



Siemens  
Industry  
Online  
Support

LIBRARY

# Library Kinematics Control

SIMATIC / Robot Control Suite / LKinCtrl / V5.2

**SIEMENS**

# Legal information

## Use of application examples

Application examples illustrate the solution of automation tasks through an interaction of several components in the form of text, graphics and/or software modules. The application examples are a free service by Siemens AG and/or a subsidiary of Siemens AG ("Siemens"). They are non-binding and make no claim to completeness or functionality regarding configuration and equipment. The application examples merely offer help with typical tasks; they do not constitute customer-specific solutions. You yourself are responsible for the proper and safe operation of the products in accordance with applicable regulations and must also check the function of the respective application example and customize it for your system.

Siemens grants you the non-exclusive, non-sublicensable and non-transferable right to have the application examples used by technically trained personnel. Any change to the application examples is your responsibility. Sharing the application examples with third parties or copying the application examples or excerpts thereof is permitted only in combination with your own products. The application examples are not required to undergo the customary tests and quality inspections of a chargeable product; they may have functional and performance defects as well as errors. It is your responsibility to use them in such a manner that any malfunctions that may occur do not result in property damage or injury to persons.

## Disclaimer of liability

Siemens shall not assume any liability, for any legal reason whatsoever, including, without limitation, liability for the usability, availability, completeness and freedom from defects of the application examples as well as for related information, configuration and performance data and any damage caused thereby. This shall not apply in cases of mandatory liability, for example under the German Product Liability Act, or in cases of intent, gross negligence, or culpable loss of life, bodily injury or damage to health, non-compliance with a guarantee, fraudulent non-disclosure of a defect, or culpable breach of material contractual obligations. Claims for damages arising from a breach of material contractual obligations shall however be limited to the foreseeable damage typical of the type of agreement, unless liability arises from intent or gross negligence or is based on loss of life, bodily injury or damage to health. The foregoing provisions do not imply any change in the burden of proof to your detriment. You shall indemnify Siemens against existing or future claims of third parties in this connection except where Siemens is mandatorily liable.

By using the application examples you acknowledge that Siemens cannot be held liable for any damage beyond the liability provisions described.

## Other information

Siemens reserves the right to make changes to the application examples at any time without notice. In case of discrepancies between the suggestions in the application examples and other Siemens publications such as catalogs, the content of the other documentation shall have precedence.

The Siemens terms of use (<https://support.industry.siemens.com>) shall also apply.

## Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, systems, machines and networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions constitute one element of such a concept.

Customers are responsible for preventing unauthorized access to their plants, systems, machines and networks. Such systems, machines and components should only be connected to an enterprise network or the internet if and to the extent such a connection is necessary and only when appropriate security measures (e.g. firewalls and/or network segmentation) are in place.

For additional information on industrial security measures that may be implemented, please visit <https://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends that product updates are applied as soon as they are available and that the latest product versions are used. Use of product versions that are no longer supported, and failure to apply the latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under <https://www.siemens.com/cert>.

# Table of contents

<b>1.</b>	<b>Introduction .....</b>	<b>6</b>
1.1.	Overview .....	6
1.2.	Mode of operation.....	7
1.2.1.	Objective .....	7
1.1.1	Architecture and concept.....	8
1.3.	Components used .....	8
<b>2.</b>	<b>Engineering .....</b>	<b>9</b>
2.1.	Project integration.....	9
2.1.1.	Requirements.....	9
2.1.2.	Import of application .....	9
2.2.	FB LKinCtrl_MC_MovePath (FB 35000).....	10
2.3.	PathData .....	15
2.3.1.	Structure.....	15
2.3.2.	Configuration .....	20
2.4.	Call LKinCtrl_MC_MovePath .....	22
2.5.	Operation .....	23
2.5.1.	General functionalities.....	23
2.5.2.	Operating modes.....	24
<b>3.</b>	<b>Programming &amp; Supported commands .....</b>	<b>26</b>
3.1.	End of path .....	27
3.2.	Point reference.....	27
3.3.	Linear commands.....	28
3.4.	Circular commands.....	28
3.4.1.	<i>CircMode</i> = 1: AuxPoint + Arc.....	29
3.4.2.	<i>CircMode</i> = 2: Endpoint + Radius .....	31
3.4.3.	<i>CircMode</i> = 0: Endpoint + AuxPoint (Circle in 3D space).....	32
3.5.	MoveDirect / synchronous Point-to-Point (sPTP) .....	33
3.6.	Modal dynamics .....	36
3.6.1.	Set path dynamics .....	36
3.6.2.	Set orientation dynamics .....	36
3.6.3.	Set sPTP dynamics .....	36
3.7.	Conveyor Tracking.....	36
3.7.1.	Track conveyor belt .....	36
3.7.2.	Desynchronize conveyor .....	41
3.8.	Frame commands.....	41
3.8.1.	OCS frame.....	41

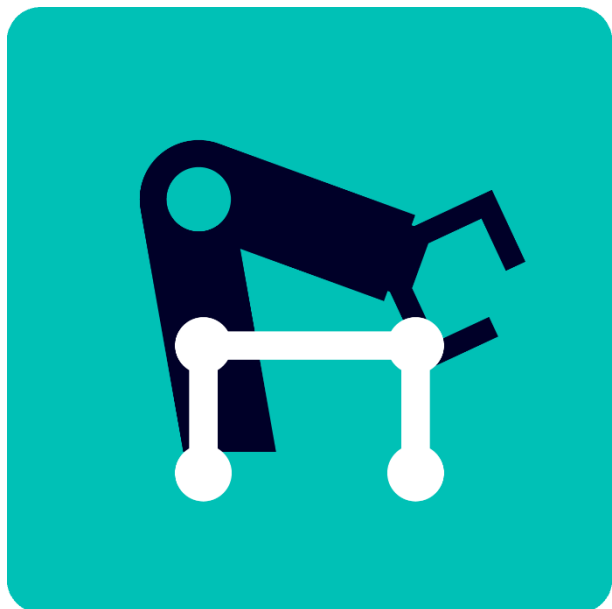
3.8.2.	UCS frame (User Coordinate System) .....	41
3.8.3.	Define Tool .....	42
3.8.4.	Set Tool .....	42
3.9.	Zone commands.....	42
3.9.1.	Define workspace zone.....	42
3.9.2.	Activate workspace zone.....	42
3.9.3.	Deactivate workspace zone .....	43
3.9.4.	Define kinematics zone.....	43
3.9.5.	Activate kinematics zone .....	43
3.9.6.	Deactivate kinematics zone.....	43
3.10.	Flags.....	45
3.10.1.	Principle of operation .....	45
3.10.2.	Flag modes .....	47
3.11.	FlagOnly commands.....	58
3.12.	Wait times.....	59
3.13.	Contour offset and radius compensation .....	59
3.14.	Relative shift of object coordinate systems.....	62
3.15.	Pick and place sequence .....	64
3.16.	Measurement command.....	66
<b>4.</b>	<b>Standalone functions .....</b>	<b>69</b>
4.1.	FB LKinCtrl_MC_GroupPower (FB 35021) .....	69
4.2.	FB LKinCtrl_MC_GroupReset (FB 35023) .....	70
4.3.	FB LKinCtrl_MC_GroupHome (FB 35022) .....	71
4.4.	FB LKinCtrl_MC_JogFrame (FB 35010) .....	72
4.5.	FB LKinCtrl_MC_MovePickAndPlaceLinear (FB 35026) .....	75
<b>5.</b>	<b>Additional information .....</b>	<b>91</b>
5.1.	Trigger multiple commands per call.....	91
5.2.	Continue at start.....	91
<b>6.</b>	<b>Error handling .....</b>	<b>92</b>
6.1.	LKinCtrl_MC_MovePath.....	92
6.1.1.	LKinCtrl_MC_ExecuteKinMotionCmd .....	93
6.1.2.	LKinCtrl_PreBuffer .....	94
6.1.3.	LKinCtrl_OffsetContour.....	96
6.2.	LKinCtrl_MC_MovePickAndPlaceLinear.....	97
6.3.	LKinCtrl_MC_JogFrame .....	98
6.4.	Auxiliary blocks .....	98
6.4.1.	LKinCtrl_MC_GroupPower .....	99
6.4.2.	LKinCtrl_MC_GroupReset .....	99

- 6.4.3. LKinCtrl\_MC\_GroupHome ..... 99
- 7. PLC Tags.....100**
  - 7.1. LKinCtrl\_Configuration..... 100
  - 7.2. LKinCtrl\_Constants ..... 101
  - 7.3. MC\_Constants ..... 103
- 8. Appendix .....106**
  - 8.1. Service and support ..... 106
  - 8.2. Application support ..... 107
  - 8.3. Links and literature ..... 107
  - 8.4. Change documentation ..... 107

# 1. Introduction

## 1.1. Overview

The library LKinCtrl provides functionalities for TO Kinematics to easily program and control granular path motions in a command list. Additionally, it offers HMI screens to commission and diagnose the TO Kinematics and its axes. The HMI part is described in a separate documentation.



### HMI manual

After the integration in the PLC, read the HMI manual to integrate the desired HMI modules.

[SIMATIC S7-1500T Kinematics Control](#)



### Getting started example project and documentation

For a quick and easy start use the Getting started project and documentation.

[SIMATIC S7-1500T Kinematics Control](#)

### MC\_MovePath

The Library Kinematics Control enables the user to easily control a kinematics to run predefined path motions using the technology object Kinematics.

Instead of executing and handling several single motion commands in a user program only one core function block controls the TO Kinematics. Path information is being provided by a list of commands. Path motions can be stopped, interrupted and continued during execution just by inputs. Detailed information on the path execution status as well as error diagnostics information is provided.

Following benefits are provided with this library:

- Comfortable path definition in a command list
- Execution of motion commands according to the command list
- Single step / automatic mode for path execution
- Diagnostic interface

- Flags for actuator control depending on path status
- Flags for sequence control (wait conditions)
- Wait times in command sequence
- Activation / deactivation of single path commands
- Compensation of tool and radius length
- Complete set of configuration functions programmable in command list
- Auxiliary functions to enable / home / reset all axes of the axes group
- Familiar interface behavior of motion commands
- Memory and runtime optimized function blocks
- Comfortable point definition in a point table
- Encapsulated command sequence for pick and place applications with integrated conveyor tracking
- Measurement command functionality to abort single commands within a path

### MC\_JogFrame

The function block MC\_JogFrame provides all the functionality to jog a kinematics in Cartesian directions X, Y, Z including the orientation axis continuously, incrementally and to a specified target position.

### MC\_MovePickAndPlaceLinear

The function block LKinCtrl\_MC\_MovePickAndPlaceLinear encapsulates a linear MC command sequence for typical Pick & Place applications. It is usable like a standard MC block and eases programming for static motions as well as conveyor tracking applications. Tracking functionality is integrated, which means that no additional command is necessary to establish tracking.

#### NOTE

The application can run on all 1500T CPU types with FW V3.0.0 or greater. The Unified HMI requires FW V19.0.0.2 or greater.

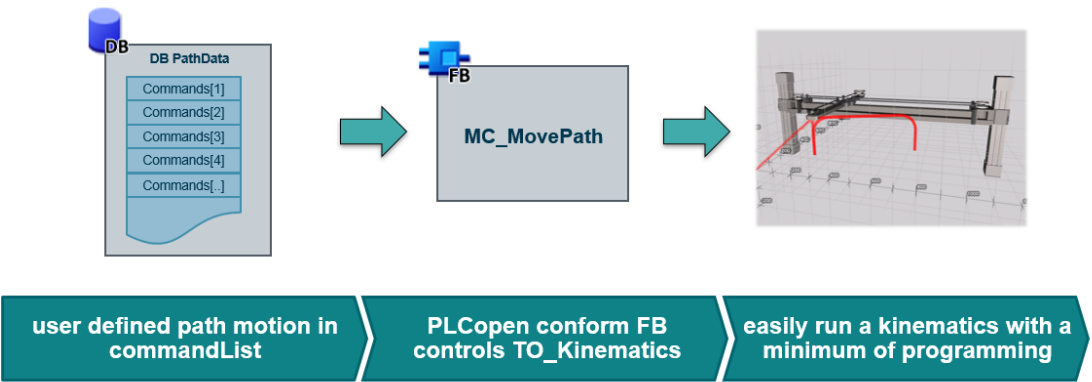
## 1.2. Mode of operation

### 1.2.1. Objective

The standard application LKinCtrl enables the user to easily run predefined path motions using the technology object Kinematics.

Instead of executing and handling several single motion commands in a user program the TO Kinematics is controlled by only one core function block executing a list of commands. [Figure 1-1](#) shows the schematic application workflow.

Figure 1-1 General application workflow



1.1.1      **Architecture and concept**

The path motion is defined by parametrizing a command list, called PathData. The parametrization of the PathData follows the parametrization of the system functions and includes the available system motion commands for TO Kinematics. The PathData length can be adapted to the use case by internal library user constant.

The PathData serves as an InOut parameter for the application’s core FB *LKinCtrl\_MC\_MovePath*. This function block controls the TO Kinematics and internally executes the motion commands as defined in the PathData. Furthermore, the function block can interrupt, continue and stop the path motion and offers detailed diagnostics information in its interface.

The MC\_MovePath also manages to steadily keep the TO’s job sequence full but does not overload it. Therefore, a maximum look ahead in terms of dynamic planning is achieved.

1.3.      **Components used**

This application example has been created with the following hard- and software components:

Component	Number	Article Number	Note
CPU 1518T-2 PN	1	6ES7 518-4UP00-0AB0	Or other S7-1500T CPU with FW 3.0.0

You can purchase these components from the [Siemens Industry Mall](#).

This application example consists of the following components:

Component	File name	Note
LKinCtrl library	LKinCtrl_V5_2_0_TiaLib_V19.zip	
LKinCtrl manual	LKinCtrl_V5_2_0_Manual_en.pdf	
LKinCtrl manual HMI	LKinCtrl_V5_2_0_Manual_HMI_en.pdf	
LKinCtrl getting started	LKinCtrl_V5_2_0_GettingStarted_TiaPrj_V19.zip	
LKinCtrl manual getting started	LKinCtrl_V5_2_0_GettingStarted_Manual_en.pdf	
LKinCtrl changelog	LKinCtrl_V5_2_0_Changelog_en.pdf	



## 2. Engineering

### 2.1. Project integration

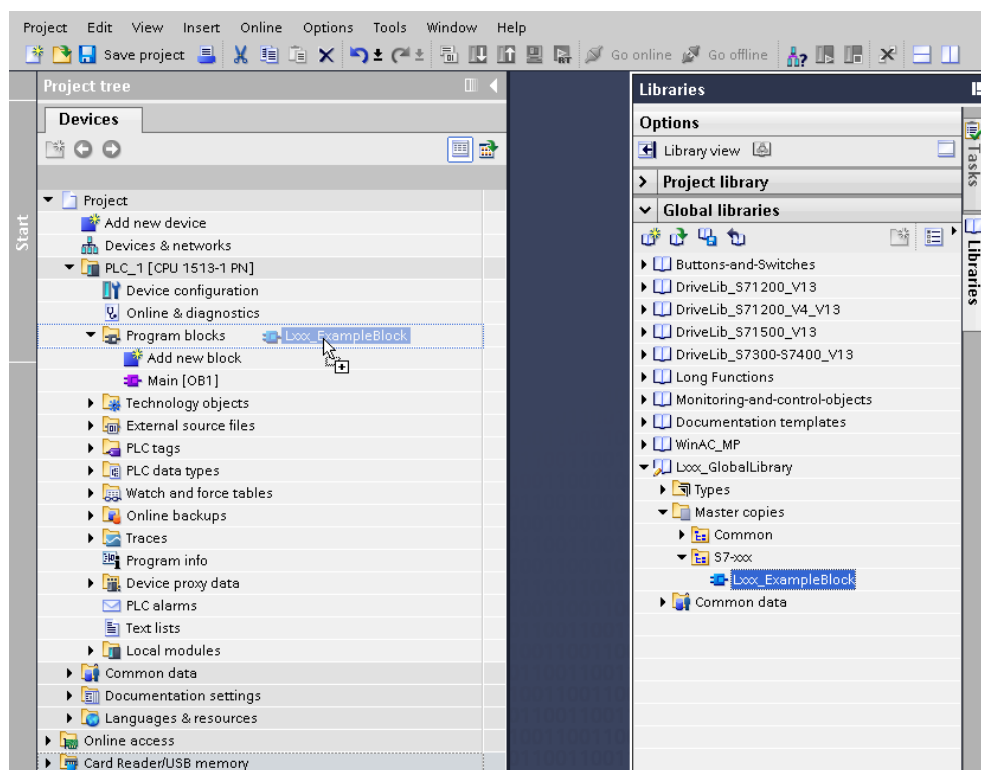
#### 2.1.1. Requirements

To run the application LKinCtrl a basic TIA portal V19 project with TO Kinematics and the corresponding number of TO PositioningAxis needs to be set up. The TO Kinematics must be configured and a basic commissioning to run the kinematics with the control panel should be completed. This can also be done with the commissioning module in the HMI.

#### 2.1.2. Import of application

The table below lists the steps for integrating the blocks of the LKinCtrl library into your STEP 7 program.

Figure 2-1: Integrating the library blocks into TIA Portal



**NOTE** Please follow the sequence of inserting the folders.

Table 2-1: Integrating the library blocks into STEP 7

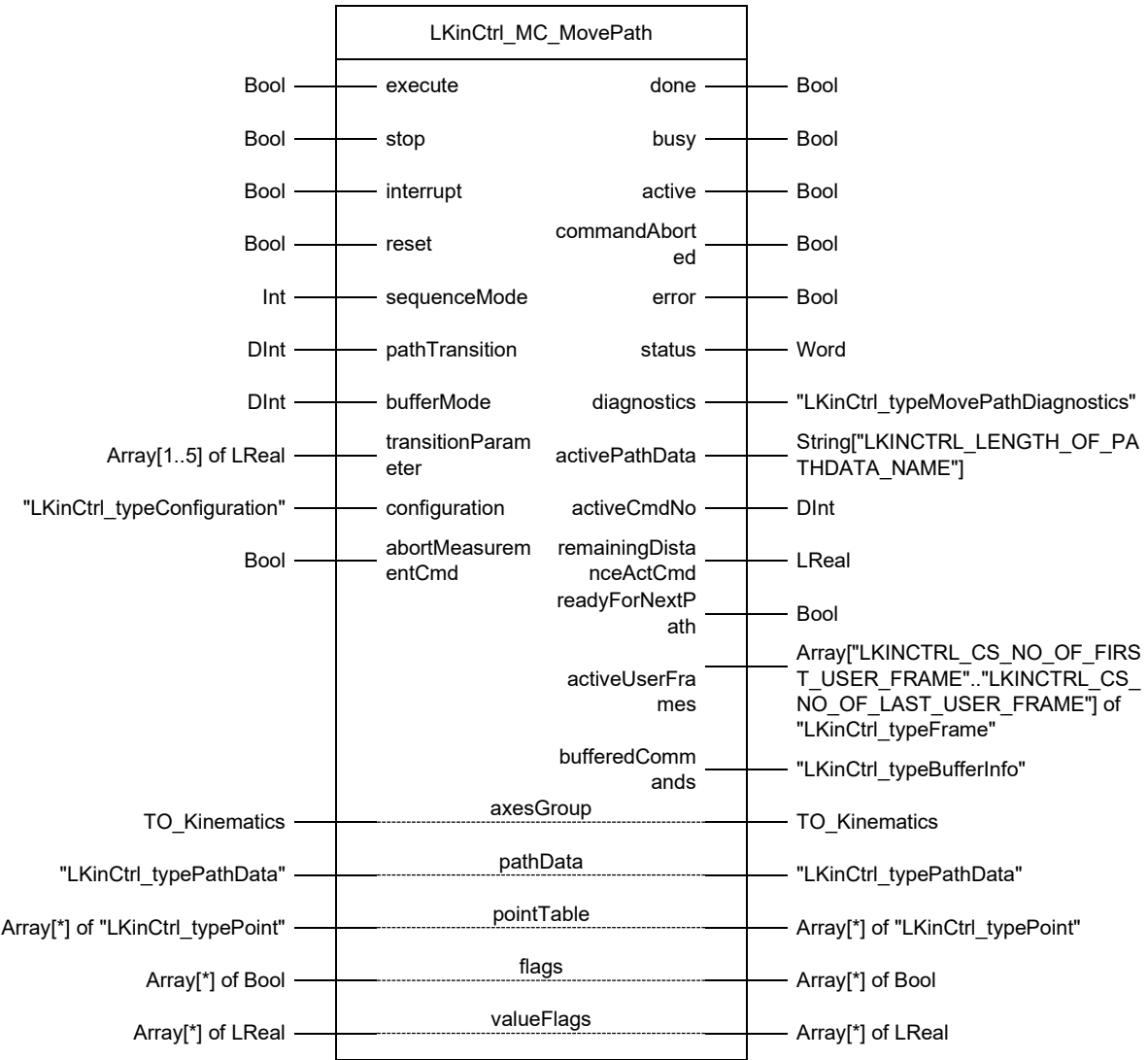
No.	Action
1	Copy the folder <i>LKinCtrl_Tags</i> with Drag & Drop into the "PLC tags" in the PLC.
2	Copy the folder <i>LKinCtrl_Types</i> with Drag & Drop into the "PLC data types" in the PLC.
3	Copy the folder <i>LKinCtrl_Blocks</i> with Drag & Drop into the "Program blocks" in the PLC.
4	(optional) Copy the folder <i>LKinCtrl_Data</i> with Drag & Drop into the "Program blocks" in the PLC. Existing data or separately created data can also be used for the interface of the library blocks.
5	Now the blocks can be configured and called in the user program.

## 2.2. FB LKinCtrl\_MC\_MovePath (FB 35000)

The interfaces and the controls of the function blocks in the LKinCtrl library refer to the PLCopen standard.

The function blocks are implemented in Structured Control Language (SCL). They are programmed for use in a cyclic task.

