the While loop.

You can use the default values for the enable axis commands.

For the home axis commands, set the home coordinates to 0 mm. The other values do not need to be changed.
5.8.5 MCC diagram

The MCC chart now has the following form:

![MCC Diagram]

Figure 5-31 MCC chart

5.8.6 Assigning MCC chart in the execution system

The MCC chart must be assigned in the execution system to any MotionTask. The MotionTask must be activated after the StartupTask.

To assign the MCC chart to a MotionTask, proceed as follows:
1. In the project navigator, select any MotionTask and move the MCC_Example.toploader program to the used programs.

![Figure 5-32 Assigning the MotionTask in the execution system](image)

2. In the task configuration of the MotionTask, select activation after StartupTask.

![Figure 5-33 Configuring the MotionTask in the execution system](image)
5.8.7 Checking a motion with trace

To see how the motion runs, a trace of the following variables is defined:

- TO.Axis_X.positioningstate.actualposition
- TO.Axis_Z.positioningstate.actualposition
- Forw_back

A log of the motion loop now has the following form:

![Graph showing motion trace](image)

Figure 5-34 Result of the example as trace
5.9 Creating a path-synchronous axis

To show the functionality of the path-synchronous motion, a path-synchronous axis is added to the project. The path-synchronous axis is used, for example, to additionally rotate products during the motion. The path-synchronous axis is to rotate the gripper by 90° between the C and D points.

Creating an axis

Add a positioning axis with the name **Axis_Sync** to the project. This axis is created as linear, virtual, non-modular axis. Note that the path axis functionality is not used for this axis.

Creating a path object

For interconnections of the path object, select the **Axis_Sync** axis as positioning axis for path-synchronous motions.
Figure 5-37  Interconnecting the "Axis_Sync" axis

Modifying MCC charts
Perform the following changes in the MCC chart:
Before the While loop, add the `enable axis` and `home axis` commands for the Axis_Sync axis analog to those for the X- and Z-axis.

- The Axis_Sync axis should also rotate synchronously in the C-D linear path. To rotate the axis as required, open the travel command for the C-D linear path and select the `Synchronous axis` tab. Make the following settings there:

![Figure 5-38 Rotating the "Axis_Sync" axis synchronously](image)

- In the F-A circular path, the Axis_Sync axis should be rotated back to the zero position.
Figure 5-39  Moving the synchronous path back to the zero position
6.1 Programming

6.1.1 Programming: Overview

The following section provides information about the commands and the alarm responses of the path object technology package. Further information is contained in the reference lists of the technology packages.

The description of the functions for the "Toploading" standard library based on the TO path object is contained on the "Application Toploading" function description.

6.1.2 Overview of commands

6.1.2.1 Information and conversion

Calculating the path length:
The following commands calculate the path length without starting or executing the path motion. The start and end points must be specified in the command.

- \_getLinearPathData()
- \_getCircularPathData()
- \_getPolynomialPathData()

Geometric path analysis:
The commands listed below are used to calculate Cartesian path data, such as path direction and path curvature, at the start point, endpoint, and a specifiable point on the path without starting or executing the path motion.

The point on the path is specified by means of the default setting of the path length distance to the start point.

The start point is specified in the command, as is the position with reference to the path length where the information data is determined.

- \_getLinearPathGeometricData()
- \_getCircularPathGeometricData()
- \_getPolynomialPathGeometricData()

With SIMOTION V4.2 and higher, the calculated geometry is stored in a geometry cache and a cacheID is provided as the return value. If the path segment is queried several times and
only the queried point changes, this cacheID can be used to access the data already calculated. This can greatly improve the performance of the information functions.

### 6.1.2.2 Conversion commands

- `_getPathCartesianPosition()`
  Calculates the Cartesian positions from the axis positions.

- `_getPathAxesPosition()`
  Calculates the axis positions from the Cartesian positions.

- `_getPathCartesianData()`
  Calculates the Cartesian data for position, velocity, and acceleration from the axis positions, axis velocities, and axis accelerations.

- `_getPathAxesData()`
  Calculates the axis positions, axis velocity, and axis accelerations from the Cartesian data for position, velocity, and acceleration.

### 6.1.2.3 Command tracking

The current processing and motion status of motion commands can be tracked using the following commands. Permanent storage of the states associated with a `CommandId` makes it possible to evaluate them beyond the lifetime of the motion command.