Figure 2-2: LKinCtrl\_MC\_MovePath



### Interface parameters

Table 2-2: Parameter of LKinCtrl\_MC\_MovePath

Name	P-Type	Data Type	Comment
execute	IN	Bool	rising edge starts action once
stop	IN	Bool	abort path motion
interrupt	IN	Bool	interrupt path motion
reset	IN	Bool	reset FB / acknowledge errors
sequenceMode	IN	Int	activate sequence mode (0) / single step mode (1)
pathTransition	IN	DInt	initial transition configuration for new path motion (1: use bufferMode and transitionParameter)
bufferMode	IN	DInt	initial bufferMode for new path motion

Name	P-Type	Data Type	Comment
transitionParameter	IN	Array[1..5] of LReal	initial transitionParameter for new path motion
configuration	IN	"LKinCtrl_typeConfiguration"	configuration structure
abortMeasurementCmd	IN	Bool	TRUE: abort active command if measurement command
done	OUT	Bool	TRUE: Commanded functionality has been completed successfully
busy	OUT	Bool	TRUE: FB is not finished and new output values can be expected
active	OUT	Bool	FB in control of axesGroup / kinematic in motion
commandAborted	OUT	Bool	TRUE: Commanded functionality has been aborted by another command
error	OUT	Bool	TRUE: An error occurred during the execution of the FB
status	OUT	Word	status of the FB
diagnostics	OUT	"LKinCtrl_typeMovePathDiagnostics"	diagnostics information of FB
activePathData	OUT	String ["LKINCTRL_LENGTH_OF_PATHDATA_NAME"]	name of actual PathData
activeCmdNo	OUT	DInt	number of actual PathData set
remainingDistanceActCmd	OUT	LReal	remaining distance of actual PathData set
readyForNextPath	OUT	Bool	TRUE: Next PathData can be triggered
activeUserFrames	OUT	Array ["LKINCTRL_CS_NO_OF_FIRST_USER_FRAME" .. "LKINCTRL_CS_NO_OF_LAST_USER_FRAME"] of "LKinCtrl_typeFrame"	currently active user frames
bufferedCommands	OUT	"LKinCtrl_typeBufferInfo"	current buffered commands including history.
axesGroup	IN_OUT	TO_Kinematics	reference to the axesGroup
pathData	IN_OUT	Variant	reference to the PathData
pointTable	IN_OUT	Array[*] of "LKinCtrl_typePoint"	reference to point table
flags	IN_OUT	Array [0.."LKINCTRL_NO_OF_LAST_SETFLAG"] of Bool	boolean defined by setFlags in PathCommand
valueFlags	IN_OUT	Array [0.."LKINCTRL_NO_OF_LAST_VALUEFLAG"] of LReal	value defined by valueFlags in PathCommand

### Status constants

Table 2-3: status constants of LKinCtrl\_MC\_MovePath

Name	Type	Value	Comment
STATUS_EXECUTION_FINISHED	Word	16#0000	status execution finished without errors
STATUS_NO_CALL	Word	16#7000	status no job being currently processed
STATUS_FIRST_CALL	Word	16#7001	status first call after incoming new job (rising edge 'execute')

Name	Type	Value	Comment
STATUS_SUBSEQUENT_CALL	Word	16#7002	status subsequent call during active processing without further details
STATUS_FB_RESETTING	Word	16#7010	status FB resetting
STATUS_WAITING_FOR_PREBUFFERED_CMD	Word	16#7070	status waiting for new command in PathData_Prebuffered
STATUS_PREBUFFERED_FULL	Word	16#7071	status internal cmd buffer filled completely
STATUS_CMD_LIN_ACTIVE	Word	16#7301	status linear motion command active at axesgroup
STATUS_CMD_CIRC_ACTIVE	Word	16#7302	status circular motion command active at axesgroup
STATUS_PATHMOTION_INTERRUPTED	Word	16#7303	status path motion interrupted
STATUS_PATHMOTION_STOPPING	Word	16#7304	status path motion stopping
STATUS_READYFORNEXTPATH	Word	16#7305	status FB ready for next path execution
STATUS_PATHMOTION_RUNNING	Word	16#7306	status path motion running
STATUS_EXECUTION_WAITING	Word	16#7307	status path execution waiting due to wait flag, wait time or single step mode
STATUS_CMD_MOVE_DIRECT_ACTIVE	Word	16#7308	status move direct motion command active at axesgroup
STATUS_STOP_DUE_TO_ERROR	Word	16#7309	status kinematics is stopped due to an error in FB Executer
STATUS_CMD_CONFIGURATION_ACTIVE	Word	16#730A	status configuration command active at axesgroup
STATUS_COMMAND_ABORTED	Word	16#7FFF	status commanded functionality has been aborted by another command

**NOTE**

Due to the internal structure, only optimized DBs can be used. When using non optimized DBs the CPU goes to STOP mode.

## Principle of operation

The LKinCtrl\_MC\_MovePath function block organizes and operates the complete functionality of this application. In addition to its internal FBs (described in the following) it controls the TO Kinematic to run the user defined motion commands and return diagnostic information regarding the state of the path motion execution. Furthermore, it implements functionalities to control actuators depending on the path motion status.

## Supported general functionalities

- Start / continue path motion execution
- Stop / interrupt path motion execution
- Reset errors at the function block (not technology object errors)
- Selection of operating mode (automatic / sequential mode)
- Control of actuators via flags

## Supported diagnostics information

- Motion status information like PLCOpen
- Error and status information
  - Function block internal
  - Motion command error
  - Technology object error
- Path motion status
  - Active PathData (name)
  - Active PathData command (number)
  - Remaining distance active command
  - Buffered command status

### NOTE

Before executing a path motion with the FB LKinCtrl\_MC\_MovePath, the axes must be enabled and possibly reset and homed by the system basic motion control functions MC\_POWER, MC\_RESET and MC\_HOME.

## MovePath configuration structure

Name	Type	Comment
offsetParameter	Array[1.."LKINCTRL_NO_OF_OFFSETS"] of "LKinCtrl_typeContourOffsetParameter"	offset parameter settings for contour offset commands
conveyorParameter	Array[1.."LKINCTRL_NO_OF_CONVEYOR"] of "LKinCtrl_typeConveyorConfiguration"	conveyor parameter settings for conveyor tracking commands
userFrames	Array["LKINCTRL_CS_NO_OF_FIRST_USER_FRAME".. "LKINCTRL_CS_NO_OF_LAST_USER_FRAME"] of "LKinCtrl_typeFrame"	user frames configuration. Definition of additional frames besides OCS1-3. Use coordinate system 11...n at command
workspaceFrames	Array[1.."LKINCTRL_NO_OF_ZONES"] of "LKinCtrl_typeWorkspaceZoneDefinition"	Workspace zone configurations to be used for zone commands
kinematicsFrame	Array[1.."LKINCTRL_NO_OF_ZONES"] of "LKinCtrl_typeKinematicsZoneDefinition"	Kinematics zone configuration to be used for zone commands

Name	Type	Comment
stopMode	DInt	MC_GroupStop dynamics; 0: stop with dynamics of active motion job; 1: stop with maximum kinematics dynamics (dynamics adaption still effective); 10: stop with dynamics of active motion job and desync conveyor; 11: stop with maximum kinematics dynamics and desync conveyor
interruptMode	DInt	MC_GroupInterrupt dynamics; 0: interrupt with dynamics of active motion job; 1: interrupt with maximum kinematics dynamics (dynamics adaption still effective)
errorStopMode	DInt	MC_GroupStop dynamics when stopping due to error; 0: stop with dynamics of active motion job; 1: stop with maximum kinematics dynamics (dynamics adaption still effective); 10: stop with dynamics of active motion job and desync conveyor; 11: stop with maximum kinematics dynamics and desync conveyor
preloadMotionQueue	USInt	Preload motion queue via interrupt and continue before start of motion. 0: No preloading, 1: Preload and start when prepared 2: Preload and start with acknowledge (rising edge at execute)
resetModalDynamics	Bool	TRUE: internal modal dynamics are reset between paths. FALSE: internal modal dynamics are kept between paths.

## 2.3. PathData

### 2.3.1. Structure

The PathData structure represents the command list for the path definition. Provided as a library data type, the PathData is meant to be initialized in a global data block.

All available system commands for TO Kinematics are supported and listed in the *PathData* to define a path. The user defines the specific command type and parametrizes the motion command along the lines of the system commands within the structure.

Additionally, flags can be set to control actuators depending on the path motion status. These are defined individually for every command entry in the list.

**NOTE**

An overview of the supported commands can be found in [Programming & Supported commands](#)

Table 2-4: Parameter of LKinCtrl\_typePathData

Name	Type	Comment
pathDataName	String["LKINCTRL_LENGTH_OF_PATHDATA_NAME"]	Name of the PathData; Displayed as active path data at MovePath
LKinLangInterface	"LKinCtrl_typeAdvPointers"	Interface to LKinLang application; Contains pointers to handle pathData as ringbuffer
commands	Array[1.."LKINCTRL_NO_OF_PATHDATA_ELEMENTS"] of "LKinCtrl_typePathDataElement"	Array of commands to be executed; Details in chapter <a href="#">3</a>

## Content of a PathData element

Table 2-5: Parameter of LKinCtrl\_typePathDataElement

Name	Comment
cmdType	CmdType to be executed, overview can be found in <a href="#">Programming &amp; Supported commands</a>
cmdActivated	(De)Activation of command
cmdName	Dame of command
point	reference to point table; used for cmdCoordinates; > -1: use point; <= -1: no reference
cmdCoordinates	Contains target coordinate description
cartesianPosition	Target position OR target distance if relative command
x	Cartesian position X OR Position A1/J1 depending on coordinate system
y	Cartesian position Y OR Position A2/J2 depending on coordinate system
z	Cartesian position Z OR Position A3/J3 depending on coordinate system
a	Cartesian position A OR Position A4/J4 depending on coordinate system
b	Cartesian position B OR Position A5/J5 depending on coordinate system
c	Cartesian position C OR Position A6/J6 depending on coordinate system
coordSystem	0: WCS; 1: OCS[1]; 2: OCS[2]; 3: OCS[3]; 100: MCS, 101: JCS
cmdParameters	Contains generic parameters for all types of commands
pathDynamics	If value is <0 then default setting from TO is used
velocity	Velocity for command If MoveDirect, then velocity factor from 0.01 to 1.0 (1% to 100%) If WaitTime, then time in ms
acceleration	Acceleration for command If MoveDirect, then acceleration factor from 0.01 to 1.0 (1% to 100%)
deceleration	Deceleration for command If MoveDirect, then deceleration factor from 0.01 to 1.0 (1% to 100%)
jerk	Jerk for command If MoveDirect, then jerk factor from 0.1 to 0.9 (10% to 90%)
OrientationDynamics	If value is <0 then default setting from TO is used
velocity	Orientation velocity for command
acceleration	Orientation acceleration for command
deceleration	Orientation deceleration for command
jerk	Orientation jerk for command



Name	Comment
orientationDirection	Direction of orientation A for commands with absolute target 1: positive direction; 2: negative direction; 3: shortest distance
bufferMode	Blending setting between this and previous command 1: standstill between commands; 2: blending with lower speed; 5: blending with higher speed
transitionParameter[1] <sup>1</sup>	Blending distance parameter >0: Maximum distance
dynamicsAdaption	<0.0: default TO setting used; 0: no dynamic adaption; 1: adaption with segmentation of path; 2: adaption without segmentation of path
toolNumber	Tool number for SetTool command (LKINCTRL_MC_SETTOOL)
moveDirectParameters	Specific parameters for move direct commands only
linkConstellation	linkConstellation
positionMode	1: absolute cartesian position and absolute orientation 2: absolute cartesian position and relative orientation
turnJoint	Turn joint setting
offsetParameters	Parameters for contour offset commands
offsetParaNo	Selection of offset parameter set from configuration structure
mainPathPlane	Main path plane for compensation
conveyorParameters	Parameters for conveyor selection and configuration
conveyorParaNo	Selection of conveyor parameter set from configuration structure
circleParameters	Specific parameters for move circular commands only
circMode	0: auxPoint defines point on circle line; 1: auxPoint defines circle center; 2: radius and endPoint define circle segment; 3: GCode notation (end & aux point)
point	Reference to point table, used for AuxPoint >-1 use point; <=-1 no reference
auxPoint	Auxiliary point for circle definition [1] = X; [2] = Y; [3] = Z
pathChoice	circMode(0): not relevant; circMode(1): 0: pos direction; 1: neg direction; circMode(2): 0: short pos segment; 1: short neg segment; 2: long pos segment; 3: long neg segment

<sup>1</sup> TransitionParameter[2..5] are reserved but currently not used

Name	Comment
circlePlane	circMode(0): not relevant; circMode(1 or 2): 0: XZ; 1: YZ; 2: XY
radius	Only in combination with circMode(2): radius
arc	Only in combination with circMode(1): arc
zoneParameters	Specific parameters for zone commands
zoneConfigurationIndex	Index of zone configuration at MovePath
zoneNumber	Zone number selection
deactivationMode	Deactivation mode for zone deactivation commands
pickAndPlaceParameters	Parameters for pick and place command
startParameter	Start vector parameter
direction	Direction of vector (will be normed by FB) from start / target
workingHeight	Length of vector
transitionArea	Permissible blending area at end of vector
dynamicsFactor	Path dynamics scaling factor
targetParameter	Target vector parameter
Direction	Direction of vector (will be normed by FB) from start / target
workingHeight	Length of vector
transitionArea	Permissible blending area at end of vector
dynamicsFactor	Path dynamics scaling factor
setFlags	Array of flag configurations
flag	Flag number to be set
flagMode	Flag behavior (set, reset, wait, remainingDistance), see <a href="#">Flag modes</a>
remainingDistance	Distance if flag is set depending on remaining distance of current command
ValueFlags	Array of value flag configurations
Flag	Index of flag to be written
Value	Value to be written at given index
flagMode	Flag behavior (set, reset, wait, remainingDistance), see <a href="#">Flag modes</a>
remainingDistance	Distance if flag is set depending on remaining distance of current command
ParameterValid	Array to set cmdCoordinates.cartesianPosition valid; not used if point reference is configured

Name	Comment
[1]	TRUE: X is valid target, FALSE: X stays at last position
[2]	TRUE: Y is valid target, FALSE: Y stays at last position
[3]	TRUE: Z is valid target, FALSE: Z stays at last position
[4]	TRUE: A is valid target, FALSE: A stays at last position
[5]	TRUE: B is valid target, FALSE: B stays at last position
[6]	TRUE: C is valid target, FALSE: C stays at last position

### 2.3.2. Configuration

The PathData is provided as a UDT in the library and can be defined in a global data block by creating a variable of the type *LKinCtrl\_typePathData*.

**NOTE** By default, the length of the PathData is set to 10 commands. According to the use case the length can be adapted by setting the library constant *LKINCTRL\_NO\_OF\_PATHDATA\_ELEMENTS* to the necessary number of elements.

#### Path definition in global DB

Figure 2-3: PathData configuration in global DB

PathData_Contour			
	Name	Data type	Start value
1	▼ Static		
2	▼ contour	"LKinCtrl_typePath...	
3	pathDataName	String["LKINCTRL_L...	'contour'
4	▶ pointers	"LKinCtrl_typeAdvP...	
5	▼ commands	Array[1.."LKINCTRL_...	
6	▼ commands[1]	"LKinCtrl_typePath...	
7	cmdType	Int	1
8	cmdActivated	Bool	TRUE
9	cmdName	String["LKINCTRL_L...	'B_up'
10	▼ cmdCoordinates	"LKinCtrl_typePoint...	
11	▼ cartesianPosition	"LKinCtrl_typeCart...	
12	x	LReal	0.0
13	y	LReal	28.28
14	z	LReal	50.0
15	a	LReal	0.0
16	coordSystem	DInt	0
17	▶ cmdParameters	"LKinCtrl_typePoint...	
18	▼ setFlags	Array[1.."LKINCTRL_...	
19	▼ setFlags[1]	"LKinCtrl_typeSetFl...	
20	flag	Int	1
21	flagMode	USInt	1
22	remainingDistance	LReal	-1.0
23	▶ setFlags[2]	"LKinCtrl_typeSetFl...	
24	▶ setFlags[3]	"LKinCtrl_typeSetFl...	
25	▶ commands[2]	"LKinCtrl_typePath...	
26	▶ commands[3]	"LKinCtrl_typePath...	

Figure 2-3 shows the first command in a PathData structure configured in the DB view. The top-level structure of the PathData contains the name of the PathData defined by a string and an array of commands. The string's length can also be adapted by a library constant called *LKINCTRL\_LENGTH\_OF\_PATHDATA\_NAME*. Default setting is a string of 10 chars.

## Path definition in SCL

The path data can be written directly in an SCL network. These can be inserted prior to the FB call. The image below (as well as the example project) shows a comfortable possibility to define a path in SCL. The following figure shows an exemplary setting of a path definition using SCL networks.

Figure 2-4: PathData definition in SCL networks

Network 1: Define PathData_PnP - P1: above pick position	
Comment	
1	<code>IF #initialCall0B1 THEN</code>
2	<code>// initialize command number in first network</code>
3	<code>#i := 1;</code>
4	<code>// command name</code>
5	<code>"DemoPathData".pathData_PnP.commands[#i].cmdName := 'P1';</code>
6	
7	<code>// command type</code>
8	<code>"DemoPathData".pathData_PnP.commands[#i].cmdType := "LKINCTRL_MC_MOVELINEARABS";</code>
9	
10	<code>// command coordinates</code>
11	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.x := 250.0;</code>
12	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.y := 150.0;</code>
13	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.z := 250.0;</code>
14	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.a := 0.0;</code>
15	
16	<code>// command parameters</code>
17	<code>"DemoPathData".pathData_PnP.commands[#i].cmdParameters.pathDynamics.velocity := 300.0;</code>
18	<code>"DemoPathData".pathData_PnP.commands[#i].cmdParameters.bufferMode := 2;</code>
19	
20	<code>// increase command number for next network</code>
21	<code>#i := #i + 1;</code>
22	<code>END_IF;</code>
Network 2: Define PathData_PnP - P2: pick position	
Comment	
1	<code>IF #initialCall0B1 THEN</code>
2	<code>// command name</code>
3	<code>"DemoPathData".pathData_PnP.commands[#i].cmdName := 'P2';</code>
4	
5	<code>// command type</code>
6	<code>"DemoPathData".pathData_PnP.commands[#i].cmdType := "LKINCTRL_MC_MOVELINEARABS";</code>
7	
8	<code>// command coordinates</code>
9	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.x := 250.0;</code>
10	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.y := 150.0;</code>
11	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.z := 50.0;</code>
12	<code>"DemoPathData".pathData_PnP.commands[#i].cmdCoordinates.cartesianPosition.a := 0.0;</code>
13	
14	<code>// command parameters</code>
15	<code>"DemoPathData".pathData_PnP.commands[#i].cmdParameters.pathDynamics.velocity := 150.0;</code>
16	<code>"DemoPathData".pathData_PnP.commands[#i].cmdParameters.bufferMode := 2;</code>
17	
18	<code>// increase command number for next network</code>
19	<code>#i := #i + 1;</code>
20	<code>END_IF;</code>

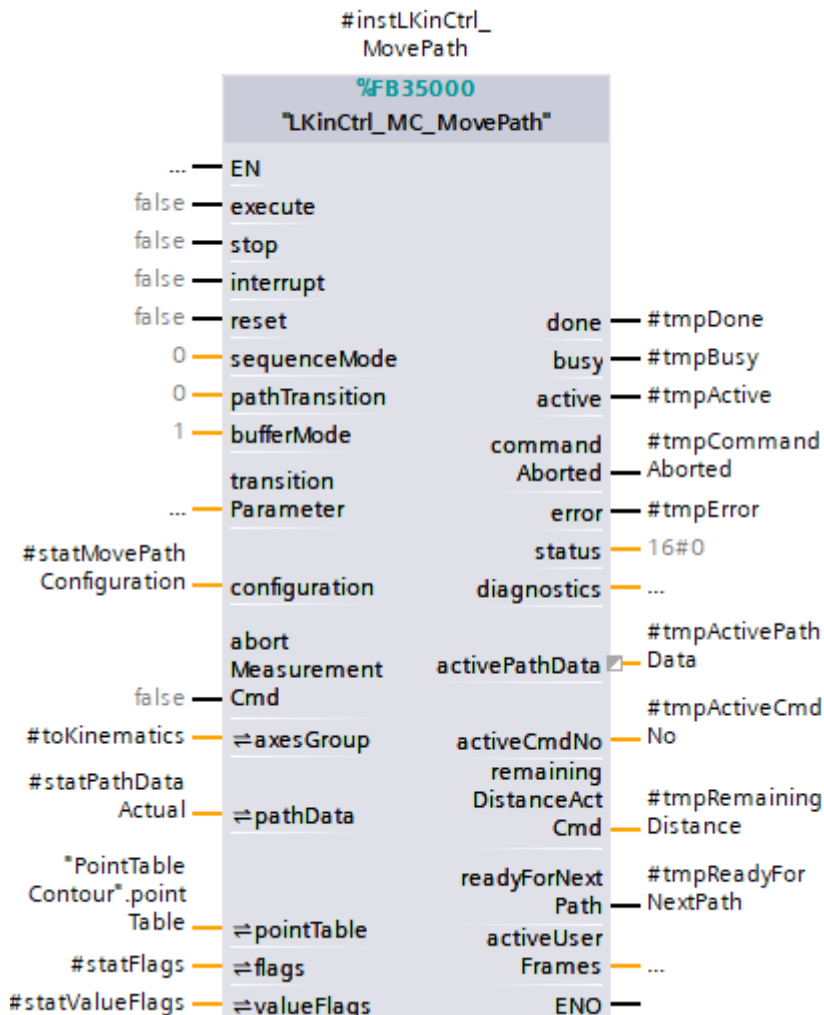
In the first network a command counter variable is initialized with value 1 as the first array index of the command list in the PathData. It is continuously increased after every command definition. Therefore, the user can easily apply changes to the command order by drag and drop of networks.

To minimize system load by defining a path motion, all definition network's code should only be executed once triggered by an event (e.g. upon PLC startup).

## 2.4. Call LKinCtrl\_MC\_MovePath

After creating a PathData structure within a DB the FB LKinCtrl\_MovePath must be called cyclically. An example for the block call with minimum signal connection is shown in the project and image below:

Figure 2-5: Call of LKinCtrl\_MC\_MovePath



It is necessary for the InOut parameters `axesGroup`, `pathData`, `pointTable` and `flags` to be connected to the corresponding objects.

The execute signal to start the path motion can either be set by connections to the input pin or in the code of an SCL block.



### CAUTION

#### Only trigger one instance at a time

When triggering multiple instances of the function block controlling the same TO Kinematics at the same time, both instances try to execute commands which results in a mixed order of the MotionQueue. This might lead to unexpected path motion of the kinematics.

Triggering multiple instances at the same time is not at all necessary, as one instance can handle multiple PathData structures subsequently in an optimized period of time.

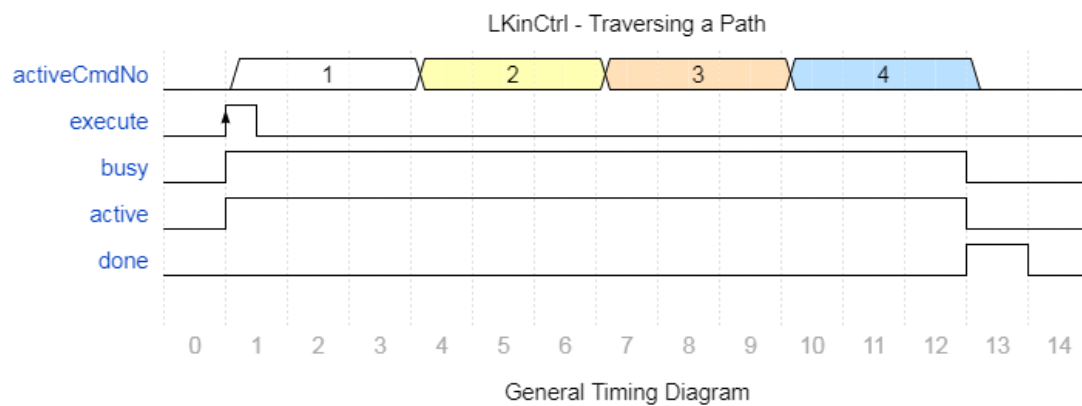
## 2.5. Operation

### 2.5.1. General functionalities

#### Interface controls

Based on the PLCOpen Part4 behavior, the execute input of the FB *LKinCtrl\_MC\_MovePath* is detecting rising edges to start. [Figure 2-6](#) presents the general behavior and timing of the FB.

Figure 2-6: General Timing Diagram of MC\_MovePath



The input *stop* internally triggers the system functions *MC\_GroupStop*. Therefore, a path motion is being aborted and all executed commands are deleted from the MotionQueue upon detection of a rising edge at *stop* input.

According to the function of a *MC\_GroupInterrupt*, a rising edge detected at input pin *interrupt* only interrupts the path motion. Actual path motion status and all executed commands stay in the MotionQueue and the motion can be continued.

After interruption, another rising edge at the *execute* input pin continues an interrupted path motion.

If a rising edge on *reset* is detected, error acknowledgment or a general reset of the function block is performed.

#### NOTE

Reset of the function block respectively acknowledging errors is only possible when *busy* = FALSE.

Using the input *sequenceMode* the function block can be executed in automatic or single step mode. Detailed information on operation modes follows in [Operating modes](#).

Using the input structure *configuration* the stop mode for the internal *MC\_GroupStop* can be specified (e.g. emergency stop). Parametrization is done accordingly to the system function *MC\_GroupStop*.

Also using the input structure *configuration*, the parameters for the offset compensation calculation can be specified for different offset parameters. Detailed information on the offset parameter parametrization can be found in [Contour offset and radius compensation](#).

#### Diagnostics & Status

The diagnostic output information separates in two sections. First section is the PLCOpen based outputs *done*, *busy*, *active*, *commandAborted*, *error* and *status*. Their timing behavior is based on the PLCOpen conform system motion commands. The output word *status* returns detailed status and error information of the function block in general.

In the first section further information is being provided to analyze path motion state and error information in the output structure diagnostics.

## Path motion status

While executing a path motion the outputs *activePathData* and *activeCmdNo* return information on the running command number and the corresponding PathData name. The output *remainingDistanceActCmd* returns the remaining distance of a single active motion command.

The Boolean *readyForNextPath* signalizes that all commands in a PathData have been internally executed to the MotionQueue. This does not mean that the commands have already been finished, but that they can be waiting in the MotionQueue. Upon change of the *readyForNextPath* bit to TRUE, the FB can be executed again with the next PathData. Therefore, the MotionQueue is kept filled and a maximum look ahead for dynamic planning is possible.

## Diagnostics structure

Besides the FB status information, the diagnostics structure contains detailed information that is written when an error occurs.

These include the concrete command number and the corresponding PathData name to identify the command possibly causing the error as well as state machine states of FBs *LKinCtrl\_MC\_MovePath* and *LKinCtrl\_ExecuteKinMotionCmd*. Furthermore, the diagnostics information provides information on the TO Kinematics by providing the status and error words in case of error.

## 2.5.2. Operating modes

### Automatic mode

To run the FB in automatic mode the input *sequenceMode* must be set to 0, which is also the default value.

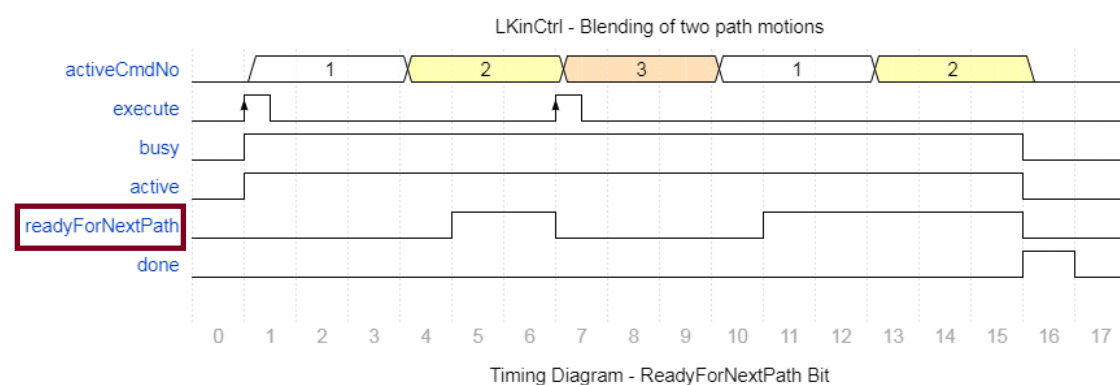
In automatic mode the FB executes the complete PathData and returns a done signal after finishing the last defined command in the PathData.

Retriggering of the FB is possible either after completion of a path motion (*done* = TRUE) or when the output *readyForNextPath* signalizes the execution of all PathData commands to the MotionQueue (explanation see below).

### Retriggering MovePath during kinematics driving executed PathData

The output bit *readyForNextPath* signalizes that all commands of a PathData transferred to the FB *LKinCtrl\_MC\_MovePath* have been registered in the MotionQueue. At that state it is possible to retrigger the FB again with a new PathData for a following path motion. That way, the MotionQueue remains filled, and the path continues immediately after the end of the first PathData. [Figure 2-7](#) presents the described situation in a timing diagram.

Figure 2-7: Timing diagram – retriggering following path during execution of actual path



### Blending one path motion into another

Possibly desired transitions of the two path motions can either be configured in the following PathData's first command entry or by using the FB inputs *pathTransition*, *bufferMode* and *transitionParameter[1..5]*. Setting the parameter *pathTransition* to 1 overwrites the subsequent PathData command's first entries with the blending parameters set at the FB interface.



Figure 2-8: Kinematics trace – no path blending

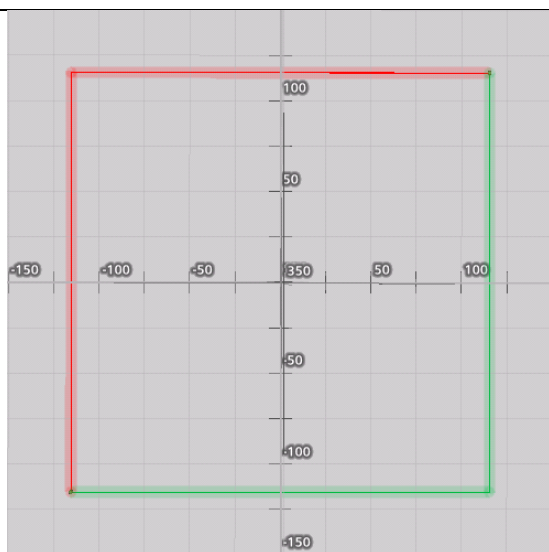
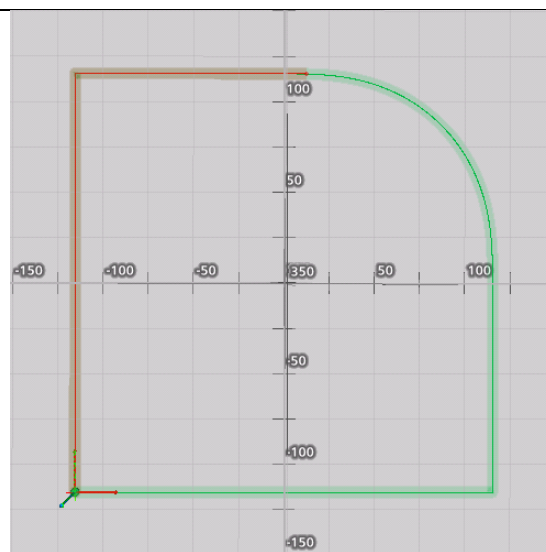


Figure 2-9: Kinematics trace – path blending



Two path motions without blending

Two path motions with blending

**NOTE**

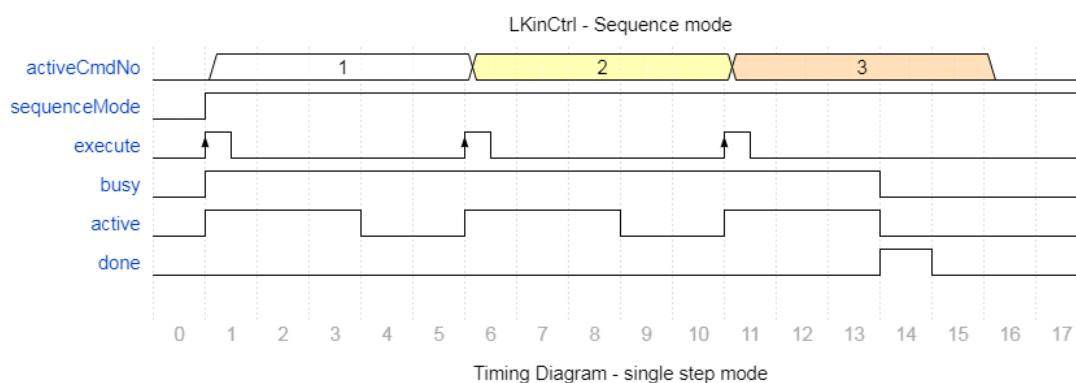
*pathTransition* = 1 enables the blending parameters *bufferMode* and *transitionParameter[1..5]* for the transition of two PathData structures.

**Single step**

To run the FB in single step mode the input *sequenceMode* must be set to 1.

In single step mode the FB executes each command separately. Every rising edge at the input *execute* triggers one command in the PathData. [Figure 2-10](#) shows the usage of the single step mode.

Figure 2-10: Timing diagram – single step mode usage

**CAUTION****bufferMode ignored**

When operating the function block in single step mode every single command in the PathData is executed separately. Therefore, no motion transitions have an effect and there is no blending between the commands.

The resulting path can differ from the path driven in automatic mode, due to the 'missing' blending segments. All programmed target positions will be reached completely.

### 3. Programming & Supported commands

The following chapters shall give an overview of the available commands. Only the special characteristics are explained for most command types. For more general information please refer to the Kinematics function manual linked in the chapter [Links and Literature](#).

cmdType	Command
-1	End of path
0	Flag only command
1	Absolute linear command
2	Relative linear command
3	Absolute circular command
4	Relative circular command
5	Absolute move direct command
6	Relative move direct command
10	Track conveyor belt command
11	Linear pick and place command
20	Set OCS frame command
21	Set UCS frame ( <u>U</u> ser <u>C</u> oordinate <u>S</u> ystem)
22	Define tool frame (tool 1) command
23	Set tool active command
30	Define workspace zone command
31	Set workspace zone active
32	Set workspace zone inactive
33	Define kinematics zone command
34	Set kinematics zone active
35	Set kinematics zone inactive
40	Disable contour offset
41	Activate contour offset on left side
42	Activate contour offset on right side
50	Desynchronize from conveyor belt
60	Relative shift of OCS
61	Reset relative shift of OCS
70	Set modal path dynamics

cmdType	Command
71	Set modal orientation dynamics
72	Set modal sPTP dynamics
100	Wait time

**CAUTION**

**Incorrect target position is possible when changing the OCS, Tool or active tool outside of the *LKinCtrl\_MC\_MovePath* while the FB is active.**

The *LKinCtrl\_MC\_MovePath* keeps track of the tool and frame definitions internally. If these are changed outside of the FB while the FB is active the internal state becomes invalid.

Only use *MC\_DefineTool*, *MC\_SetTool* and *MC\_SetOCSFrame* when the *LKinCtrl\_MC\_MovePath* is not busy. While the FB is busy use the LKinCtrl commands *Define Tool* [3.8.3](#), *Set Tool* [3.8.4](#) and *OCS frame* [3.8.1](#) as part of the path data. This way the internal state is consistent and remains valid.

### 3.1. End of path

The end of a path is signaled with configuring the *cmdType* = -1. No commands after this entry are executed.

### 3.2. Point reference

With version 4.0 the PathData was updated to contain a point reference. The point can be referenced from the Point Table attached at the MovePath. The coordinates are resolved when the command is copied to the internal buffer. For circular commands the variable "AuxPoint" can also be connected to the point table via point parameter in the circle parameters.

#### NOTE

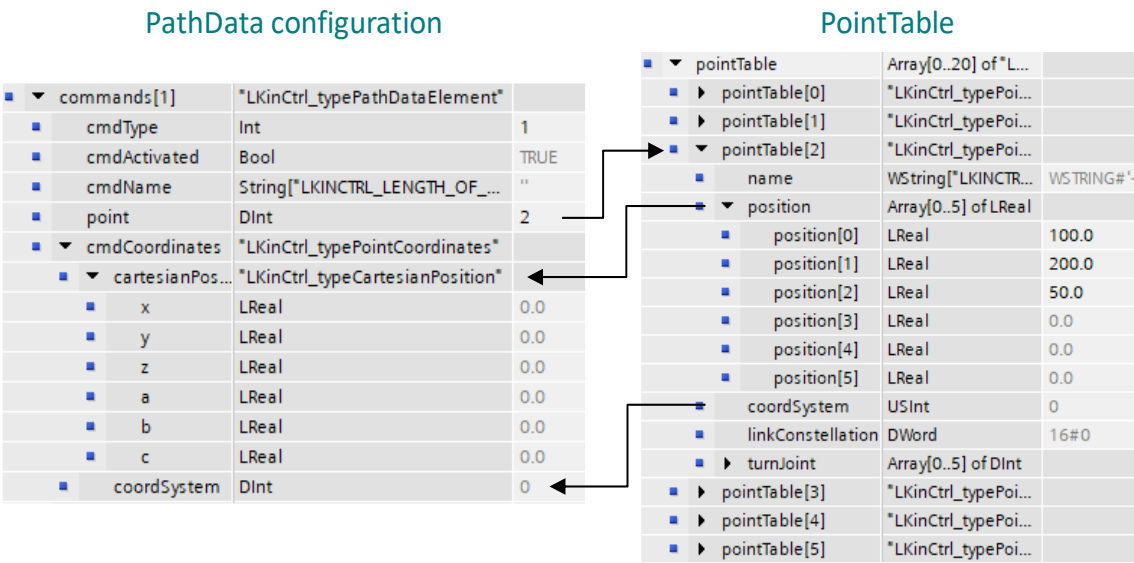
Any value equal or greater 0 in the variable point is interpreted as point reference.  
If no reference to the point table shall be used, the value must be set to -1.  
If an invalid point number is specified, an error is thrown.

Point references are valid for the following commands:

- Linear command
- Circular command + aux point mode 0 and 1
- Pick and place command
- Point to point command
- OCS frame command
- UCS frame command
- Define tool command

The following image shows an exemplary point reference. The command refers to point number 2 in the point table. When the FB MovePath resolves the reference, the values from the point table are used for the command.

Figure 3-1: Example of point reference



3.3. Linear commands

Linear commands can be executed with absolute target coordinates (*cmdType* = 1) or relative coordinates (*cmdType* = 2). When using relative commands, the cartesian coordinates are interpreted as distance to travel in the respective direction.

3.4. Circular commands

A circular motion is programmed with *CmdType* = 3 (absolute coordinates) or *CmdType* = 4 (relative coordinates). Three different types of circle definition are supported, the types are selected via the *circMode* variable in the command. The following chapters are explaining the different circle modes in detail.

For some circle modes it is necessary to define the plane in which the circle is executed. The following settings are possible. The images in the lower chapters always refer to coordinate 1 and 2 as shown in the table.

Table 3-1: Circle plane settings

CirclePlane	System term	Coordinate 1	Coordinate 2
0	X-Z plane	Z	X
1	X-Y plane	Y	Z
2	X-Y plane	X	Y

Furthermore, it can be necessary to define a path choice. The following settings are possible. The images below refer to them with a unique color for each setting.

Table 3-2: Path choice settings

PathChoice	Circle segment	Math	Clock
0	Shorter positive	Positive	Counter-clockwise
1	Shorter negative	Negative	Clockwise
2	Longer positive	Positive	Counter-clockwise

PathChoice	Circle segment	Math	Clock
3	Longer negative	Negative	Clockwise

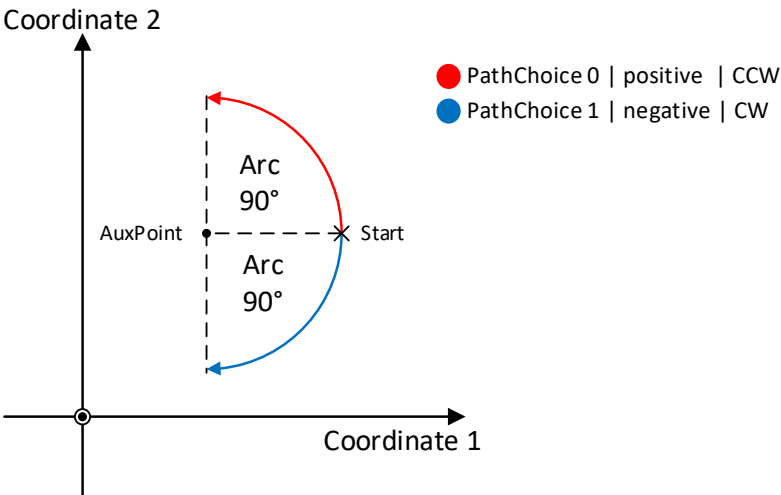
**NOTE**

PathChoice 2 & 3 are only valid when defining the circle via endpoint and radius (*circMode* = 2).

3.4.1.      **CircMode = 1: AuxPoint + Arc**

The circle is specified by the circle center (*AuxPoint*) and the opening angle (*Arc*). Additionally, the circle direction (*PathChoice*) and the circle plane (*CirclePlane*) must be specified. This type of specification allows for full circles in a 2D plane.

Figure 3-2: Arc example



The following figure highlights the necessary variables for this circle definition in a PathData command.

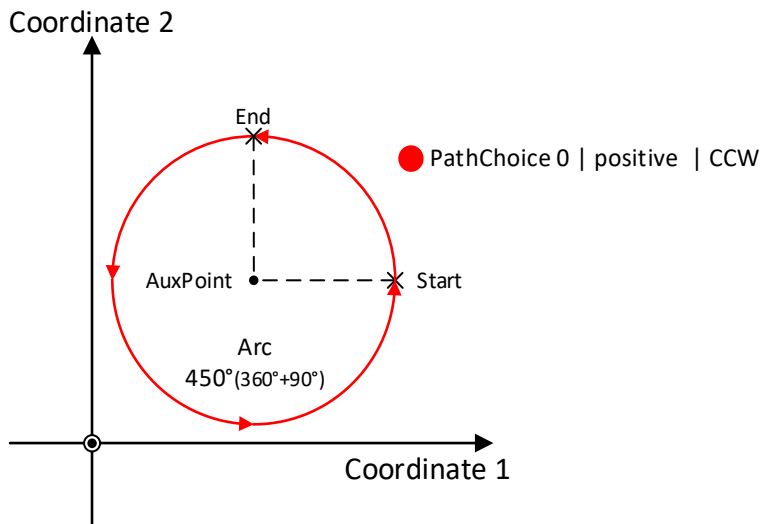
Figure 3-3: Necessary variables in PathData Command

cmdParameters	"LKInCtrl_typePointParameter"	
pathDynamics	"LKInCtrl_typePathDynamics"	
orientationDynamics	"LKInCtrl_typePathDynamics"	
orientationDirection	DInt	3
bufferMode	DInt	1
transitionParameters	Array[1..5] of LReal	
dynamicAdaption	DInt	-1
toolNumber	DInt	0
moveDirectParameters	"LKInCtrl_typeMoveDirectParameter"	
offsetParameters	"LKInCtrl_typeToolParameters"	
conveyorParameters	"LKInCtrl_typeConveyorParameter"	
circleParameters	"LKInCtrl_typeCircleParameters"	
circMode	DInt	0
point	DInt	-1
auxPoint	Array[1..3] of LReal	
auxPoint[1]	LReal	0.0
auxPoint[2]	LReal	0.0
auxPoint[3]	LReal	0.0
pathChoice	DInt	0
circlePlane	DInt	0
radius	LReal	0.0
arc	LReal	0.0
toolParameters	"LKInCtrl_typeToolParameters"	
setFlags	Array[1..*LKINCTRL_NO_OF_CMD_SE...	

### Full circle or more

To specify a full circle, set *arc* to a value greater or equal  $360^\circ$ . As shown in the following image with an *arc* setting of  $450^\circ$ .

Figure 3-4:  $450^\circ$  Arc example



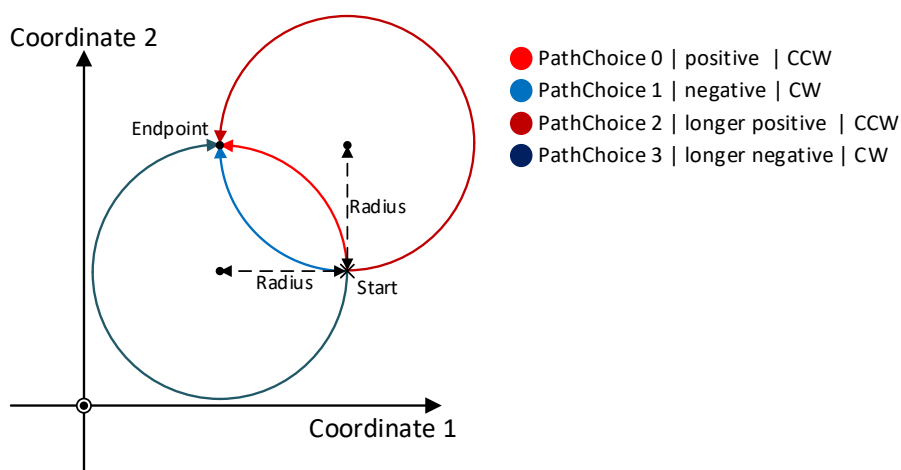
### 3.4.2. **CircMode = 2: Endpoint + Radius**

The circle is specified via the circle endpoint and the radius (*Radius*). Additionally, the circle direction (*PathChoice*) and the circle plane (*CirclePlane*) must be specified. This type of specification allows for circles below  $360^\circ$  in a 2D plane.

#### NOTE

The coordinates for the circle endpoint must be entered in the structure *cartesianPosition*.

Figure 3-5: Circle examples



The following figure highlights the necessary variables for this circle definition in a PathData command.

Figure 3-6: Necessary variables in PathData Command

cmdType	Int	-1
cmdActivated	Bool	TRUE
cmdName	String["LKINCTRL_LENGTH_OF_CMD..."]	"
point	DInt	-1
cmdCoordinates	"LKInCtrl_typePointCoordinates"	
cartesianPosition	"LKInCtrl_typeCartesianPosition"	
x	LReal	0.0
y	LReal	0.0
z	LReal	0.0
a	LReal	0.0
b	LReal	0.0
c	LReal	0.0
coordSystem	DInt	0
cmdParameters	"LKInCtrl_typePointParameter"	
pathDynamics	"LKInCtrl_typePathDynamics"	
orientationDynamics	"LKInCtrl_typePathDynamics"	
orientationDirection	DInt	3
bufferMode	DInt	1
transitionParameters	Array[1..5] of LReal	
dynamicAdaption	DInt	-1
toolNumber	DInt	0
moveDirectParameters	"LKInCtrl_typeMoveDirectParameter"	
offsetParameters	"LKInCtrl_typeToolParameters"	
conveyorParameters	"LKInCtrl_typeConveyorParameter"	
circleParameters	"LKInCtrl_typeCircleParameters"	
circMode	DInt	0
point	DInt	-1
auxPoint	Array[1..3] of LReal	
auxPoint[1]	LReal	0.0
auxPoint[2]	LReal	0.0
auxPoint[3]	LReal	0.0
pathChoice	DInt	0
circlePlane	DInt	0
radius	LReal	0.0
arc	LReal	0.0
zoneParameters	"LKInCtrl_typeZoneParameters"	

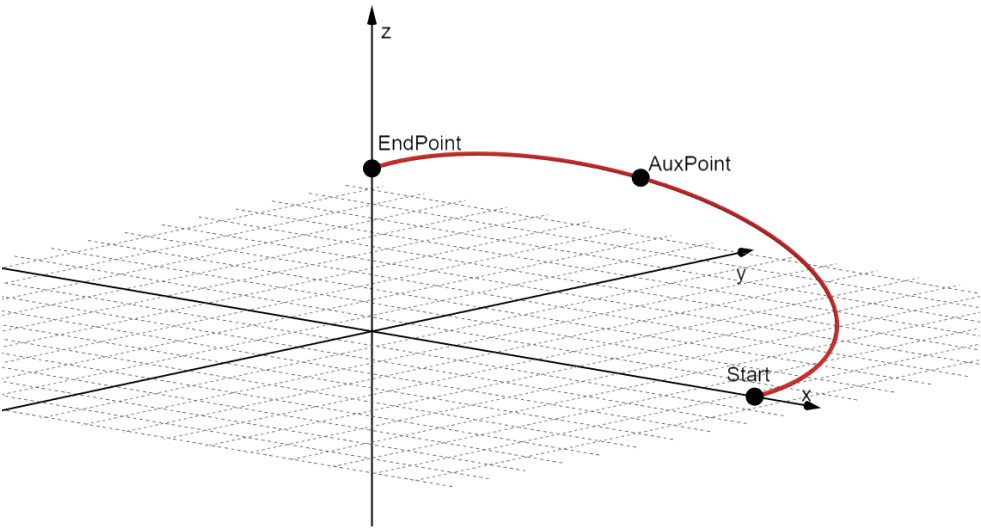
### 3.4.3. **CircMode = 0: Endpoint + AuxPoint (Circle in 3D space)**

The circle is specified via the circle endpoint (*EndPoint*) and an intermediate point on the circle (*AuxPoint*). The circle direction (*PathChoice*) and the circle plane (*CirclePlane*) are **irrelevant**. This type of specification allows for circles below 360° in 3D space.

**NOTE**

The coordinates for the circle endpoint must be entered in the structure *cartesianPosition*.

Figure 3-7: Circle in 3D space



The following figure highlights the necessary variables for this circle definition in a PathData command.