- `_getStateOfPathObjectCommand()`
- `_getMotionStateOfPathObjectCommand()`
- `_bufferPathObjectCommandId()`
- `_removeBufferedPathObjectCommandId()`

### 6.1.2.4 Motion

- `_movePathLinear()`
  Interpolation of linear paths (Page 26)
  - 2D in a main plane
  - 3D

- `_movePathCircular()`
  Interpolation of circular paths (Page 27)
  - 2D in a main plane with radius, end point, and orientation
  - 2D in a main plane with center point and angle
  - 2D with intermediate and end points
  - 3D with intermediate and end points

- `_movePathPolynomial()`
  Interpolation of polynomial paths (Page 32)
  - 2D in a main plane
  - 3D
6.1.2.5 Object and Alarm Handling

- **cancelPathObjectCommand()**
  Cancels a command that is waiting or active in the IpO. The command to be canceled is specified by the indication of its CommandId in the `commandToBeCancelled` parameter.

- **enablePathObjectSimulation()**
  Places a path object in simulation mode, path values are calculated, but are not output on the axes.

- **disablePathObjectSimulation()**
  Ends simulation mode.

- **resetPathObject()**
  This command resets the path interpolator to its initial state. Active commands are stopped, commands are aborted, and errors are reset. If required, system variables can be reset to their default values or configuration data can be read in again.

- **resetPathObjectError()**
  This command resets all errors or a specific error on the path object.

- **getPathObjectErrorNumberState()**
  Reads out the status of a specific error.

- **getPathObjectErrorState()**
  This command provides information on whether alarms are pending on the path object and if so, how many. It also outputs information about these errors.

- **resetPathObjectConfigDataBuffer()**
  Resets the configuration data buffer.

- **getStateOfPathObjectMotionBuffer()**
  Returns the status of the command queue of the path object.

- **resetPathObjectMotionBuffer()**
  Clears all pending commands from the command queue. The 030002 Command aborted alarm is issued for each of the deleted commands.

6.1.2.6 Object coordinates

- **enablePathObjectTrackingSuperimposed()**
  Starts the synchronization action of a path object on an OCS.

- **getPathObjectBCSFromOCSData()**
  Calculates a position (x, y, z) in the base coordinate system of the path object using the position in the object coordinate system.

- **getPathObjectOCSFromBCSData()**
  Calculates a position (x, y, z) in the object coordinate system using the position in the base coordinate system of the path object.
6.1 Programming

- **_redefinePathObjectOCS()**
  Displaces the object coordinate system along the X-axis (travel direction).

- **_setPathObjectOCS()**
  Defines the translational and rotatory displacement of the object coordinate system compared with the base coordinate system of the path object.

### 6.1.3 Command execution

#### 6.1.3.1 Command buffer

The path object has three command buffers for every command.

- One buffer for motion commands has an immediate (**IMMEDIATELY**) and sequential (**SEQUENTIAL**) effect
- A separate buffer for stopping path motion (**_stopPath()**) and continuing path motion (**_continuePath()**)
- A buffer for other (i.e. superimposed) instructions