Figure 3-8: Necessary variables in PathData Command

cmdType	Int	-1
cmdActivated	Bool	TRUE
cmdName	String["LKINCTRL_LENGTH_OF_CMD..."]	""
point	DInt	-1
cmdCoordinates	"LKInCtrl_typePointCoordinates"	
cartesianPosition	"LKInCtrl_typeCartesianPosition"	
x	LReal	0.0
y	LReal	0.0
z	LReal	0.0
a	LReal	0.0
b	LReal	0.0
c	LReal	0.0
coordSystem	DInt	0
cmdParameters	"LKInCtrl_typePointParameter"	
pathDynamics	"LKInCtrl_typePathDynamics"	
orientationDynamics	"LKInCtrl_typePathDynamics"	
orientationDirection	DInt	3
bufferMode	DInt	1
transitionParameters	Array[1..5] of LReal	
dynamicAdaption	DInt	-1
toolNumber	DInt	0
moveDirectParameters	"LKInCtrl_typeMoveDirectParameter"	
offsetParameters	"LKInCtrl_typeToolParameters"	
conveyorParameters	"LKInCtrl_typeConveyorParameter"	
circleParameters	"LKInCtrl_typeCircleParameters"	
circMode	DInt	0
point	DInt	-1
auxPoint	Array[1..3] of LReal	
auxPoint[1]	LReal	0.0
auxPoint[2]	LReal	0.0
auxPoint[3]	LReal	0.0

### 3.5. MoveDirect / synchronous Point-to-Point (sPTP)

An sPTP motion is programmed with *CmdType* = 5 (absolute coordinates) or *CmdType* = 6 (relative coordinates).

#### Coordinates

The target coordinates can be specified as Cartesian (WCS & OCS) or as axis positions (MCS).

To select axis positions the *coordSystem* must be set to 100. The cartesian coordinates are then interpreted as axis values.

To select joint positions the *coordSystem* must be set to 101. The cartesian coordinates are then interpreted as joint values.

The assignment of the cartesian coordinates to the respective values is shown in the following table.

Table 3-3: Respective axes coordinates

Cartesian position	Axis position	Joint position
X	A1	J1
Y	A2	J2
Z	A3	J3
A	A4	J4
B	A5	J5
C	A6	J6

## Link constellation

Depending on the kinematics type, a kinematics system can reach Cartesian coordinates via various axis positions. The kinematics type defines the possible axis positions and the positive and negative axis position space.

The kinematics technology object indicates the current axis position in the *StatusKinematics.LinkConstellation* tag.

The kinematics system cannot exit the axis position space during a linear or circular (continuous path) motion. However, it is possible to change the link constellation with sPTP commands in WCS & OCS. The desired link constellation can be set at the command. If no change shall be done the link constellation must be set to the default value of `16#FFFF_FFFF`.

When using axis coordinates, the link constellation is not regarded, as the axis positions already define the link constellation.

Figure 3-9: sPTP settings

▼ commands[1]	"LKInCtrl_typePathDataElement"	
cmdType	Int	-1
cmdActivated	Bool	TRUE
cmdName	String["LKINCTRL_LENGTH_OF_CMD..."]	"
▼ cmdCoordinates	"LKInCtrl_typePointCoordinates"	
cartesianPosition	"LKInCtrl_typeCartesianPosition"	
coordSystem	DInt	0
▼ pathParameters	"LKInCtrl_typePathParameters"	
pathDynamics	"LKInCtrl_typePathDynamics"	
orientationDynamics	"LKInCtrl_typePathDynamics"	
orientationDirection	DInt	3
bufferMode	DInt	1
transitionParameters	Array[1..5] of LReal	
dynamicAdaption	DInt	-1
▼ moveDirectParameters	"LKInCtrl_typeMoveDirectParameter"	
linkConstellation	DWord	16#FFFF_FFFF
positionMode	DInt	1
turnJoint	Array[1..6] of DInt	
▼ conveyorParameters	"LKInCtrl_typeConveyorParameter"	

## Position mode

The position mode is only used for absolute sPTP commands (*CmdType* = 5). It can be used to program the orientation A with absolute or relative coordinates. It is only relevant for movements within WCS & OCS.

*PositionMode* = 1 → Absolute orientation A

*PositionMode* = 2 → Relative orientation A

## TurnJoint

Depending on the kinematics type, a kinematics system can reach Cartesian coordinates with multiples of the same joint position. To define the target joint value, the parameter TurnJoint can be used. If turnJoint is set to 0 the shortest distance is travelled. If a value not equal to 0 is set, the respective range is used.

For more details refer to the TO Kinematics manual linked in [8.3](#)

## Dynamics


































The dynamics are a factor of the axis dynamic limits. The dynamics are scaled from 0.0 to 1.0. Meaning that if the velocity factor is set to 1.0, the maximum axis velocity is used. If set to 0.1, only 10% of the maximum axis velocity is taken. Same applies to acceleration and deceleration.

### NOTE

Jerk is limited from 0.1 to 0.9. A jerk of 0.0 means that there is no limitation. Resulting in the motion following a trapezoidal dynamics profile.

When setting the dynamics to a negative value (generally -1), the default factors in the Kinematics DB are used.

Figure 3-10: MoveDirect default dynamics in kinematics DB

11			▼ DynamicDefaults	TO_Struct_Kinema...		
12			▶ Path	TO_Struct_Kinema...		
13			▶ Orientation	TO_Struct_Kinema...		
14			DynamicAdaption	DInt	2	
15			▼ MoveDirect	TO_Struct_Kinema...		
16			VelocityFactor	LReal	0.1	
17			AccelerationFactor	LReal	0.1	
18			DecelerationFactor	LReal	0.1	
19			JerkFactor	LReal	0.5	
20			▶ DynamicLimits	TO_Struct_Kinema...		
21			▶ MotionQueue	TO_Struct_Kinema...		

**NOTE**

Velocity, acceleration or deceleration cannot be 0.0.

**Remaining Distance / ExecutionTimeStatus**

sPTP commands do not show a remaining distance. Instead, the execution time status is shown, which is a time-based factor from 0.0 to 1.0. Meaning that if the execution time status is 0.5, the command has executed 50% of its calculated duration.

## 3.6. Modal dynamics

Modal dynamics commands are used to set the dynamics for all subsequent motion commands that are programmed with default dynamics (-1). The modal dynamics are initialized with the kinematics default values. This means that if the modal dynamics commands are not used the kinematics default are used. The modal dynamics are separated by path, orientation and sPTP. They are reset upon the next path execution by default. This reset can be controlled with the move path configuration parameter *resetModelDynamics*.

Each modal dynamics command has a parameter for velocity, acceleration, deceleration and jerk. Each of these parameters can be active or inactive. If the parameter is inactive the current modal value for this parameter is kept. To deactivate a parameter set it to -2. If a parameter is active it can either be set to the default kinematics value (-1) or any other manual value (>0).

### 3.6.1. Set path dynamics

A set path dynamics command is programmed with the CmdType 70. The dynamic values are specified under *cmdParameters.pathDynamics*. In the example the velocity parameter is active with a manual value, the acceleration and deceleration are active with default values, and the jerk is inactive.

Figure 3-11: Example set path dynamics command

```
// 1. Command - Set path dynamics
"ExampleProject_Data".pathdataArray["PATH_INDEX_SCENARIO_2"].commands[#tempCounter].cmdType := "LKINCTRL_MC_SET_PATH_DYNAMICS";
"ExampleProject_Data".pathdataArray["PATH_INDEX_SCENARIO_2"].commands[#tempCounter].cmdParameters.pathDynamics.velocity := 100; // active - manual value
"ExampleProject_Data".pathdataArray["PATH_INDEX_SCENARIO_2"].commands[#tempCounter].cmdParameters.pathDynamics.acceleration := -1; // active - kinematics default value
"ExampleProject_Data".pathdataArray["PATH_INDEX_SCENARIO_2"].commands[#tempCounter].cmdParameters.pathDynamics.deceleration := -1; // active - kinematics default value
"ExampleProject_Data".pathdataArray["PATH_INDEX_SCENARIO_2"].commands[#tempCounter].cmdParameters.pathDynamics.jerk := -2; // inactive - current value is kept
#tempCounter += 1;
```

### 3.6.2. Set orientation dynamics

A set orientation dynamics command is programmed with the CmdType 71. The dynamic values are specified under *cmdParameters.orientationDynamics*.

### 3.6.3. Set sPTP dynamics

A set path dynamics command is programmed with the CmdType 72. The dynamic values are specified under *cmdParameters.pathDynamics*.

The dynamics are scaled from 0.0 to 1.0. Meaning that if the velocity factor is set to 1.0, the maximum axis velocity is used. If set to 0.1, only 10% of the maximum axis velocity is taken. Same applies to acceleration and deceleration.

#### NOTE

Jerk is limited from 0.1 to 0.9. A jerk of 0.0 means that there is no limitation. Resulting in the motion following a trapezoidal dynamics profile.

## 3.7. Conveyor Tracking

### 3.7.1. Track conveyor belt

A track conveyor belt command is used to synchronize a kinematics with a conveyor belt. A track conveyor belt command is programmed by setting *CmdType* to 10.

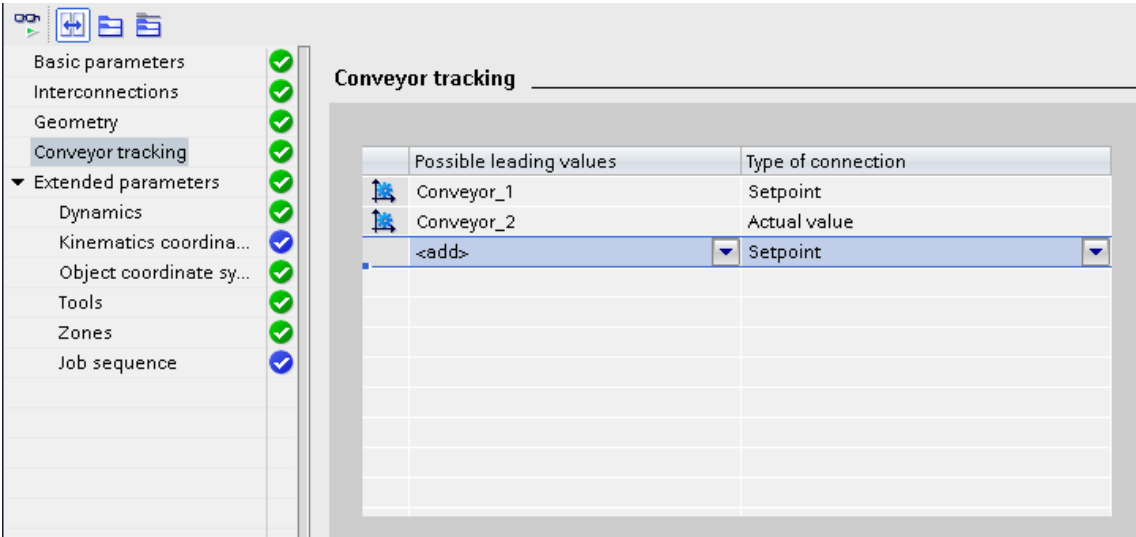
To track a conveyor the technology object kinematics uses an OCS frame and updates it constantly by a given leading value e.g. a conveyor position. The OCS frame selection can be made in the *PathData* at the track conveyor command.

For more information on conveyor tracking please refer to "SIMATIC S7-1500 Amendments to documentation" linked in the chapter [Links and Literature](#).

#### Prerequisites

Every conveyor axis must be listed at the TO Kinematics conveyor tracking tab.

Figure 3-12: Conveyor axes connection



Afterwards the conveyor parameter must be configured at the FB MovePath input *Configuration.ConveyorParameter*. There the axis technology object (*conveyorBeltAxis*), *ConveyorBeltOrigin* and *InitialObjectPosition* need to be set.

Figure 3-13: Configuration input

10	configuration	"LKInCtrl_typeConfiguration"	
11	offsetParameter	Array[1.."LKINCTRL_NO_OF_OFFSETS"] of "LKInCtrl_typeContourOffsetParameter"	
12	conveyorParameter	Array[1.."LKINCTRL_NO_OF_CONVEYOR"] of "LKInCtrl_typeConveyorConfiguration"	
13	conveyorParameter[1]	"LKInCtrl_typeConveyorConfiguration"	
14	conveyorBeltAxis	DB_ANY	0
15	conveyorBeltOrigin	TO_Struct_Kinematics_Frame	
16	x	LReal	0.0
17	y	LReal	0.0
18	z	LReal	0.0
19	a	LReal	0.0
20	b	LReal	0.0
21	c	LReal	0.0
22	initialObjectPosition	TO_Struct_Kinematics_Frame	
23	x	LReal	0.0
24	y	LReal	0.0
25	z	LReal	0.0
26	a	LReal	0.0
27	b	LReal	0.0
28	c	LReal	0.0
29	conveyorParameter[2]	"LKInCtrl_typeConveyorConfiguration"	

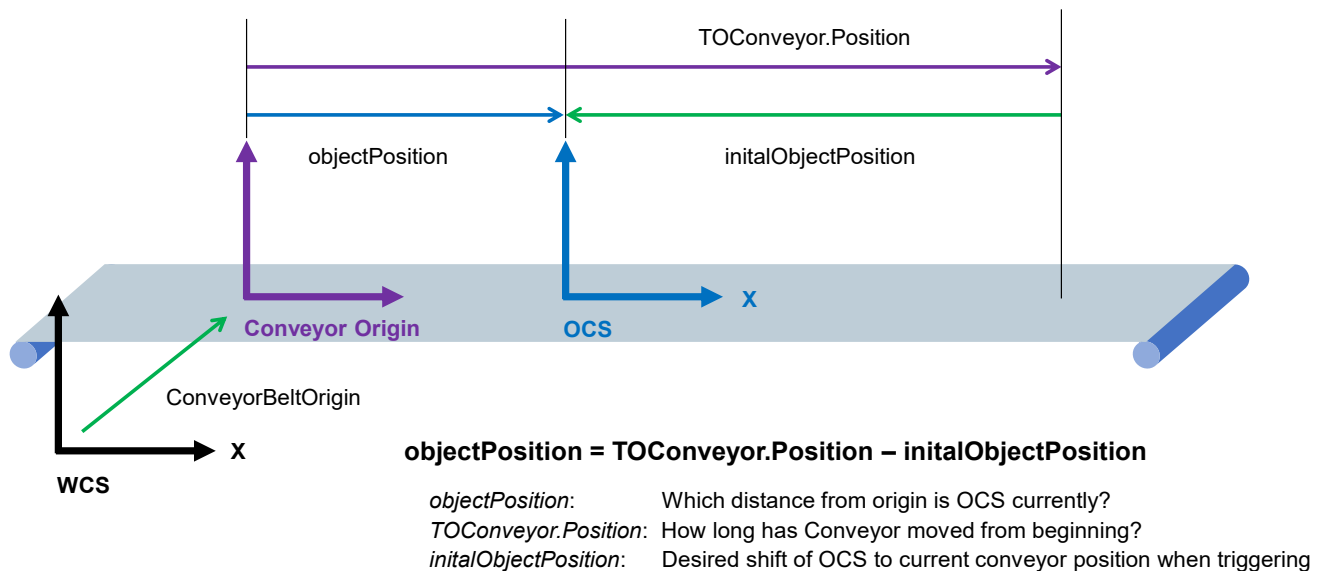
The *ConveyorBeltOrigin* usually describes the position of the measuring system, which gives a trigger signal when a product is detected.

The *initialObjectPosition* then receives the measured position value.

With the current position of the conveyor and the *initialObjectPosition*, the actual object position is calculated internally. The origin of the connected OCS should then be equal to the moving product position.

The following figure gives an overview of these settings.

Figure 3-14: Conveyor overview



The following figure shows an exemplary conveyor configuration.

Figure 3-15: Conveyor configuration

```

1 // Conveyor 1
2 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].conveyorBeltAxis := "Conveyor_1";
3 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].conveyorBeltOrigin.x := 0.0;
4 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].conveyorBeltOrigin.y := 50.0;
5 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].conveyorBeltOrigin.z := 0.0;
6 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].conveyorBeltOrigin.a := 45.0;
7 #instLKinCtrl_MovePath.configuration.conveyorParameter[1].initialObjectPosition.x := "Conveyor_1".Position;
8
9 // Conveyor 2
10 #instLKinCtrl_MovePath.configuration.conveyorParameter[2].conveyorBeltAxis := "Conveyor_2";
11 #instLKinCtrl_MovePath.configuration.conveyorParameter[2].conveyorBeltOrigin.x := 0.0;
12 #instLKinCtrl_MovePath.configuration.conveyorParameter[2].conveyorBeltOrigin.y := -100.0;
13 #instLKinCtrl_MovePath.configuration.conveyorParameter[2].conveyorBeltOrigin.z := 0.0;
14 #instLKinCtrl_MovePath.configuration.conveyorParameter[2].initialObjectPosition.x := "Conveyor_2".Position;
15

```

Because no external position measuring device is used, the conveyor axis position is used as a substitute.

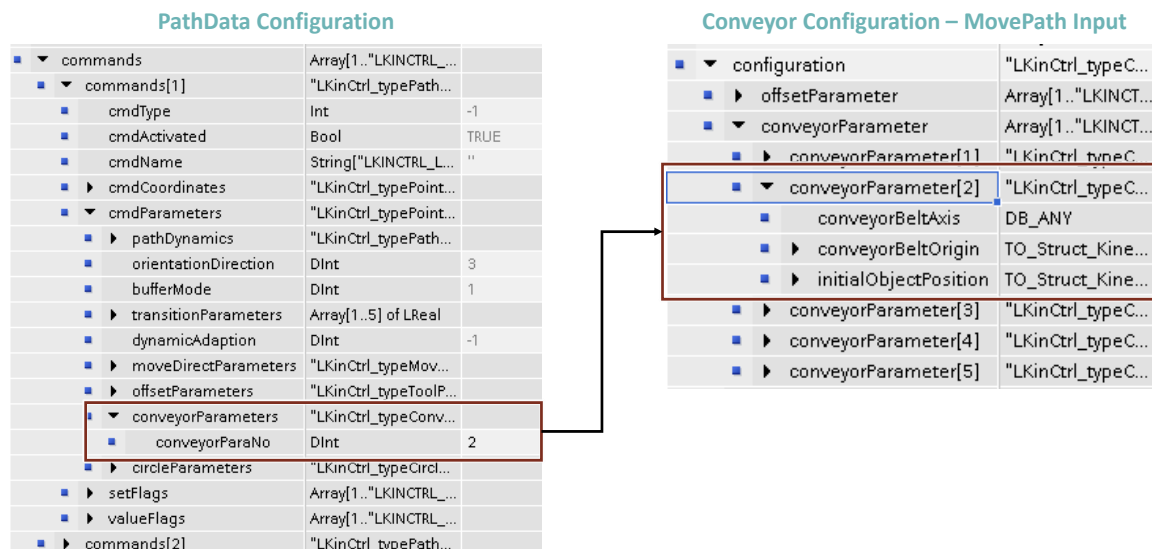
#### NOTE

The setting *initialObjectPosition* only allows values for "x". All other coordinates must be set to 0.0.

### Configuration in PathData

To program a conveyor tracking command in the *PathData*, the *CmdType* must be set to 10. Additionally, the object coordinate system (OCS), which shall be used for tracking, must be specified in the variable *cmdCoordinates.coordSystem* (only 1-3 allowed). Finally, the conveyor configuration needs to be selected. Set the variable *cmdParameters.conveyorParameters.conveyorParaNo* to the index of which conveyor configuration you want to track. The following figure shows the connection.

Figure 3-16: Conveyor parameter selection



**Example:**

Figure 3-17: Example path data

```
// 1. Command - Move to start
#pathData.commands[1].cmdType := "LKINCTRL_MC_MOVELINEARABS";
// command coordinates
#pathData.commands[1].cmdCoordinates.cartesianPosition.x := 10.0;
#pathData.commands[1].cmdCoordinates.cartesianPosition.y := 20.0;
#pathData.commands[1].cmdCoordinates.cartesianPosition.z := 30.0;
#pathData.commands[1].cmdCoordinates.coordSystem := "CS_WCS";
// -----

// 2. Command - Track conveyor
#pathData.commands[2].cmdType := "LKINCTRL_MC_TRACKCONVEYORBELT";
// Set conveyor parameter number
#pathData.commands[2].cmdParameters.conveyorParameters.conveyorParaNo := 1;
// Set OCS to be used for tracking
#pathData.commands[2].cmdCoordinates.coordSystem := "CS_OCS1";
// -----

// 3. Command - Sync to conveyor
#pathData.commands[3].cmdType := "LKINCTRL_MC_MOVELINEARABS";
// command coordinates
#pathData.commands[3].cmdCoordinates.cartesianPosition.x := 0.0;
#pathData.commands[3].cmdCoordinates.cartesianPosition.y := 0.0;
#pathData.commands[3].cmdCoordinates.cartesianPosition.z := 0.0;
#pathData.commands[3].cmdCoordinates.coordSystem := "CS_OCS1";
// -----
```

**Result:**

The conveyor tracking command is added to the motion queue of the kinematics. After the successful execution of the command, the tracking is running. Meaning the OCS is coupled to the conveyor belt position (leading value). With the next command referring to the same OCS, the kinematics synchronizes with the conveyor belt.

**NOTE**

Up to FW 3.0.x, dynamic adaption must be deactivated for all commands during conveyor tracking!

**NOTE**

sPTP commands are not allowed during conveyor tracking!

**Blending behavior**

Up to FW 3.0.x, Blending is not possible for the following motions:

- Into a motion job that completes the tracking at the conveyor ("In Sync" → "Sync Off")
- From a motion job for moving into the first position in the tracked OCS into the subsequent motion job in the tracked OCS ("Sync On" → "In Sync")
- From a motion job that exits tracking to the subsequent motion job which enables tracking again ("Sync Off OCS1" → "Sync On OCS2"), if the second command is executed while the first one is already active

**Stop tracking**

The synchronization is ended if a command in WCS or untracked OCS is executed. Desynchronization commands can only be absolute linear or absolute circular commands (*CircMode* = 0).





### Kinematics is not desynchronized with MC\_GroupStop

A MC\_GroupStop command stops all currently running commands and deletes the motion queue. However, if the kinematics is currently synchronized with a conveyor belt, the synchronization is not canceled. The kinematics continues to move synchronously with the conveyor.

### Stop behavior

The FB MovePath offers the possibility to desynchronize automatically from the conveyor in case of stop or error. If activated, a MC\_GroupStop is called to stop the current movement in OCS and to delete the motion queue, then a linear command to the current WCS position is executed. This results in a desynchronization at the current location with a minimal movement.

To activate the automatic desynchronization, set the input *configuration.stopMode* to 10 or 11, depending on the dynamics you want to execute the MC\_GroupStop with.

Table 3-4: StopMode overview

StopMode	Stop dynamics	Desynchronization
0	Stop with current dynamics	No desynchronization
1	Stop with maximum dynamics	No desynchronization
10	Stop with current dynamics	Desynchronization after stop
11	Stop with maximum dynamics	Desynchronization after stop

### 3.7.2. Desynchronize conveyor

With *cmdType* 50 you can desynchronize from on a conveyor with only a small movement from the last position. Internally a linear command is executed with the current TCP position as target.

## 3.8. Frame commands

### 3.8.1. OCS frame

With *CmdType* 20 you can execute a MC\_SetOCSFrame command. The instruction defines a position and rotation of an object coordinate system (OCS) in relation to the world coordinate system (WCS)

Select the coordinate system you want to set at *cmdCoordinates.coordSystem* (only 1-3 allowed) and set the frame parameters at *cmdCoordinates.cartesianPosition*. For some types of kinematics not all coordinates can be changed. An overview can be found in [131](#)

### 3.8.2. UCS frame (User Coordinate System)

With *CmdType* 21 you can set a user frame. The instruction defines a position and rotation of a user coordinate system (UCS) in relation to the world coordinate system (WCS)

Select the coordinate system you want to set at *cmdCoordinates.coordSystem* and set the frame parameters at *cmdCoordinates.cartesianPosition*.