<table>
<thead>
<tr>
<th>Command</th>
<th>Function</th>
<th>Position*</th>
</tr>
</thead>
<tbody>
<tr>
<td>_movePathLinear()</td>
<td>Starts linear path motion</td>
<td>4</td>
</tr>
<tr>
<td>_movePathCircular()</td>
<td>Starts circular path motion</td>
<td>4</td>
</tr>
<tr>
<td>_movePathPolynomial()</td>
<td>Starts polynomial path motion</td>
<td>4</td>
</tr>
<tr>
<td>_stopPath()</td>
<td>Stops motion</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>_continuePath()</td>
<td>Resume motion</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>_getLinearPathData()</td>
<td>Linear path length</td>
<td>5</td>
</tr>
<tr>
<td>_getCircularPathData()</td>
<td>Circular path length</td>
<td>5</td>
</tr>
<tr>
<td>_getPolynomialPathData()</td>
<td>Polynomial path length</td>
<td>5</td>
</tr>
<tr>
<td>_getPathCartesianPosition()</td>
<td>Axis to path</td>
<td>5</td>
</tr>
<tr>
<td>_getPathAxesPosition()</td>
<td>Path to axis</td>
<td>5</td>
</tr>
<tr>
<td>_getPathCartesianData()</td>
<td>Axis to path with dynamic response data</td>
<td>5</td>
</tr>
<tr>
<td>_getPathAxesData()</td>
<td>Path to axis with dynamic response data</td>
<td>5</td>
</tr>
<tr>
<td>_getLinearPathGeometricData()</td>
<td>Geometric linear path data</td>
<td>5</td>
</tr>
<tr>
<td>_getCircularPathGeometricData()</td>
<td>Geometric circular path data</td>
<td>5</td>
</tr>
<tr>
<td>_getPolynomialPathGeometricData()</td>
<td>Geometric polynomial path data</td>
<td>5</td>
</tr>
<tr>
<td>_enablePathObjectSimulation()</td>
<td>Places path object in simulation mode</td>
<td>3</td>
</tr>
<tr>
<td>_disablePathObjectSimulation()</td>
<td>Resets the path object out of simulation</td>
<td>3</td>
</tr>
<tr>
<td>_resetPathObject()</td>
<td>Resets the path object</td>
<td>5</td>
</tr>
<tr>
<td>_resetPathObjectError()</td>
<td>Reset error</td>
<td>5</td>
</tr>
<tr>
<td>_getStateOfPathObjectCommand()</td>
<td>Read out command status</td>
<td>5</td>
</tr>
<tr>
<td>_getMotionStateOfPathObjectCommand()</td>
<td>Read out motion phase of a command</td>
<td>5</td>
</tr>
<tr>
<td>Command</td>
<td>Function</td>
<td>Position*)</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>------------</td>
</tr>
<tr>
<td>_bufferPathObjectCommandId()</td>
<td>Permanently stores the command ID</td>
<td>5</td>
</tr>
<tr>
<td>_removeBufferedPathObjectCommandId()</td>
<td>Terminates permanent storage</td>
<td>5</td>
</tr>
<tr>
<td>_getPathObjectErrorNumberState()</td>
<td>Reads out the status of a path object error</td>
<td>5</td>
</tr>
<tr>
<td>_getPathObjectErrorState()</td>
<td>Reads out the status and number of the pending path object error</td>
<td>5</td>
</tr>
<tr>
<td>_resetPathObjectConfigDataBuffer()</td>
<td>Deletes that configuration data collected in the buffer since the last activation</td>
<td>5</td>
</tr>
<tr>
<td>_getStateOfPathObjectMotionBuffer()</td>
<td>Returns the status of the command queue of the path object.</td>
<td>5</td>
</tr>
<tr>
<td>_resetPathObjectMotionBuffer()</td>
<td>Clears all pending commands from the command queue.</td>
<td>5</td>
</tr>
<tr>
<td>_enablePathObjectTrackingSuperimposed()</td>
<td>Starts the synchronization action of the path object on an OCS.</td>
<td>3</td>
</tr>
<tr>
<td>_getPathObjectBCSFromOCSData()</td>
<td>Calculates a position in the BCS using a position in the OCS.</td>
<td>5</td>
</tr>
<tr>
<td>_getPathObjectOCSFromBCSData()</td>
<td>Calculates a position in the OCS using a position in the BCS.</td>
<td>5</td>
</tr>
<tr>
<td>_redefinePathObjectOCS()</td>
<td>Displaces the OCS along the X-axis.</td>
<td>3</td>
</tr>
<tr>
<td>_setPathObjectOCS()</td>
<td>Defines the displacement of the OCS compared with the BCS.</td>
<td>3</td>
</tr>
<tr>
<td>_cancelPathObjectCommand()</td>
<td>Cancels a command that is waiting or active in the Ipo.</td>
<td>1</td>
</tr>
</tbody>
</table>

*) Legend:

1  Buffer for Stop-Continue commands
2  Not used
3  Buffers for superimposed commands
4  Sequential command buffer
5  Not assigned to the command buffers (commands can be executed in parallel)

### 6.1.3.2 Override behavior

The response to command insertion is defined in the `mergeMode` command parameter.

- Override current interpolator command
  (`mergeMode:=IMMEDIATELY`)
- Overwrite commands in the buffer
  (`mergeMode:=NEXT_MOTION`)
  This command is executed as soon as the current interpolator command has been executed.
- Append to commands already in the buffer
  (`mergeMode:=SEQUENTIAL`)
  If the buffer is full, the command can wait until an entry becomes available or the command execution can continue without a wait time.
Commands are decoded in the task context, i.e. in the execution level of the user task that issued the command (before it is entered in the command buffer).

mergeMode has the same settings and action as in the Axis technology object.