The first user frame number is defined with the PLC Tag: *LKINCTRL\_CS\_NO\_OF\_FIRST\_USER\_FRAME*

The available number of frames is defined with the PLC Tag: *LKINCTRL\_CS\_NO\_OF\_LAST\_USER\_FRAME*

Start values of user frames are set in configuration input of MovePath and are used if the MovePath is executed from standstill. The status of the user frames can be seen at the MovePath output *activeUserFrames*.

To execute motion commands in an UCS, set the coordinate system to a number between the first and last user frame index.

**NOTICE****Some circle modes not possible with user frame rotation!**

If a user frame has a rotation set (A, B or C  $\neq$  0.0), then circular commands with circModes 1 or 2 cannot be used within this user coordinate system. If programmed anyway, the FB MovePath will output an error.

**3.8.3. Define Tool**

With *CmdType* 22 you can execute an MC\_DefineTool command. The instruction redefines the tool frame of the tool 1.

Set the frame parameters at *cmdCoordinates.cartesianPosition*. For some types of kinematics not all coordinates can be changed. An overview can be found in [31](#)

**NOTE**

The job can only be executed if the kinematics is in standstill. The MovePath handles this automatically by letting the motion queue run dry. Blending at this command is therefore not possible.

**3.8.4. Set Tool**

With *CmdType* 23 you can execute a MC\_SetTool command. The instruction activates the specified tool. The tool number is set at *cmdParameters.toolNumber*.

**NOTE**

The job can only be executed if the kinematics is in standstill. The MovePath handles this automatically by letting the motion queue run dry. Blending at this command is therefore not possible.

**3.9. Zone commands****3.9.1. Define workspace zone**

With *CmdType* 30 you can execute a MC\_DefineWorkspaceZone command. The instruction redefines the specified workspace zone.

The new zone definition is stored at the MovePath input *configuration.workspaceZones*.

The zone definition needs to be selected at *cmdParameters.zoneParameters.zoneConfigurationIndex*.

The zone number at the TO Kinematics which shall be redefined is set at *cmdParameters.zoneParameters.zoneNumber*.

**NOTE**

The zone definitions at the MovePath configuration input are not active by default!

**3.9.2. Activate workspace zone**

With *CmdType* 31 you can execute a MC\_SetWorkspaceZoneActive command. The instruction activates the specified workspace zone.

The number must be specified at *cmdParameters.zoneParameters.zoneNumber*.

### 3.9.3. Deactivate workspace zone

With *CmdType* 32 you can execute a *MC\_SetWorkspaceZoneInactive* command. The instruction deactivates workspace zones. Depending on the deactivation mode one or more zones are deactivated. The mode is set at *cmdParameters.zoneParameters.deactivationMode*.

Following modes are possible:

Table 3-5: Zone deactivation modes

Mode	Functionality	LKinCtrl constant
0	Specific zone	ZONE_DEACTIVATION_MODE_SPECIFIC
1	All workspace zones	ZONE_DEACTIVATION_MODE_ALL
2	All blocked zones	ZONE_DEACTIVATION_MODE_BLOCKED
3	All signal zones	ZONE_DEACTIVATION_MODE_SIGNAL
4	Currently active workspace	ZONE_DEACTIVATION_MODE_WORKSPACE

#### NOTE

When using mode 0 (specific zone) the zone number must be specified at *cmdParameters.zoneParameters.zoneNumber*.

### 3.9.4. Define kinematics zone

With *CmdType* 33 you can execute a *MC\_DefineKinematicsZone* command. The instruction redefines the specified kinematics zone.

The new zone definition is stored at the MovePath input *configuration.kinematicsZones*.

The zone definition needs to be selected at *cmdParameters.zoneParameters.zoneConfigurationIndex*.

The zone number at the TO Kinematics which shall be redefined is set at *cmdParameters.zoneParameters.zoneNumber*.

#### NOTE

The zone definitions at the MovePath configuration input are not active by default!

### 3.9.5. Activate kinematics zone

With *CmdType* 34 you can execute an *MC\_SetKinematicsZoneActive* command. The instruction activates the specified workspace zone.

The number must be specified at *cmdParameters.zoneParameters.zoneNumber*.

### 3.9.6. Deactivate kinematics zone

With *CmdType* 35 you can execute an *MC\_SetKinematicsZoneInactive* command. The instruction deactivates kinematics zones. Depending on the deactivation mode one or more zones are deactivated. The mode is set at *cmdParameters.zoneParameters.deactivationMode*.

Following modes are possible:

Table 3-6: Zone deactivation modes

Mode	Functionality	LKinCtrl constant
0	Specific zone	ZONE_DEACTIVATION_MODE_SPECIFIC
1	All kinematics zones	ZONE_DEACTIVATION_MODE_ALL

**NOTE**

When using mode 0 (specific zone) the zone number must be specified at *cmdParameters.zoneParameters.zoneNumber*.

### 3.10. Flags

#### 3.10.1. Principle of operation

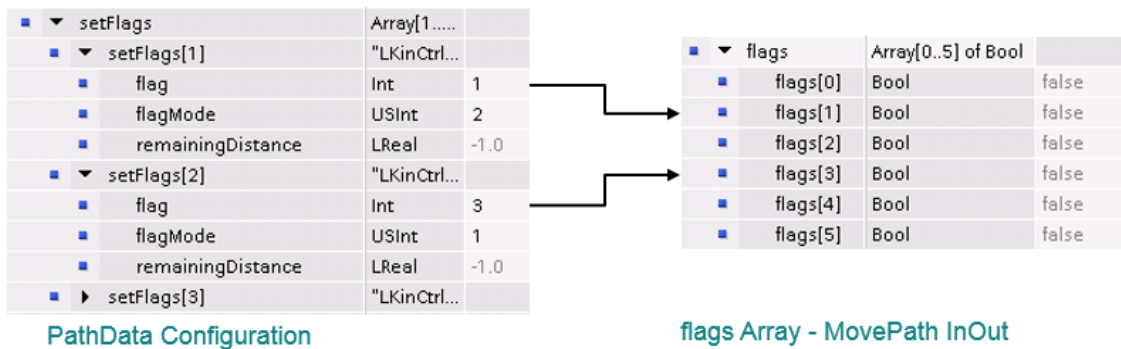
Flags provide the possibility to control actuators according to the active PathData command. The configuration of flags takes place in the PathData structure. Every command entry in the PathData enables the individual setting of flags. Therefore, Boolean or LReal InOut signals at the *LKinCtrl\_MC\_MovePath* interfaces *setFlags* and *valueFlags* can be set accordingly to the command that is active at TO Kinematics. By selecting a specific flag mode, the timing to set and reset the InOut flags can be specified. Also wait points can be programmed by waiting for an external acknowledge of a flag set during the path motion. In the following, explanations will exemplarily refer to *setFlags* only, as the basic principle is identical for *setFlags* and *valueFlags*.

NOTE

Different flag modes offer specific timing possibilities for the flags to be set and reset.

#### Configuration of Flags

Figure 3-18: setFlags in PathData define flags in MovePath interface to be set



The *setFlags* structure shown left in the upper figure is available for every command entry in the PathData. The *flag* entry in the *setFlags* array refers to the array index of the *flags* InOut parameter of the FB. The referred array entry in the *flags* array is set or reset depending on the used *flagMode*.

The default value for all *flags* is set to -1. This default value does not trigger any changes to the flags. It also does not reset any flags.

Also by default, the number of available flags of both types to define for each command is set to 3. Therefore 3 flags of each type can be set at the same time, once a specific command is activated. Depending on the use case, this number can be changed by the library constants *LKINCTRL\_NO\_OF\_CMD\_SETFLAGS* for *setFlags* and *LKINCTRL\_NO\_OF\_CMD\_VALUEFLAGS* for *valueFlags*.

If more flags at the InOut parameter of *LKinCtrl\_MC\_MovePath* are needed, the default number of 6 flags can be changed in the same way by adapting the library constant *LKINCTRL\_NO\_OF\_LAST\_FLAG*.

NOTICE

Actual number of flags[0.. *LKINCTRL\_NO\_OF\_LAST\_FLAG*]

Due to the *flags* arrays beginning with the index count 0, the number of available flags is one count higher than the value of the constant *LKINCTRL\_NO\_OF\_LAST\_FLAG*.

NOTICE

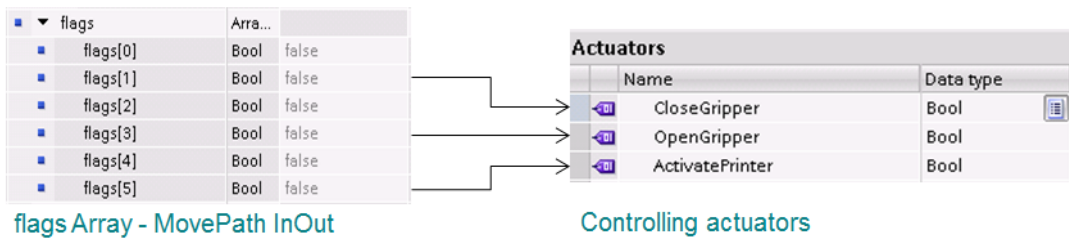
Use of *flags* in multiple PathData structures:

To separate flags set in different PathData structures blending into one another, different PathDataNames need to be defined in each PathData.

Controls

The flags can be connected to outputs to use them as actuator controls.

Figure 3-19: *flags[0...n]* connected to actuators



Reset of flags

By default, the flag arrays at the InOut parameter of *LKinCtrl\_MC\_MovePath* are reset in the following cases:

- Start of PathData
- Done signal of LKinCtrl\_MC\_MovePath
- To deactivate the reset, the global PLC Tag *LKINCTRL\_DISABLE\_FLAG\_RESET* can be set to true.

### 3.10.2. Flag modes

To define the timing to set and reset the flag and to have the command execution for external acknowledge the parameter *flagMode* can be used. [Table 3-7](#) shows an overview of the available flag modes.

Table 3-7: Available flag modes

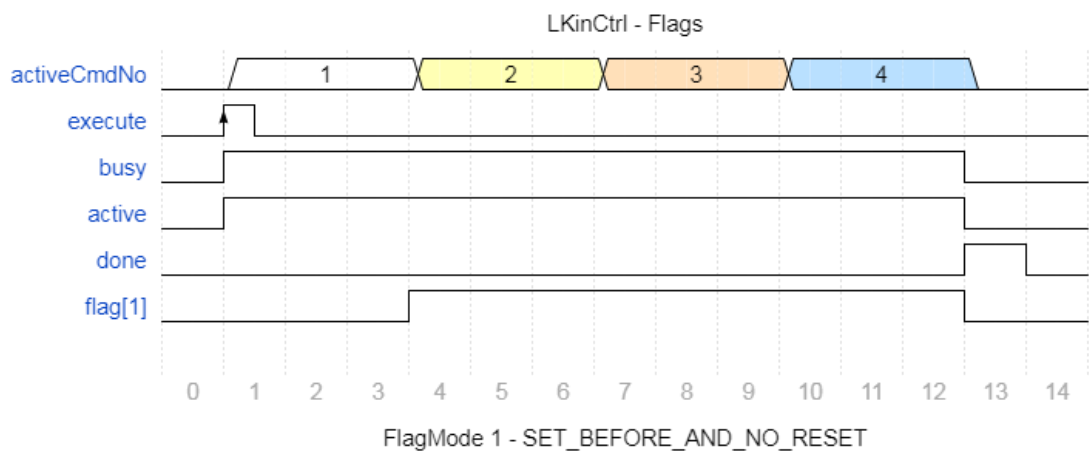
Flag mode	Flag function
0	Flag deactivated
1	SET_BEFORE_AND_NO_RESET
2	SET_BEFORE_AND_RESET_AFTER
3	SET_BEFORE_AND_RESET_AFTER_ONE_CYCLE
5	SET_BEFORE_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE
10	SET_IN_REMAINING_DISTANCE_TO_TARGET
11	SET_AFTER_AND_NO_RESET
13	SET_AFTER_AND_RESET_AFTER_ONE_CYCLE
15	SET_AFTER_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE
20	RESET_BEFORE
21	RESET_AFTER
22	RESET_AT_REMAINING_DISTANCE
25	NO_SET_AND_WAIT_FOR_FALSE_ONCE
26	NO_SET_AND_WAIT_FOR_TRUE_ONCE

In the following the different flag modes and their functionalities will be described in detail.

#### Flag mode 1: SET\_BEFORE\_AND\_NO\_RESET

Flag mode 1 sets a flag immediately when the defining command in the PathData is active. There will be no reset of the flag until the path ends, unless the flag is reset externally. [Figure 3-19](#) shows a timing diagram illustrating the behavior.

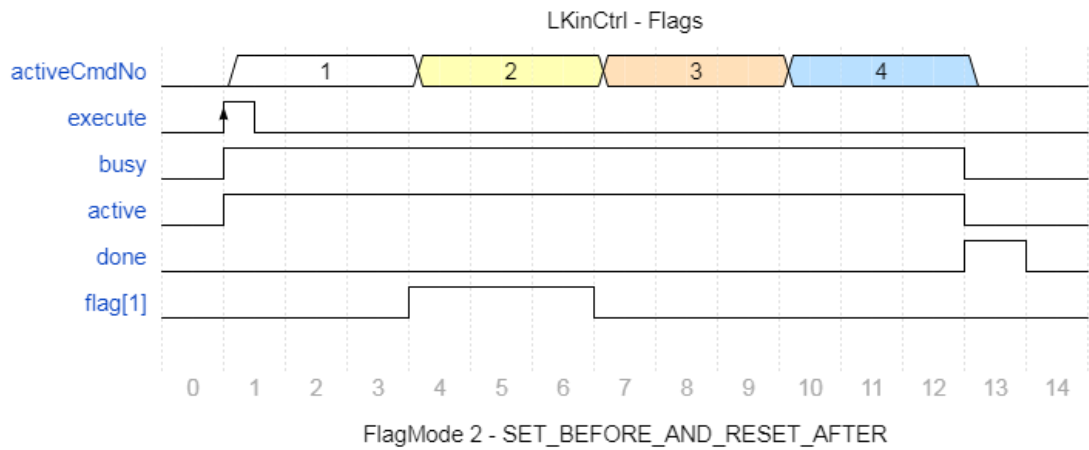
Figure 3-20: Timing Diagram – flag mode 1



**Flag mode 2: SET\_BEFORE\_AND\_RESET\_AFTER**

Flag mode 2 sets a flag immediately when the defining command in the PathData is active. The flag will be reset after the command is finished and the next command starts. [Figure 3-20](#) shows a timing diagram illustrating the behavior.

Figure 3-21: Timing Diagram – flag mode 2

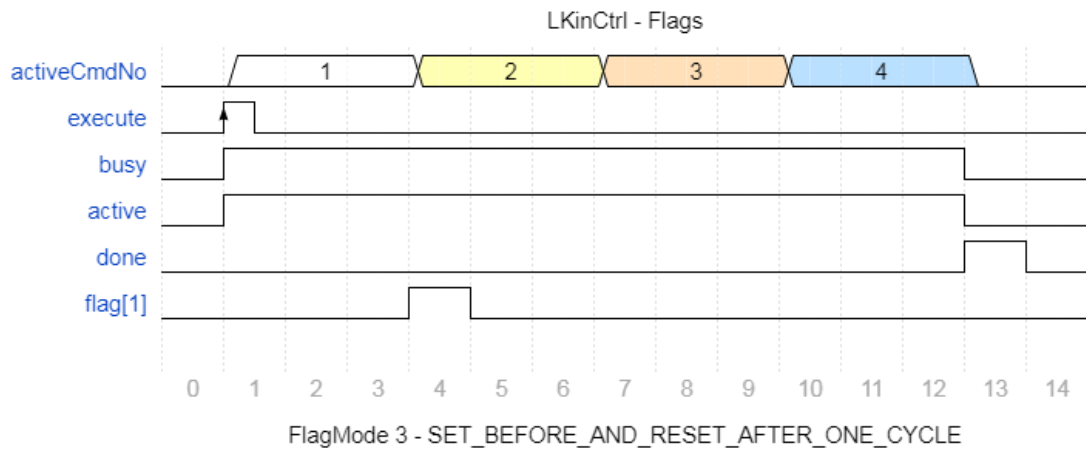


**Flag mode 3: SET\_BEFORE\_AND\_RESET\_AFTER\_ONE\_CYCLE**

Flag mode 3 sets a flag immediately when the defining command in the PathData is active. The flag will be reset after one cycle. [Figure 3-21](#) shows a timing diagram illustrating the behavior.

Figure 3-22: Timing Diagram – flag mode 3



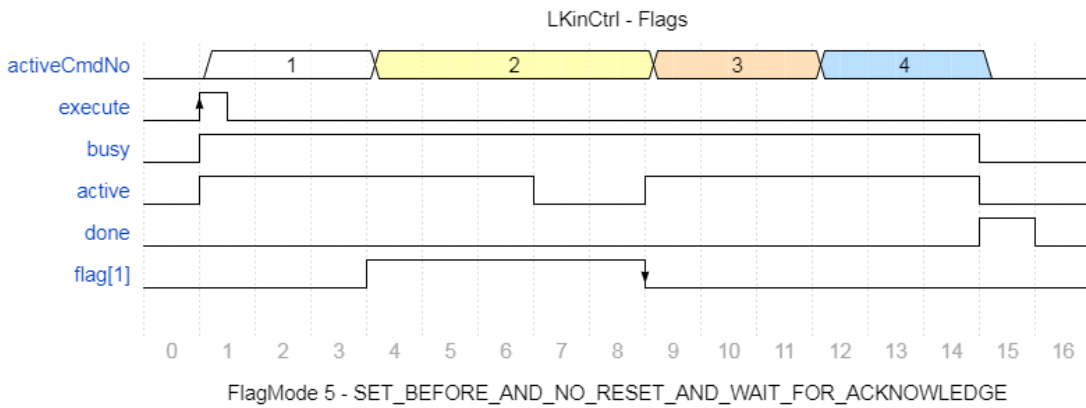


### Flag mode 5: SET\_BEFORE\_AND\_NO\_RESET\_AND\_WAIT\_FOR\_ACKNOWLEDGE

Flag mode 5 sets a flag immediately when the defining command in the PathData is active. There will be no reset, unless the flag is reset externally. The further path execution will wait after finishing the flag setting command. As soon as the flag is reset ('acknowledged'), the execution and therefore the path motion continues.

This flag mode can be used to create 'wait points' requiring external acknowledgement. [Figure 3-22](#) shows a timing diagram illustrating the behavior.

Figure 3-23: Timing Diagram – flag mode 5



#### NOTICE

##### Interlock of flags programmable

In combination with flagModes 11, 13 and 15 activated at the same command as a flag with flagMode 5, the once set flag can not be reset. Therefore, this combination should not be programmed to avoid an interlock of the specific flag.

#### NOTICE

##### Acknowledge of valueFlags

The acknowledgement of *valueFlags* can be performed by resetting the respective valueFlag to the default LREAL value of the flag (-1.0).

**Flag mode 10: SET\_IN\_REMAINING\_DISTANCE\_TO\_TARGET**

Flag mode 10 sets a flag in a specified distance to the commands target position. There will be no reset of the flag until the path ends, unless the flag is reset externally. [Figure 3-23](#) shows a trace diagram illustrating the behavior.

Figure 3-24: Trace timing diagram – flagMode 10

**NOTE**

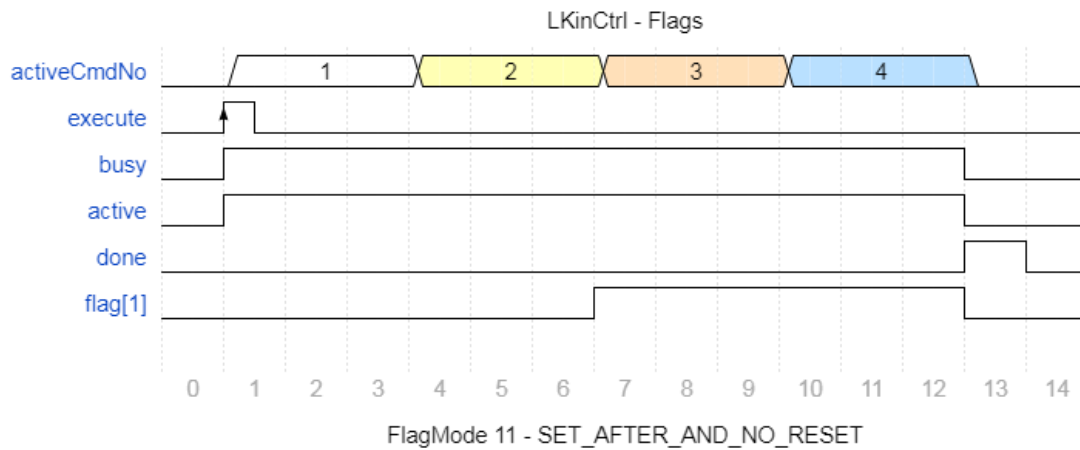
Please remember that the end (target position) of a command is defined, where the blending radius (defined as `transitionParameter[1]`) of the subsequent command starts.

Therefore, a reached `remainingDistance` of 0.0 does not always signalize the programmed target position being reached, but its blending radius.

**Flag mode 11: SET\_AFTER\_AND\_NO\_RESET**

Flag mode 11 sets a flag after the completion of the defining command in the PathData. Subsequently, the next command starts. There will be no reset of the flag until the path ends, unless the flag is reset externally. [Figure 3-24](#) shows a timing diagram illustrating the behavior.

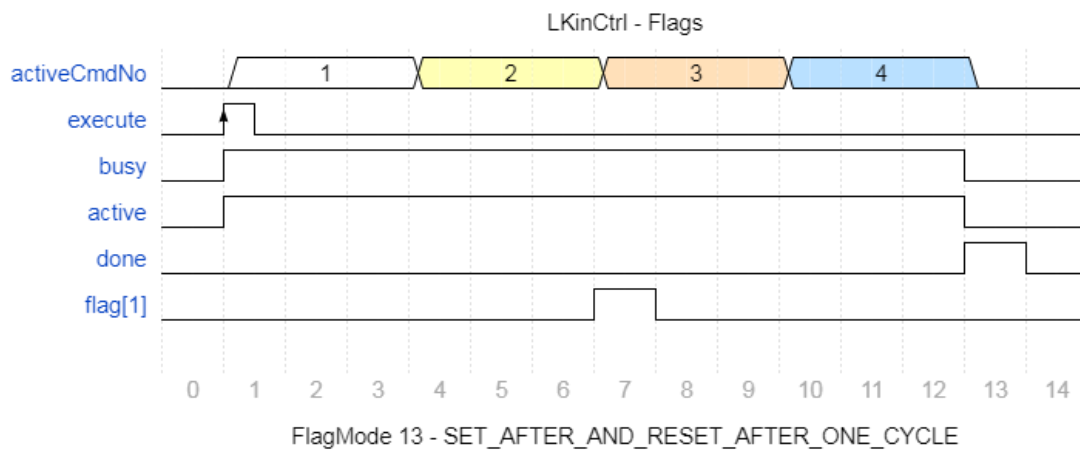
Figure 3-25: Timing Diagram – flag mode 11



### Flag mode 13: SET\_AFTER\_AND\_RESET\_AFTER\_ONE\_CYCLE

Flag mode 13 sets a flag after the completion of the defining command in the PathData (respectively when the next command starts). The flag will be reset after one cycle. [Figure 3-25](#) shows a timing diagram illustrating the behavior.

Figure 3-26: Timing Diagram – flag mode 13

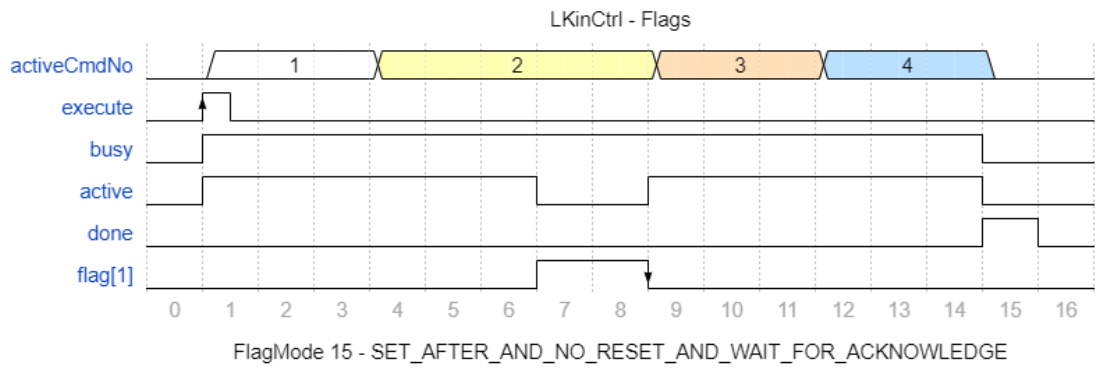


### Flag mode 15: SET\_AFTER\_AND\_NO\_RESET\_AND\_WAIT\_FOR\_ACKNOWLEDGE

Flag mode 15 sets a flag immediately after the defining command in the PathData is done respectively the next command starts. There will be no reset, unless the flag is reset externally. The further path execution will wait after finishing the flag setting command. As soon as the flag is reset ('acknowledged'), the execution and therefore the path motion continues.

This flag mode can be used to create 'wait points' requiring external acknowledgement. [Figure 3-26](#) shows a timing diagram illustrating the behavior.

Figure 3-27: Timing Diagram – flag mode 15



**NOTICE**

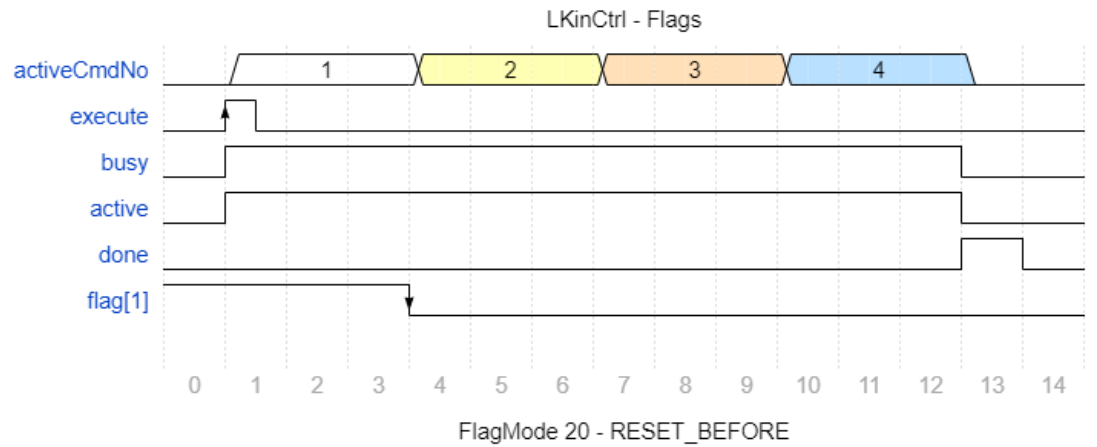
**Acknowledge of valueFlags**

The acknowledgement of *valueFlags* can be performed by resetting the respective valueFlag to the default LREAL value of the flag (-1.0).

**Flag mode 20: RESET\_BEFORE**

Flag mode 20 resets a flag immediately when the defining command in the PathData is active. [Figure 3-27](#) shows a timing diagram illustrating the behavior.

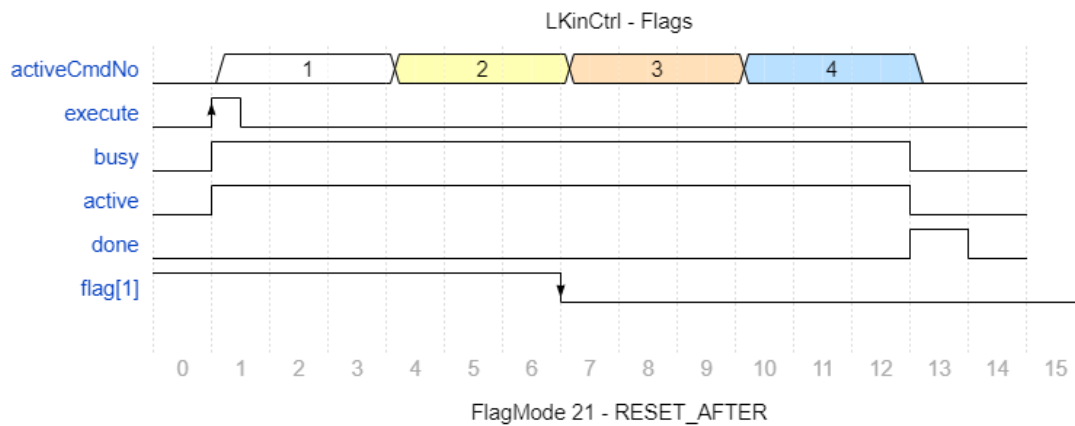
Figure 3-28: Timing Diagram – flag mode 20



**Flag mode 21: RESET\_AFTER**

Flag mode 21 resets a flag after the completion of the defining command in the PathData (respectively when the next command starts). [Figure 3-28](#) shows a timing diagram illustrating the behavior.

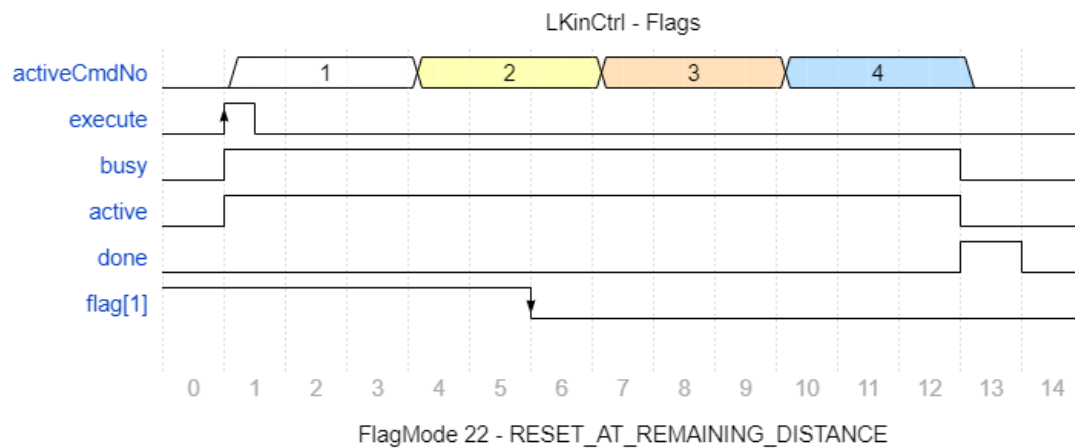
Figure 3-29: Timing Diagram – flag mode 21



### Flag mode 22: RESET\_AT\_REMAINING\_DISTANCE

Flag mode 22 resets a flag in a specified distance to the commands target position. [Figure 3-29](#) shows a timing diagram illustrating the behavior.

Figure 3-30: Timing Diagram – flag mode 22



#### NOTE

Please remember that the end (target position) of a command is defined, where the blending radius (defined as transitionParameter[1]) of the subsequent command starts.

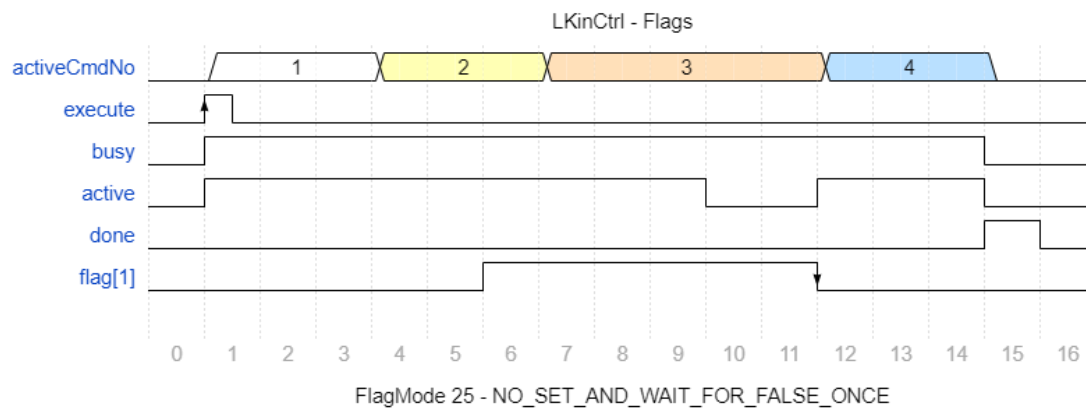
Therefore, a reached remainingDistance of 0.0 does not always signalize the programmed target position being reached, but its blending radius.

### Flag mode 25: NO\_SET\_AND\_WAIT\_FOR\_FALSE\_ONCE

Flag mode 25 does not set or reset a flag. It will wait until the flag at the given flag number is false once. The further path execution will wait after finishing the command. As soon as the flag status is false, the execution and therefore the path motion continues.

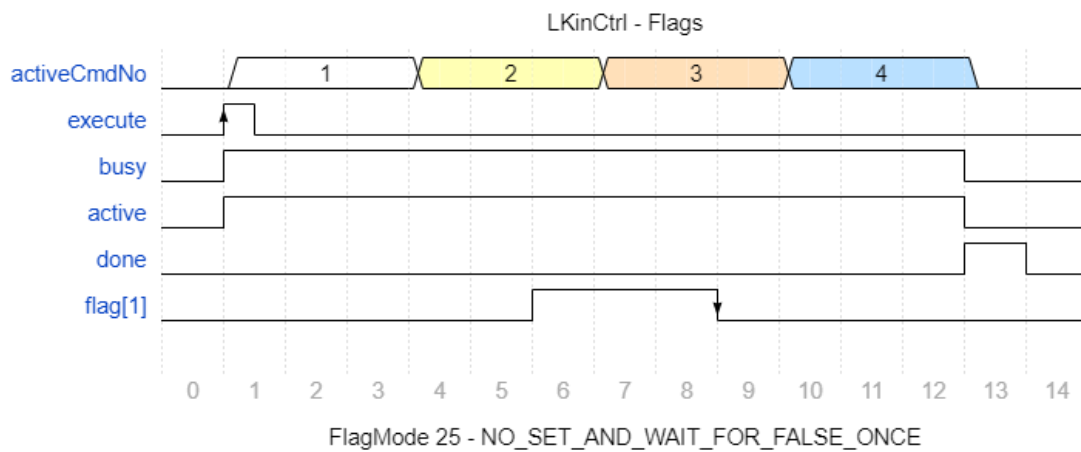
This flag mode can be used to create 'wait points' requiring external acknowledgement. [Figure 3-30](#) shows a timing diagram illustrating the behavior.

Figure 3-31: Timing Diagram – flag mode 25



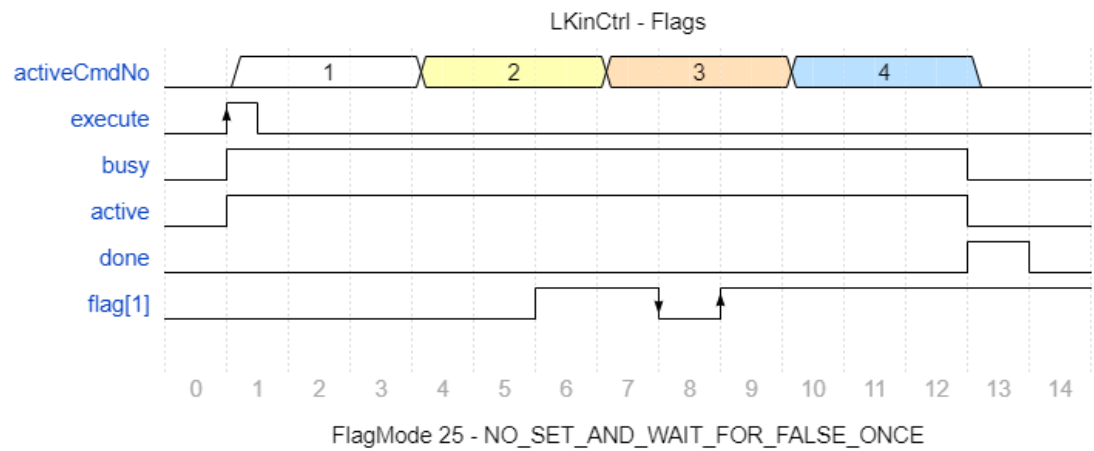
It is also possible to reset the flag during the motion. The flag is then acknowledged early, and the further path execution is not halted at the end of the command.

Figure 3-32: Timing Diagram – flag mode 25 – early reset



If a flag is acknowledged once it is not reactivated again. So, toggling the flag will not halt the execution at the end of the command again.

Figure 3-33: Timing Diagram – flag mode 25 – toggle



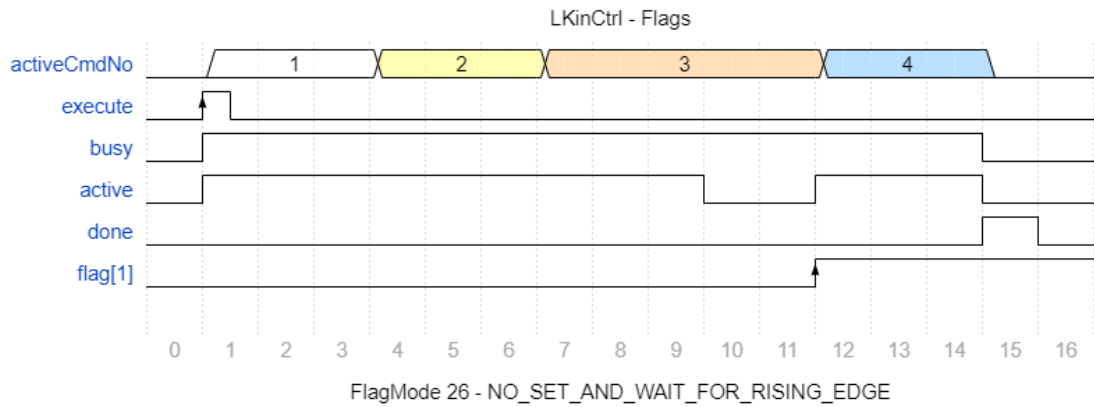
**NOTICE**      **Acknowledge of valueFlags**  
 The acknowledgement of *valueFlags* can be performed by resetting the respective valueFlag to the default LREAL value of the flag (-1.0).

**Flag mode 26: NO\_SET\_AND\_WAIT\_FOR\_TRUE\_ONCE**

Flag mode 26 does not set or reset a flag. It will wait until the flag at the given flag number is true once. The further path execution will wait after finishing the command. As soon as the flag status is true, the execution and therefore the path motion continues.

This flag mode can be used to create ‘wait points’ requiring external acknowledgement. [Figure 3-33](#) shows a timing diagram illustrating the behavior.

Figure 3-34: Timing Diagram – flag mode 26



**NOTE**      For acknowledging the flag early or toggling the flag, the same rules apply as shown for flagMode 25, but with inverted logic.

**NOTICE**

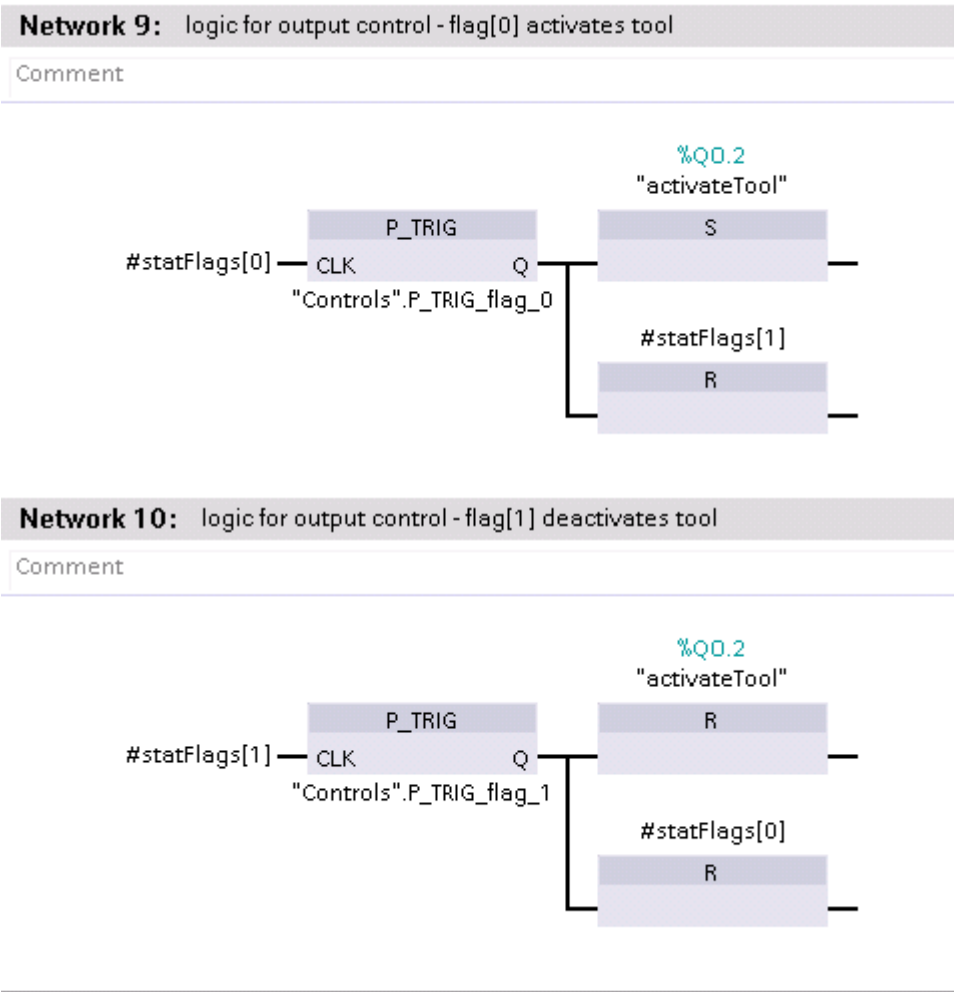
**Acknowledge of valueFlags**

The acknowledgement of *valueFlags* can be performed by setting the respective valueFlag to a value which is **not** the default LREAL value of the flag (-1.0).

**External reset of flags**

As *setFlags* in certain flag modes (like flag mode 1) only set *flags[0...n]* at the MC\_MovePath interface to TRUE, the activation or deactivation of them has to reset the other one and vice versa. [Figure 3-34](#) shows an example of how to use two flags to set and reset a Boolean actuator by simple Boolean logic.

Figure 3-35: Logic for setting and resetting actuators via flags



The example code in the figure above shows how two setFlags control a Boolean output signal. In this case the static variable *statFlags[0...n]* is attached to the flags InOut parameter of the *LKinCtrl\_MC\_MovePath* interface and contains the flag signals. To use two flags for setting and resetting one actuator Boolean the flags need to set and reset as well the actuator itself as the corresponding flag.

**Internal reset of flags**

Alternatively, to the external reset of the *setFlags* can also be reset by combinations of the previously described flag modes. Using the *flagModes* in combination to reset a specific flag, the flag will be TRUE for at least two commands.



**NOTICE****Automatic reset of flags at completion of path motion**

When the final target position programmed in the PathData is reached, the FB LKinCtrl\_MC\_MovePath will return a *done* signal. At the same time all flags will be automatically reset.

If any actuators controlled by flags need to remain set after the path motion is finished, that logic needs to be programmed externally.

**NOTICE****Automatic reset of flags when triggering a new path motion**

When triggering a new path motion with a rising edge at *execute*, all flags will be reset automatically.

Flags will remain set after a *stop* of the path motion. Therefore, the eventually controlled actuators will remain set when stopping the path motion during execution (e.g. vacuum gripper does not let go of the product when stopped).

### 3.11. FlagOnly commands

To use the flag functionality described in chapter [3.9](#) without programming a motion command, the command type of a FlagOnly command can be used.

To program a FlagOnly command, the cmdType entry in the PathData command needs to be set to 0 as shown in the image below.

Figure 3-36: Definition of a FlagOnly command in PathData DB

■	▼	commands[10]	"LKInCtrl_typePathDataElement"	
■		cmdType	Int	0
■		cmdActivated	Bool	TRUE
■		cmdName	String["LKINCTRL_LENGTH_OF_CMD..."]	"
■	▶	cmdCoordinates	"LKInCtrl_typePointCoordinates"	
■	▶	cmdParameters	"LKInCtrl_typePointParameter"	
■	▼	setFlags	Array[1.."LKINCTRL_NO_OF_CMD_S..."]	
	■	▼	setFlags[1]	"LKInCtrl_typeSetFlagElement_Bool"
		■	flag	Int
		■	flagMode	USInt
		■	remainingDistance	LReal
	■	▶	setFlags[2]	"LKInCtrl_typeSetFlagElement_Bool"
	■	▶	setFlags[3]	"LKInCtrl_typeSetFlagElement_Bool"
■	▼	valueFlags	Array[1.."LKINCTRL_NO_OF_CMD_V..."]	
	■	▼	valueFlags[1]	"LKInCtrl_typeSetFlagElement_LRE..."
		■	flag	Int
		■	value	LReal
		■	flagMode	USInt
		■	remainingDistance	LReal
	■	▶	valueFlags[2]	"LKInCtrl_typeSetFlagElement_LRE..."
	■	▶	valueFlags[3]	"LKInCtrl_typeSetFlagElement_LRE..."
	■	▶	valueFlags[4]	"LKInCtrl_typeSetFlagElement_LRE..."

#### NOTE

All flag modes are supported for FlagOnly commands, although many of the modes will work very similar, as there is no motion command to wait for to be finished.

Acknowledgement modes work by means of interrupting the execution of further commands. Therefore, interim points in the path can be programmed this way.

### 3.12. Wait times

Wait times can be programmed by a specific command type in the PathData structure. Configuring the `cmdType = 100` for a PathData command enables a wait time after the previous command has been completed and before the following command is about to be executed.

The wait time duration of the command needs to be written to the parameter `cmdparameters.pathDynamics.velocity` in milliseconds [ms].

An exemplary definition of a wait time command is presented in [Figure 3-36](#).

Figure 3-37: Definition of wait time command in PathData DB

■	▼	commands[6]	"LKInCtrl_typePath..."	
■		cmdType	Int	100
■		cmdActivated	Bool	TRUE
■		cmdName	String["LKINCTRL_L..."	"
■	▶	cmdCoordinates	"LKInCtrl_typePoint..."	
■	▼	cmdParameters	"LKInCtrl_typePoint..."	
■	▼	pathDynamics	"LKInCtrl_typePath..."	
■		velocity	LReal	3000.0
■		acceleration	LReal	-1.0
■		deceleration	LReal	-1.0
■		jerk	LReal	-1.0
■		orientationDirection	DInt	3
■		bufferMode	DInt	1
■	▶	transitionParameters	Array[1..5] of LReal	
■		dynamicAdaption	DInt	-1
■	▶	circleParameters	"LKInCtrl_typeCircl..."	
■	▶	setFlags	Array[1.."LKINCTRL_..."	

During the elapsing wait time the output `LKinCtrl_MC_MovePath.activeCmdNo` will show the index of the command in the PathData array. The output `LKinCtrl_MC_MovePath.remainingDistanceActCmd` will show the remaining wait time in milliseconds [ms].