6.1.4 Interactions between the path object and the axis

6.1.4.1 Override behavior

The response to other active motions on the axis is defined with the mergeMode parameter of the path object motion command.

With the substitute setting, all active assigned axis motions are canceled (as a function of the transferSuperimposedPosition setting in the configuration data).

- When a path motion is substituted by an additional path command, an immediate transition takes place to the path that yields the new end point at the point of the substitution.

- When a path is substituted by a motion command, such as _move...(), the other axes involved in the path motion and, if applicable, the positioning axis for path-synchronous motion stop with the maximum dynamic values.
  The action of the other override responses is the same as for synchronous operation.
  The following applies when motion commands occur simultaneously on the respective object within one interpolation cycle, i.e. one on the synchronous object, one on the path object, and one on the axis:
    - When mergeMode=SEQUENTIAL/NEXT_MOTION, the synchronous/path command is executed.
    - When mergeMode=IMMEDIATELY, the command on the axis is executed.

- The _stop() command from an axis involved in the path motion or participating via the path-synchronous motion does not stop the path motion on the path object, i.e. it has no effect (this is the same behavior as for the synchronous motion on the Synchronous technology object).
  The _stop() only stops motions that were initiated on the axis.
  The _stopEmergency() command is also effective on motions initiated on the Synchronous operation and Path object technology objects.

- Superimpositions can only take place on the axis (axis motion or synchronous operation), not on the path object.

- If a positioning axis for path-synchronous motion of the path group is overridden by means of an axis command on this axis, this has the same effect on the path as when a path axis is overridden.

- Path axes and positioning axes for path-synchronous motion have the same response.

With the other settings, the path motion is started after the end of all previous axis motions or an active path motion.
6.1.4.2 Sequence of effectiveness

Technology objects are processed in the sequence: Path object - Synchronous object - Axis object. In the case of simultaneous motion commands on several technology objects that are interconnected with an axis and have the same override response, the motion commands take effect according to the processing order and the setting in the mergeMode parameter:

- When mergeMode:=IMMEDIATELY, the motion command on the axis takes effect (last processed command).
- When mergeMode:=SEQUENTIAL/NEXT_MOTION, the motion command on the path object takes effect (first effective command).

6.1.4.3 Interaction with the axis

If the motion of an axis is stopped as a result of its local alarm response or if the interconnection of the path object to the axis becomes invalid, the path motion is canceled, the other axes are also traversed with the maximum dynamic values to velocity 0.0, and a technological alarm is issued.

If the remaining distance is smaller than the deceleration distance, the new target position is overshot, and the axis travels back to the target position (with a reversing motion). The other axes travel with their maximum dynamic values to velocity 0.0. However, these axes do not travel back to the cancellation point. Depending on the general conditions (number of participating axes, dynamic values), the path is no longer maintained.

See Motion Control Technology Objects Axis Electric/Hydraulic, External Encoder Function Manual, "Motion transitions"

6.1.4.4 Interactions with other path motions

If a path axis is interconnected with several path objects, and if a path motion on a path axis substitutes another path motion on another path object, the other path axes are traversed with the maximum dynamic values to velocity 0.0, and a technological alarm is issued.
6.2 Local alarm response

Local alarm responses are specified by means of the system. The following responses are possible:

- **NONE**: No response
- **DECODE_STOP**: Command decoding is canceled, but the current motion and command in the buffer remain active.
- **END_OF_MOTION_STOP**: Abort at end of error-causing command; motion on the path stops.
- **MOTION_STOP**: Controlled motion stop with programmed dynamic path values on the path. Motion can be continued by acknowledging the error.
- **MOTION_EMERGENCY_STOP**: Controlled motion stop with maximum dynamic path values/limit values for the axis. Motion can be continued by acknowledging the error.
- **MOTION_EMERGENCY_ABORT**: Controlled motion stop with maximum dynamic path values on the path. Active commands in the path interpolator are canceled; read-in of new commands is prevented and is only possible following error acknowledgement. Active commands (IPO) are fed back to the user program with an error code.
- **DISABLE_MOTION**: Motion stop on path axes and positioning axis for path-synchronous motion. The path motion component is realized by means of a stop with the maximum dynamic values of the axis followed by cancellation of the path motion component. Thus, the path group is ungrouped. Active commands in the path interpolator are canceled; read-in of new commands is prevented and is only possible following error acknowledgement.
A.1 Specific kinematics with TrafoID 1001

Type of kinematics

The kinematics can be used with the TrafoID 1001. However, it can only be selected via the expert list by entering the corresponding ID there.