#### NOTE

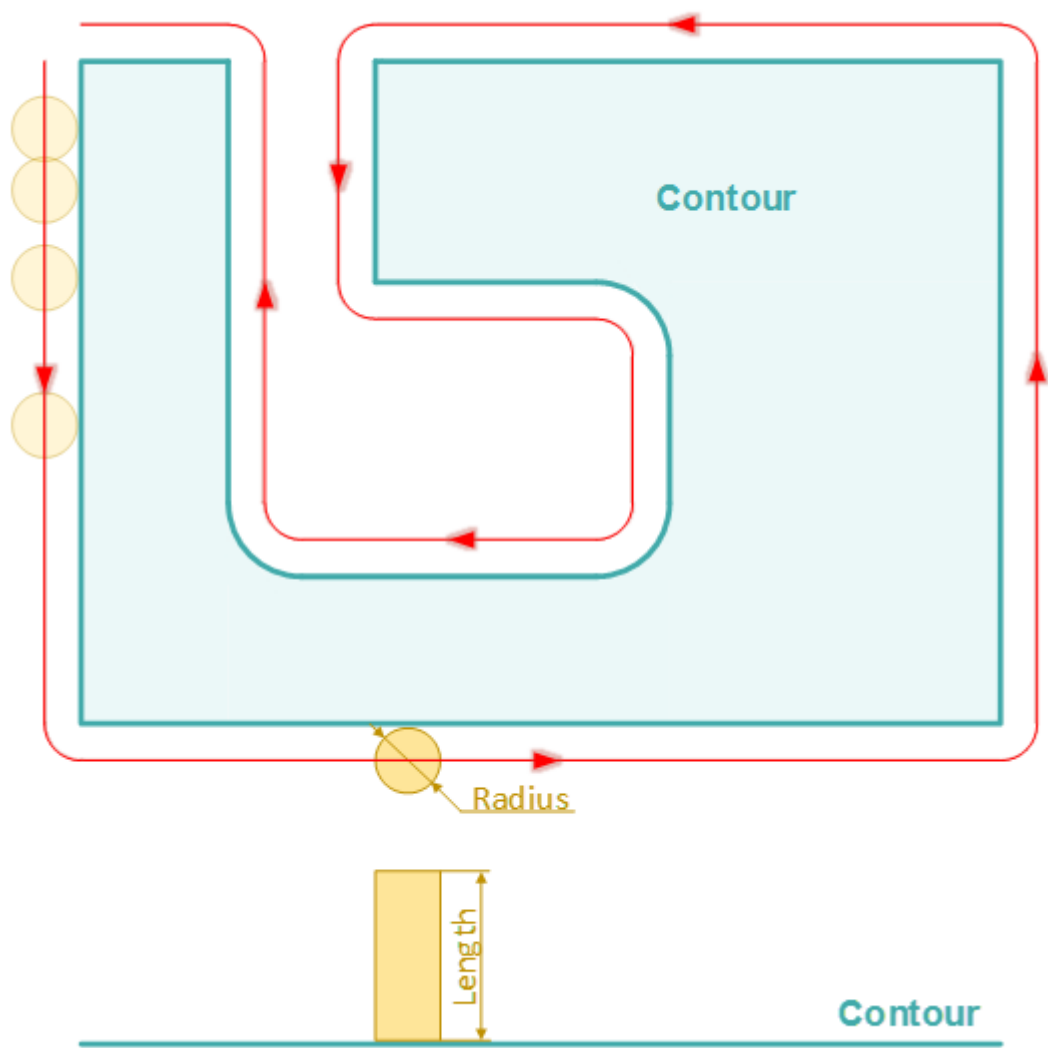
When defining a wait time command, the execution of all PathData commands will stop after the wait time command until the wait time is elapsed.

This will result in the MotionQueue running empty and the path motion to stop (velocity = 0.0) during the wait time.

### 3.13. Contour offset and radius compensation

For some use cases (e.g. like grinding) it is necessary to follow the path of a contour exactly, but in a specific distance given by the dimensions of a tool like its radius and length. [Figure 3-37](#) shows an exemplary overview of the described use case for contour offset and radius compensation.

Figure 3-38: Overview of contour offset and radius compensation



The red line in [Figure 3-37](#) follows the main contour in distance of the *radius* on one side (left or right compensation selectable) and in a height *length* of the actual contour plane.

To traverse a path like the red line, the PathData will be filled with path information defining the main contour. One command before the main contour definition in the PathData begins the compensation will be activated and parametrized by a PathData command of *cmdType*: 41 or *cmdType*: 42, depending on the compensation direction. The end of the compensation needs to be signaled by a PathData command of *cmdType*: 40. The constants in the table below are provided by the LKinCtrl\_Tags tag table to support the user to easily define the *cmdType* in FBD, LAD and SCL editors.

Table 3-8: PathData configuration command assignment: *cmdType*

PathData configuration command	Value
LKINCTRL_CONFIG_END_OFFSET_COMP	40
LKINCTRL_CONFIG_START_OFFSET_COMP_LEFT	41
LKINCTRL_CONFIG_START_OFFSET_COMP_RIGHT	42

To parametrize the offset compensation the parameters have to be inserted in the PathData structure *cmdParameters.offsetParameters* and the *configuration* structure at the *LKinCtrl\_MC\_MovePath* interface according to the assignment and codification in [Table 3-10](#):

Table 3-9: Offset compensation parameter assignment and codification: PathData

PathData parameter	Offset compensation parameter
configuration.offsetParameter[n].name	Tool name
configuration.offsetParameter[n].radius	Tool radius [in mm]
configuration.offsetParameter [n].length	Tool length [in mm]
PathData.commands[i].cmdParameters.offsetParameters[n].offSetParaNo	Selected offset parameter set (defined in configuration structure)
PathData.commands[i].cmdParameters.offsetParameters[n].mainPathPlane	Path plane of radius compensation 0: CP_XZ 1: CP_YZ 2: CP_XY Constants in MC_Constants tag table

### 3.14. Relative shift of object coordinate systems

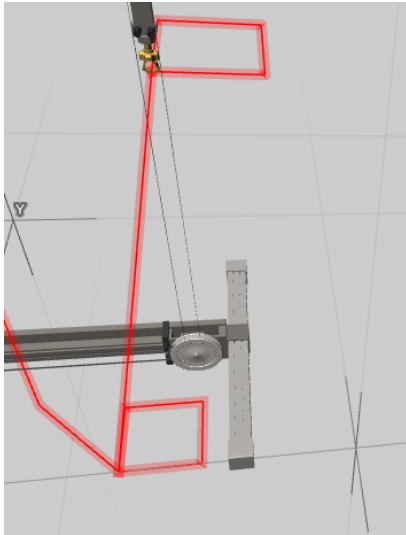
With *cmdType* 60 you can execute a *MC\_Shift\_CS\_Rel* command. To repeat certain steps while processing, the object coordinate systems can be shifted in a translatory and rotary manner. The shifts are assigned to the cartesian coordinates of the path data and later written onto shift coordinates.

The shift is always valid for the parameterized coordinate system.

There can be only one shift at a time. If a new coordinate system is shifted then the previous shifts are reset.

If the same coordinate system is set then multiple shifts are be stacked.

Figure 3-39: Relative shift of squares



The relative shift works for following command types:

- LKINCTRL\_MC\_MOVELINEARABS
- LKINCTRL\_MC\_MOVEDIRECTABS
- LKINCTRL\_MC\_MOVECIRCULARABS
- LKINCTRL\_MC\_MOVELINEARPICKANDPLACE
- LKINCTRL\_MC\_MOVELINEARREL
- LKINCTRL\_MC\_MOVEDIRECTREL
- LKINCTRL\_MC\_MOVECIRCULARREL

#### NOTE

When using a circular command directly after shifting the OCS, insert a linear command to the origin of the coordinate system first!

## Example

Figure 3-40: Example path data

```
#statPathData.commands[#i].cmdType := "LKINCTRL_MC_SHIFT_CS_REL";
#statPathData.commands[#i].cmdCoordinates.coordSystem := "CS_OCS1";
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.x := 100.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.y := 100.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.z := 500.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.a := 0.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.b := 0.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.c := 0.0;
#i := #i + 1;

#statPathData.commands[#i].cmdType := "LKINCTRL_MC_MOVELINEARABS";
#statPathData.commands[#i].cmdCoordinates.coordSystem := "CS_OCS1";
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.x := 0.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.y := 0.0;
#statPathData.commands[#i].cmdCoordinates.cartesianPosition.z := 0.0;
#i := #i + 1;
```

The relative shift is set back implicitly by changing between different coordinate systems. If a certain coordinate system should be shifted again, the shift must be actively initiated by the user. Alternatively, the shift can be set back explicitly by inserting a command of *cmdType* 61 "MC\_Reset\_Shift". No additional parameters must be added.

Figure 3-41: Example of command call

```
#statPathData.commands[#i].cmdType := "LKINCTRL_MC_RESET_SHIFT";
#i := #i + 1;
```

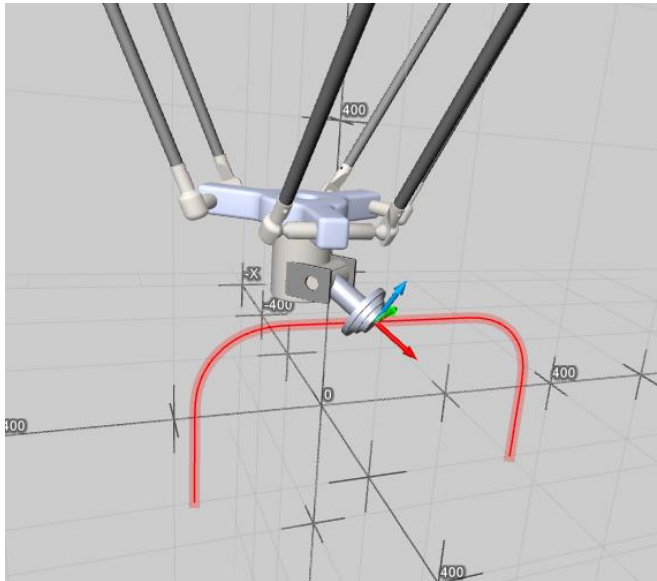
### 3.15. Pick and place sequence

#### NOTE

This section describes the MC\_MovePath PathData command parametrization. For more information about the pick and place command itself, refer to chapter [4.5](#).

`cmdType = 11` (LKINCTRL\_MC\_MOVELINEARPICKANDPLACE) is executing a linear motion sequence with optional conveyor tracking according to the FB LKinCtrl\_MC\_MovePickAndPlaceLinear.

Figure 3-42: Typical pick and place sequence



General command parameter (Buffer Mode, Transition Parameter, Dynamics etc.) can be parametrized in the respective PathData parameter. Additional parameters are start vector, target vector and conveyor tracking settings. The parametrization of start position and start coordinate system is done automatically by MC\_MovePath.

#### Start and target vector

The parametrization for start and target vector can be found within `cmdParameters.pickAndPlaceParameters`. All rules from chapter [4.5](#) apply.

#### Conveyor tracking

Conveyor tracking will be automatically established by setting the conveyor parametrization bigger than 0 (`cmdParameters.conveyorParameters.conveyorParaNo > 0`). The coupled OCS must be specified in `cmdCoordinates.coordSystem`. No additional command is necessary for tracking. For more information about conveyor tracking parametrization, refer to [3.6](#).

#### Example

[Figure 3-43](#) is showing an example path data command with automatic conveyor tracking (`cmdParameters.conveyorParaNo = 2`, `cmdCoordinates.coordSystem = OCS1`). Both start and target vector are present with blending and dynamics reduction.

Figure 3-43: Example path data



```

// Pick and place command to place position
#statPathData.commands[#i].cmdType := "LKINCTRL_MC_MOVELINEARPICKANDPLACE";
// Tracking
#statPathData.commands[#i].cmdParameters.conveyorParameters.conveyorParaNo := 2; // Use conveyor parametrization #2
// Command coordSystem and target position
#statPathData.commands[#i].cmdCoordinates.coordSystem := "CS_OCS1"; // Assign OCS1 to be tracked
#statPathData.commands[#i].cmdCoordinates.cartesianPosition := #statPlacePosition; // Assign target position
// Buffer mode and dynamics setting
#statPathData.commands[#i].cmdParameters.bufferMode := "BM_BUFFERED"; // Buffer mode
#statPathData.commands[#i].cmdParameters.pathDynamics.velocity := 1000.0; // set path velocity
// Start vector parameters
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.direction[1] := 0.0;
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.direction[2] := 0.0;
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.direction[3] := 1.0; // depart in Z-direction
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.workingHeight := 60.0; // 60mm working height
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.transitionArea := 10.0; // 10mm blending radius
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.startParameter.dynamicsFactor := 50.0; // 50% dynamics reduction
// Target vector parameters
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.direction[1] := 0.0;
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.direction[2] := 0.0;
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.direction[3] := 1.0; // approach in Z-direction
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.workingHeight := 100.0; // 100mm working height
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.transitionArea := 50.0; // 50mm blending radius
#statPathData.commands[#i].cmdParameters.pickAndPlaceParameters.targetParameter.dynamicsFactor := 80.0; // 80% dynamics reduction

```

### 3.16. Measurement command

The measurement command can be used when a motion command must be aborted without aborting the whole path. A typical use case is a measurement run onto a pallet to see if there is any product on it. The abortion takes place via an input that can be triggered by a sensor or be manually activated.

#### NOTE

The measurement command functionality is provided via a parameter in the PathData and is not a standalone command! Blending after a measurement command is not possible!

#### Mode configuration

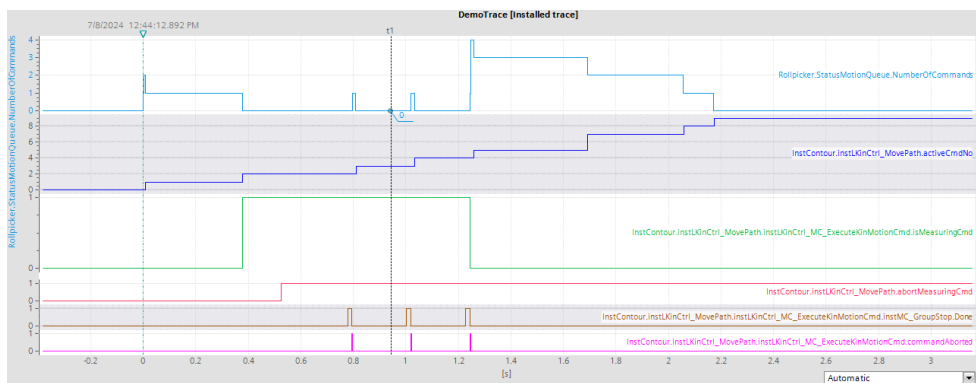
The measurement command functionality is by default disabled (mode = 0) and must be enabled for every single desired motion command.

When the command must be aborted (e.g. because a distance sensor is triggered) and the following commands should be executed without acknowledging the abortion, mode 1 has to be selected. Contrary to mode 2, where the user must acknowledge the abortion and the movement will only continue after that.

#### Timing and behavior

Case 1: Multiple measurement commands (mode = 1) in a row and the abort input is triggered (no falling edge)

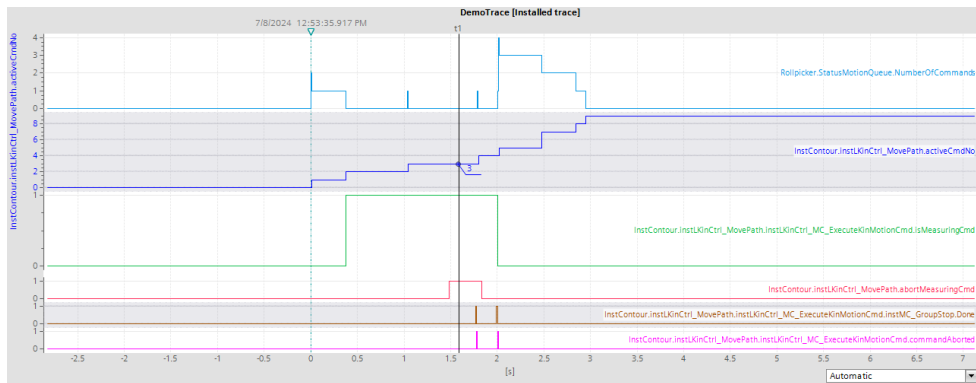
Figure 3-44: Trace 1



Since there is no acknowledge needed, command two is aborted and the motion immediately continues with command three. As command three is also a measurement command and the abort input is still triggered, this command is also aborted.

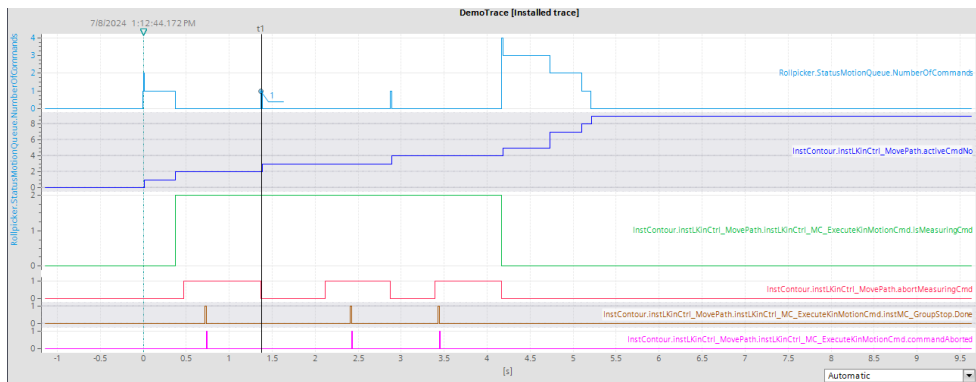
Case 2: Multiple measurement commands (mode = 1) in a row and the abort input is reset

Figure 3-45: Trace 2



Case 3: Multiple measurement commands (mode = 2) in a row

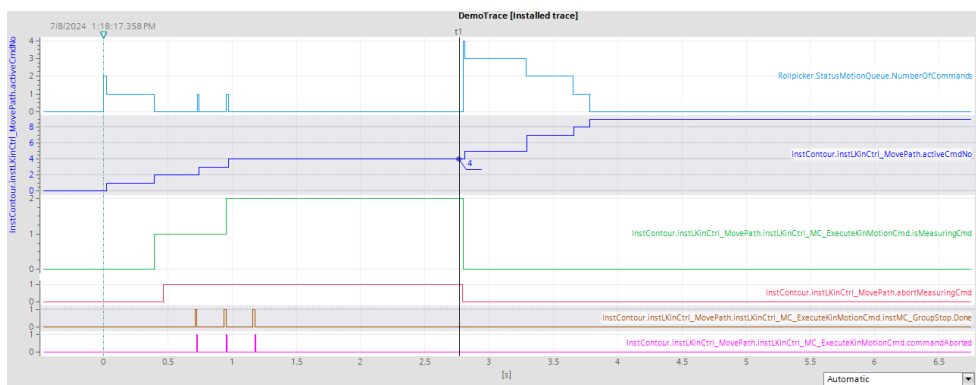
Figure 3-46: Trace 3



A stop is executed as soon as there is a rising edge on the abort input. The command is aborted successfully but the motion program continues not until the abort input is reset.

Case 4: Multiple measurement commands of mode 1 and one single measurement command of mode 2

Figure 3-47: Trace 4



The triggering of the abort input leads to an abortion of all of the measurement commands from which two are of mode 1 and the last one of mode 2. But the execution will only continue when the last abortion has been acknowledged.

### Example

Commands within the PathData can be marked as measurement commands via configuring the parameter „measuringCmd“. This can be either done directly in the PathData or via the HMI Interface.

Figure 3-48: Configuration in the PathData

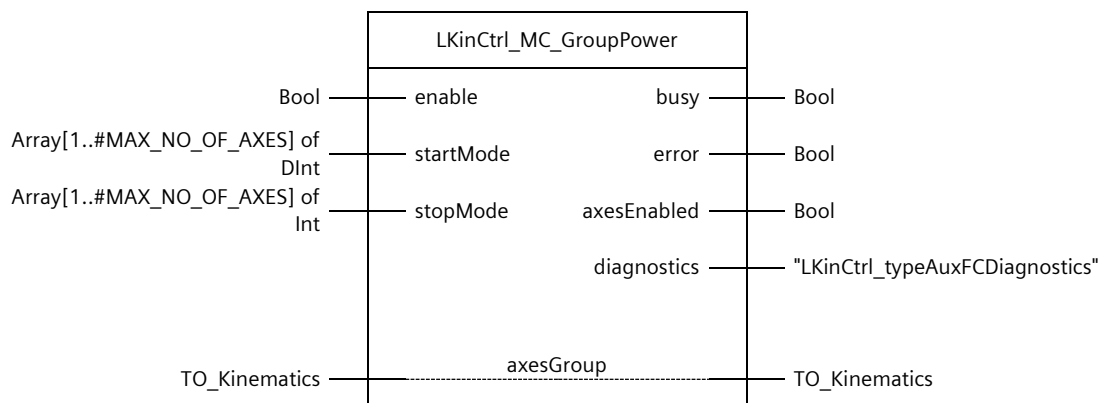
▼ commands	Array[1..*LKINCTRL...			
▼ commands[1]	"LKinCtrl_typePath...			
cmdType	Int	-1	1	
cmdActivated	Bool	TRUE	TRUE	
cmdName	String["LKINCTRL_L...	"	'B_up'	
point	DInt	-1	-1	
measuringCmd	USInt	0	2	
▶ cmdCoordinates	"LKinCtrl_typePoint...			
▶ cmdParameters	"LKinCtrl_typePoint...			
▶ setFlags	Array[1..*LKINCTRL...			
▶ valueFlags	Array[1..*LKINCTRL...			

## 4. Standalone functions

### 4.1. FB LKinCtrl\_MC\_GroupPower (FB 35021)

The FB *LKinCtrl\_MC\_GroupPower* enables or disables all axes interconnected to the attached *axesGroup*. Equal to the interface of *MC\_Power* the *startMode* and *stopMode* can be specified according to the settings in the comments.

Figure 4-1: LKinCtrl\_MC\_GroupPower



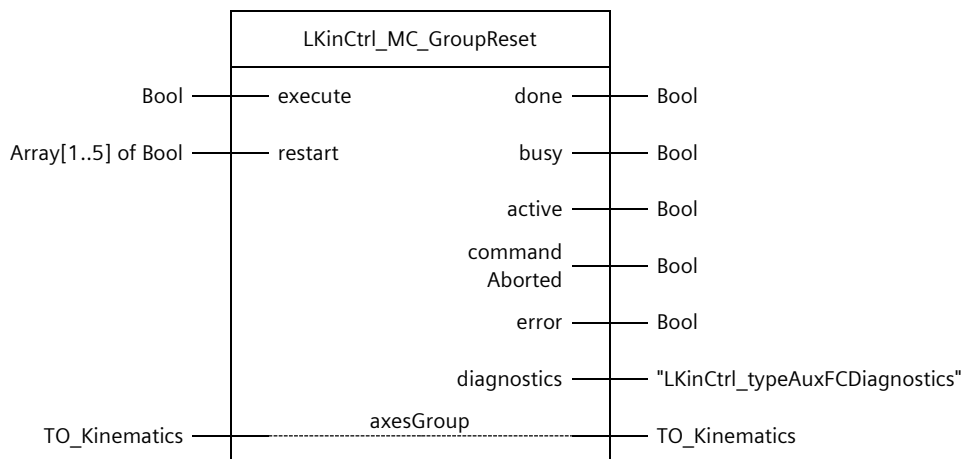
4-1: Parameter of LKinCtrl\_MC\_GroupPower

Name	P-Type	Data Type	Comment
enable	IN	Bool	TRUE: Enable axes; FALSE: Disable axes
startMode	IN	Array[1..#MAX_NO_OF_AXES] of DInt	startMode of axis; 0: enable NOT position controlled; 1: enable position controlled
stopMode	IN	Array[1..#MAX_NO_OF_AXES] of Int	stopMode of axis; 0: emergency stop; 1: immediate stop; 2: stop with max dynamics
busy	OUT	Bool	TRUE: FB is not finished and new output values can be expected
error	OUT	Bool	TRUE: An error occurred during the execution of the FB
axesEnabled	OUT	Bool	TRUE: all axes of axesGroup enabled; FALSE: all axes of axesGroup disabled
diagnostics	OUT	"LKinCtrl_typeAuxFCDiagnostics"	Diagnostics information of FB (optional)
axesGroup	IN_OUT	TO_Kinematics	reference to the axesGroup

## 4.2. FB LKinCtrl\_MC\_GroupReset (FB 35023)

The FB *LKinCtrl\_MC\_GroupReset* resets all TO axes interconnected to the attached *axesGroup* and the *TO\_Kinematics* itself. Equal to the interface of *MC\_Reset* the *restart* mode can be specified according to the settings in the comments.