Kinematics 1001 are 2D kinematics (X-Y)

The following types of axes can be used:

- Axis A1 (X axis): Rotary or linear axis without modulo function
- Axis A2 (Y axis): Rotary axis without modulo function

Display of kinematics

![Kinematics display](image)

Figure A-1 Kinematics display

Settings on path object

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematics.typeOfKinematics = SPECIFIC (6)</td>
<td>TCP (end point)</td>
</tr>
<tr>
<td>Kinematics.specTrafoId = 1001</td>
<td>Gear ratio (e.g. range / 360°)</td>
</tr>
</tbody>
</table>
## A.1 Specific kinematics with TrafoID 1001

### Table A-1  Kinematics parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Use</th>
<th>Unit</th>
<th>Type</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinematics.parameter[1]</td>
<td>basicOffsetX</td>
<td>e.g. mm</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[2]</td>
<td>basicOffsetY</td>
<td>e.g. mm</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[3]</td>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kinematics.parameter[4]</td>
<td>ratioA1</td>
<td>e.g. mm/°</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[5]</td>
<td>length1</td>
<td>e.g. mm</td>
<td>LREAL</td>
<td>0</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[6]</td>
<td>length2</td>
<td>e.g. mm</td>
<td>LREAL</td>
<td>0</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[7]</td>
<td>distanceE1</td>
<td>e.g. mm</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[8]</td>
<td>verticalPosition2</td>
<td>0: lower, ≠0: upper</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
<tr>
<td>Kinematics.parameter[9]</td>
<td>offsetA2</td>
<td>e.g.</td>
<td>LREAL</td>
<td>-1E+12</td>
<td>1E+12</td>
</tr>
</tbody>
</table>

### Kinematics reference

The position of the TCP/ end point depends on the orientation (see **Traversing range** section) of rod 2. If the **lower** orientation is selected (as in the kinematics reference diagram), the position of the TCP/ EP can be determined as follows if the kinematics are in a neutral position:

\[
X_{(EP\_bottom)} = 0.0 - \text{basicOffsetX} \\
Y_{(EP\_bottom)} = -\sqrt{(\text{length2})^2 - (\text{length1-distanceE1})^2 - \text{basicOffsetY}}
\]

If the **upper** orientation is selected:

\[
X_{(EP\_top)} = 0.0 - \text{basicOffsetX} \\
Y_{(EP\_top)} = \sqrt{(\text{length2})^2 - (\text{length1-distanceE1})^2 - \text{basicOffsetY}}
\]
Figure A-2 Kinematics reference
Appendix A

A.1 Specific kinematics with TrafoID 1001

Transformation in X direction
Transformation in X direction can be used for linear and rotary axes (belt drive, spindle etc.).

BCS.x.position=(axis position A1 * parameter[4]) - parameter[1]

BCS.x.position=(axis position A1 * ratioA1) - basicOffsetX

Figure A-3 Schematic display - kinematics 1001

The kinematics zero point is A1=0, A2=0 and, when offset horizontally by E1, is at the height of A2.
Transformation in Y direction

Within the traversing range, transformation (MCS -> BCS) of the path object provides valid Cartesian values. If the axis is outside this range, 0.0 is output.

![Diagram showing the traversing range](image)

**Figure A-4** physical traversing range

Traversing range

The traversing range is determined by the orientation (position) of the sliding range. This can be determined using Kinematics.parameter[8].
Appendix A

A.1 Specific kinematics with TrafoID 1001

Upper orientation

Kinematics.parameter[8] ≠ 0.0 (e.g. 1.0)

Figure A-5  Traversing range with orientation selected as upper
Lower orientation

Kinematics.parameter[8] = 0.0

Figure A-6 Traversing range with orientation selected as lower
Index

Path object
Limits, 126

2
2D articulated arm, 84
2D delta picker, 74
2D roller picker, 67
2D swivel arm, 90
2D user function, 98
2D/3D gantry, 66

3
3D articulated arm, 86
3D cylindrical robot, 92
3D delta picker, 78
3D roller picker, 72
3D user function, 98

A
Axes
Creating with path interpolation, 116

B
Basic coordinate system, 15
BCS, 15
Blending
Path-synchronous motion, 54
with dynamic adaptation, 44
without dynamic adaptation, 45

C
Calibration, 132
CAM_EXT, 20
Cartesian 2D/3D gantry, 66
Cartesian axes, 16
Cartesian coordinate system, 21
Cartesian zero point, 56
Circular path, 16
Configuration
Path object, 123
Continuous-path control, 15
Conveyor, 15, 100
Conveyor-tracking, 100

D
Default, 143
Path object, 119

E
Error path axis, 52
Example
Path interpolation (overview), 135

F
Frame transformation, 16

I
Interconnections
Path object, 127
Interpolation types
Path interpolation, 24

K
Kinematic adaptation, 16, 56
Transformation, 56
Kinematic end point, 56
Kinematic offset, 61
Kinematic zero point, 56
Shift, 61
Kinematics, 16
2D articulated arm, 84
2D delta picker, 74
2D roller picker, 67
2D swivel arm, 90
2D user function, 98
2D/3D gantry, 66
3D articulated arm, 86
3D cylindrical robot, 92
3D delta picker, 78
3D roller picker, 72
Index

3D user function, 98
Conversion, 60
Overview, 62
SCARA, 80
User function, 98
Kinematics transformation, 16, 56

L
Limits
  Path object, 126
Linear path, 16
Link constellations, 59
Local alarm response
  Path interpolation, 176

M
Main plane, 21
Modulo properties, 22
Monitoring, 50
Motion end
  Blending with dynamic adaptation, 44
  Blending without dynamic adaptation, 45
  Overview, 43
  Stopping, 44
Motion sequence, 15, 100
Motion sequence reference value, 16, 101
MotionOut.x/y/z-Interface, 113

O
Object coordinate system, 100
Object coordinate system (OCS), 17
Objects
  Path interpolation, 19
OCS, 100
  Application example, 107
  Coupled, 17
  Path commands, 171
  Stopping, 107
OCS reference position, 17

P
PATH, 20
Path axes
  Creating, 138
Path axis, 13
  Error, 52
  Role, 20
Path axis offset, 60
Path commands
  Calculating the path length, 169
  Circular path, 27
  Circular path via center, angle, 30
  Circular path via intermediate point, end point, 31
  Circular path via radius, end point, orientation, 28
  Command tracking, 170
  Conversion commands, 170
  Geometric path analysis, 169
  Linear path, 26
  Motion, 170
  Object and Alarm Handling, 171
  Object coordinate system, 171
  Override behavior, 173
  Polynomial path, 32
  Polynomial path via attach continuously, 36
  Polynomial path via direct specification, 34
  Polynomial path via specification of start point data, 34
Path control panel, 15
Path dynamics, 38
  Limits, 39
  Specification via cam, 39
  Specification via command parameters, 38
Path interface, 16
Path interpolation, 13, 15
  Functionality, 19
  Local alarm response, 176
  Objects, 19
  Sample project, 135
Path interpolation grouping, 15
Path interpolation paths, 25
Path motion, 15
  Continue, 42
  Display, 50
  Monitoring, 50
  Stop, 42
Path object, 13, 15
  Calibration, 132
  Configuration, 123
  Creating, 140
  Default, 119
  Dynamic response, 119
  Interconnections, 127
  Object coordinate systems, 100, 119
  Parameter Assignment, 119
  System variables, 57
Path-axis interface, 15, 113
Path-synchronous motion, 17
  Blending, 54
TO Path Interpolation
Function Manual, 04/2014
Dynamic response, 54
Functionality, 53
Output coordinates, 55
Output path length, 55
Specification, 53
Pitch, 16
Polynomial path, 17
Position axis
  For path-synchronous motion, 13
Positioning axis interface, 113

R
Reference points, 56
References, 3
Right-handed system, 21
Roll, 16
Rotation, 16

S
Sample project
  Create axes., 138
  Creating a path object, 140
  Defining the kinematics, 141
  Programming path segments, 145
  Setting the default, 143
  Technology package, 137
SCARA, 80
Selecting a technology package, 137
Simulation operation, 114
System variables, 57

T
TO pathObject, 13
TrackingIn interface, 17
trackingInPosition, 16
Translation, 16
Traversing range limits
  of path and positioning axes, 51

U
Units
  Path interpolation, 23
User function, 98

W
Work offset, 60

Y
Yaw, 16