Figure 4-2: LKinCtrl\_MC\_GroupReset



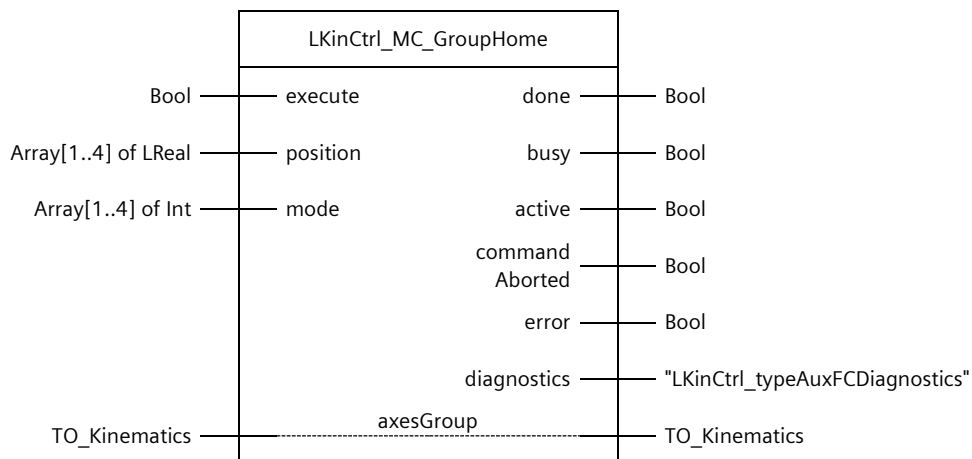
4-2: Parameter of LKinCtrl\_MC\_GroupReset

Name	P-Type	Data Type	Comment
execute	IN	Bool	Rising edge starts action once
restart	IN	Array[1..7] of Bool	reinitialization of technology object TRUE: with restart FALSE: without restart
done	OUT	Bool	axesGroup successfully referenced
busy	OUT	Bool	FB is not finished and new output values can be expected
active	OUT	Bool	FB in control of axesGroup / kinematic in motion
commandAborted	OUT	Bool	command is aborted by another command
error	OUT	Bool	an error occurred during the execution of the FB
diagnostics	OUT	"LKinCtrl_typeAuxFCDiagnostics"	Diagnostics information of FB (optional)
axesGroup	IN_OUT	TO_Kinematics	reference to the axesGroup

### 4.3. FB LKinCtrl\_MC\_GroupHome (FB 35022)

The FB *LKinCtrl\_MC\_GroupHome* references all TO axes interconnected to the attached *axesGroup*. Equal to the interface of *MC\_Home* the referencing *mode* can be specified equally to *MC\_Home*.

Figure 4-3: LKinCtrl\_MC\_GroupHome

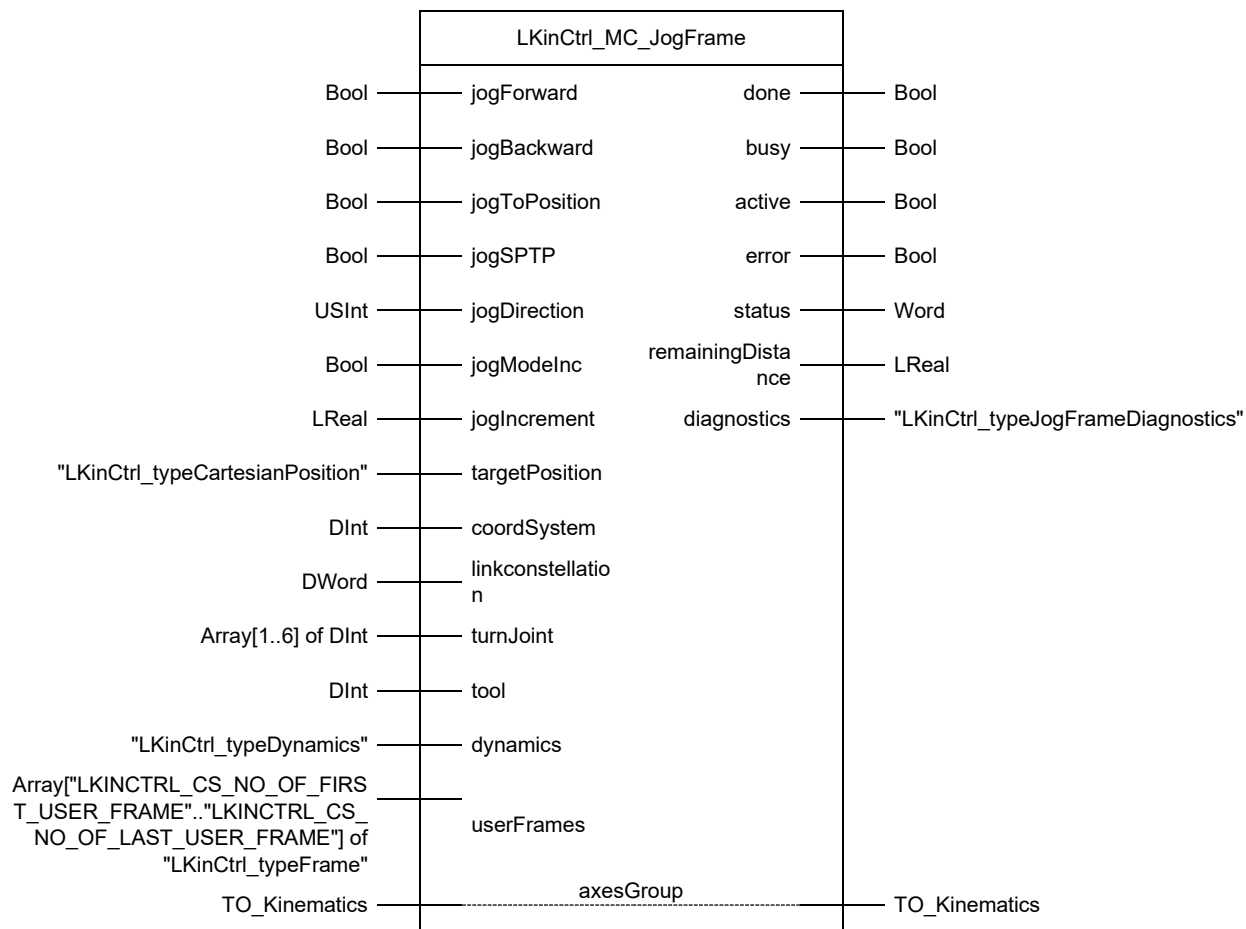


4-3: Parameter of LKinCtrl\_MC\_GroupHome

Name	P-Type	Data Type	Comment
execute	IN	Bool	Rising edge starts action once
position	IN	Array[1..#MAX_NO_OF_AXES] of LReal	reference positions / orientations
mode	IN	Array[1..#MAX_NO_OF_AXES] of Int	homing modes for each axis
done	OUT	Bool	axesGroup successfully referenced
busy	OUT	Bool	FB is not finished and new output values can be expected
active	OUT	Bool	FB in control of axesGroup / kinematic in motion
commandAborted	OUT	Bool	command is aborted by another command
error	OUT	Bool	an error occurred during the execution of the FB
diagnostics	OUT	"LKinCtrl_typeAuxFCDiagnostics"	Diagnostics information of FB (optional)
axesGroup	IN_OUT	TO_Kinematics	reference to the axesGroup

## 4.4. FB LKinCtrl\_MC\_JogFrame (FB 35010)

Figure 4-4: LKinCtrl\_MC\_JogFrame



### Principle of operation

The function block LKinCtrl\_MC\_JogFrame provides all the functionality to jog a kinematics object in Cartesian directions X, Y, Z as well as to jog the orientation axis of the kinematics. Furthermore, the positioning of the kinematics in the Cartesian space to a target position including the orientation axis is supported.

The function block has been created with "execute" inputs for the jog commands "jogForward", "jogBackward" and "jogToPosition". As soon as a rising edge is applied to these inputs, the function block begins with the jog operation. With a falling edge at the input, the movement is stopped with a stop command.

It is possible to jog the kinematics in different coordinate systems. To do this, the corresponding selection of the coordinate system must be made at the "coordSystem" input. The same applies to the selection of the tool to be used for jogging. This is selected at the "tool" input. The dynamic parameters for jogging can be defined at the "dynamics" input. If the input is not configured with parameters, the default dynamics of the kinematics technology object are used.

When the function block is called, the "busy" output will be set. An active movement of the kinematics is indicated at output "active". The successfully completed jog command is displayed at output "done". An error while jogging is indicated at the "error" output as long as a jog command is set for at least one program cycle. Further details on the error can be found in the "diagnostics" output structure. The current status of the function block is displayed at output "status". The remaining jog distance is shown at the output "remainingDistanceActCmd" while jogging the kinematics with incremental mode or jogging to a target position.

### Continuous jog mode

The kinematic movement is started in continuous mode with an edge at the "jogForward" or "jogBackward" inputs. The movement is only stopped by a falling edge at the same input or by an error at the technology object.



**NOTE**

When jogging in MCS or JCS, the respective limits are considered.  
The movement is stopped at the configured joint limit or software limit switch.

**Incremental jog mode**

Incremental jog mode can be selected at the "jogModeInc" input. The distance to be traversed must be defined at the "jogIncrement" input. The movement can then be started with a positive edge at the "jogForward" or "jogBackward" inputs. After reaching the defined distance, the movement will be stopped. A falling edge at the jog input stops the movement with a stop command, even if the increment has not been completed.

**Jog to a target position**

To jog the kinematics to a target position, the "jogToPosition" input must be set. A target position must first be defined at the "targetPosition" input. In order to move, the input must remain set. The kinematics will stop when the target position is reached. With a falling edge, the movement is stopped even if the target position has not yet been reached.

**Jog to target position via synchronous Point-to-Point movement**

To jog via an sPTP movement, the "jogSPTP" input must be set. A target position must first be defined at the "targetPosition" input. In order to move, the input must remain set. The kinematics will stop when the target position is reached. With a falling edge, the movement is stopped even if the target position has not yet been reached. If the target position is in the world or object coordinate system, the link constellation can also be regarded with the input "linkConstellation". If the coordinate system input is set to 100, the machine coordinate system is used. The target position is interpreted as axes values instead of cartesian.

**Supported functionalities:**

- Jog kinematics in X, Y, Z, A, B and C direction (incremental / continuously)
- Jog kinematics to a target position
- Specification of the tool (tool1...3)
- Specification of the jog coordinate system (WCS, OCS1...3, MCS, JCS, UCS)
- Specification of the jog velocity (path / orientation)

**NOTICE****Jog command is not set during motion job in progress**

Jog commands triggered during another kinematics movement will not be executed.

**NOTICE****Simultaneous setting of several jog inputs**

If more than one input is set simultaneously, no movement command will be executed, and an ongoing movement will be stopped.

## Interface parameters

Table 2-4: Parameter of LKinCtrl\_MC\_JogFrame

Name	P-Type	Data Type	Comment
jogForward	IN	Bool	Jog the kinematics forward in the selected jog direction
jogBackward	IN	Bool	Jog the kinematics backward in the selected jog direction
jogToPosition	IN	Bool	Jog the kinematics to the specified target position
jogSPTP	IN	Bool	Jog the kinematics synchronous point to point
jogDirection	IN	USInt	1=x-direction, 2=y-direction, 3=z-direction, 4=a-direction, 5=b-direction, 6=c-direction, 7=Z-axis - about A,, 8=Y-axis - about B, 9=X-axis - about C
jogModelInc	IN	Bool	1=incremental jog mode, 0=continuous jog mode
jogIncrement	IN	LReal	Specification of the jog increment
targetPosition	IN	"LKinCtrl_typeCartesianPosition"	Specification of the target position
coordSystem	IN	DInt	0=WCS, 1= OCS1, 2=OCS2, 3=OCS3, 4=TCS, 100=MCS, 101=JCS (only KinPlus)
linkconstellation	IN	DWord	Target joint position space
tool	IN	DInt	1= tool1, 2=tool2, 3=tool3
dynamics	IN	"LKinCtrl_typeDynamics"	Specification of the jog dynamics
done	OUT	Bool	TRUE: Commanded functionality has been completed successfully
busy	OUT	Bool	TRUE: FB is not finished and new output values can be expected
active	OUT	Bool	TRUE: The setpoints are calculated
error	OUT	Bool	TRUE: An error occurred during the execution of the FB
status	OUT	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
remainingDistance	OUT	LReal	Distance-to-go of the current job
diagnostics	OUT	"LKinCtrl_typeJogFrameDiagnostics"	Diagnostics information
axesGroup	IN_OUT	TO_Kinematics	Kinematics technology object

## Status constants

Table 2-5: status constants of LKinCtrl\_MC\_JogFrame

Name	Data Type	Value	Comment
STATUS_EXECUTION_FINISHED	Word	16#0000	The command execution is finished by reaching the target position or stopping the jog command
STATUS_NO_CALL	Word	16#7000	No call of FB
STATUS_FIRST_CALL	Word	16#7001	First cycle of FB after enabling
STATUS_SUBSEQUENT_CALL	Word	16#7002	FB enabled

## 4.5. FB

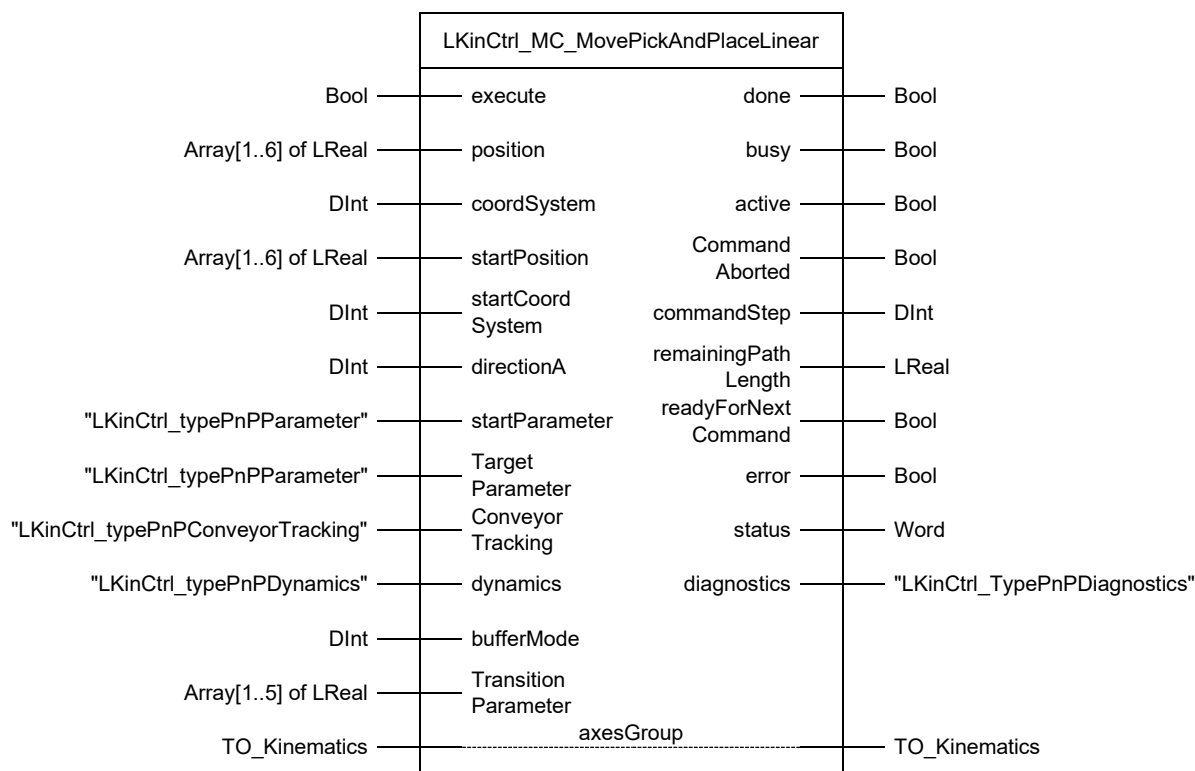
# LKinCtrl\_MC\_MovePickAndPlaceLinear

### (FB 35026)

#### NOTE

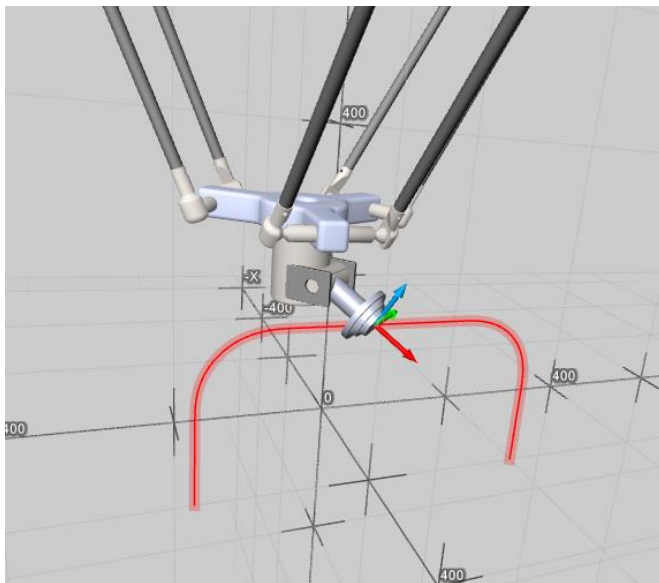
This chapter shows the general function description of MC\_MovePickAndPlaceLinear with focus on stand-alone usage. For further information on the corresponding MC\_MovePath command see chapter [3.14.](#)

Figure 4-5: LKinCtrl\_MC\_MovePickAndPlaceLinear



## Principle of operation

Figure 4-6: Typical pick and place sequence



The function block LKinCtrl\_MC\_MovePickAndPlaceLinear encapsulates a linear MC command sequence for typical Pick & Place applications. It is usable like a standard MC block and eases programming for static motions as well as conveyor tracking applications:

- Less programming effort
- Direct change between two tracked OCS possible (automatic calculation of WCS intermediate point by FB)
- Interface parameters

4-6: Parameter of LKinCtrl\_MC\_MovePickAndPlaceLinear

Name	P-Type	Data Type	Comment
execute	IN	Bool	Rising edge starts motion sequence; FB status must be 16#7000
position	IN	Array[1..6] of LReal	Target position, [1]: X coordinate, [2]: Y coordinate, [3]: Z coordinate, [4]: A coordinate, [5]: B coordinate, [6]: C coordinate
coordSystem	IN	DInt	Target coordinate system, = 0: WCS, = 1: OCS1, = 2: OCS2, = 3: OCS3
startPosition	IN	Array[1..6] of LReal	Start position (only valid if startCoordSystem >= 0), [1]: X coordinate, [2]: Y coordinate, [3]: Z coordinate, [4]: A coordinate, [5]: B coordinate, [6]: C coordinate
startCoordSystem	IN	DInt	Start coordiante system,

Name	P-Type	Data Type	Comment
			< 0: use actual, = 0: WCS, = 1..3: OCS1..3
directionA	IN	DInt	Direction for movement of A (Only necessary for kinematic types 2D+A/3D+A), = 1: positive, = 2: negative, = 3: shortest way
startParameter	IN	"LKinCtrl_typePnPParameter"	Start vector parameter
targetParameter	IN	"LKinCtrl_typePnPParameter"	Target vector parameter
conveyorTracking	IN	"LKinCtrl_typePnPConveyorTracking"	Conveyor tracking parameter
dynamics	IN	"LKinCtrl_typePnP Dynamics"	Dynamics specification
bufferMode	IN	DInt	bufferMode for start of path motion, = 1: Append motion = 2: Smooth with the lower velocity = 5: Smooth with the higher velocity
transitionParameter	IN	Array[1..5] of LReal	Transition parameter for start of path motion [1] >= 0: The specified value is used [1] < 0: The maximum rounding clearance defined by <AxesGroup>.transition. FactorBlendingLength is used
done	OUT	Bool	TRUE: Commanded functionality has been completed successfully
busy	OUT	Bool	TRUE: FB is not finished and new output values can be expected
active	OUT	Bool	TRUE: Active interpolation
commandAborted	OUT	Bool	TRUE: Commanded functionality has been aborted by another command
commandStep	OUT	DInt	Active command step, = 0: Not active, = 1: Linear start vector, = 2: Blending at start vector, = 3: Linear transition, = 4: Blending at target vector, = 5: Linear target vector
remainingPathLength	OUT	LReal	Remaining path length of sequence
readyForNextCommand	OUT	Bool	TRUE: All commands buffered and prepared – subsequent commands can be buffered
error	OUT	Bool	TRUE: An error occurred during the execution of the FB
status	OUT	Word	16#0000 - 16#7FFF: Status of the FB, 16#8000 - 16#FFFF: Error identification
diagnostics	OUT	"LKinCtrl_TypePnP Diagnostics"	Diagnostics information of FB
axesGroup	IN_ OUT	TO_Kinematics	Reference to the axes group

4-7: Status constants of LKinCtrl\_MC\_MovePickAndPlaceLinear

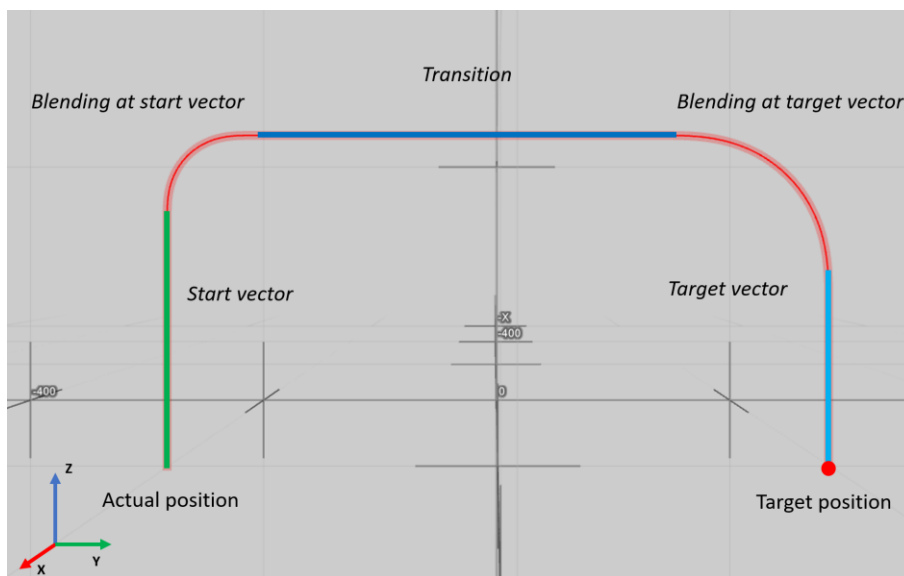
Name	Type	Value	Comment
STATUS_EXECUTION_FINISHED	Word	16#0000	Execution finished without errors
STATUS_NO_CALL	Word	16#7000	No job being currently processed
STATUS_FIRST_CALL	Word	16#7001	First call after incoming new job (rising edge 'execute')
STATUS_SUBSEQUENT_CALL	Word	16#7002	Subsequent call during active processing without further details
STATUS_WAIT_UNTIL_PREVIOUS_MOTION_DONE	Word	16#7100	Wait until previous motion is done
STATUS_BUFFER_CMDS	Word	16#7101	Buffer commands
STATUS_WAIT_FOR_PREPARATION	Word	16#7102	Wait until commands are prepared (Abort by MC_GroupStop)
STATUS_PATHMOTION_RUNNING	Word	16#7306	Path motion running
STATUS_PATHMOTION_INTERRUPTED	Word	16#7303	Path motion interrupted
STATUS_COMMAND_ABORTED	Word	16#7FFF	Commanded functionality has been aborted by another command

## Motion sequence

The following motion sequence is wrapped by the FB:

- Enable tracking on target coordinate system (optional)
- Linear motion along start vector (optional)
- Blending into transition motion (optional)
- Transition motion
- Blending into target vector (optional)
- Linear motion along target vector (optional)

Figure 4-7: Motion sequence



A rising edge on "execute" input starts the motion sequence to the target position specified in "position". The target coordinate system can be in WCS or OCS1..3 ("coordSystem").

The start and target vector as well as blending is not mandatory and deactivated by default. It can be specified in "startParameter" and "targetParameter".

The output "commandStep" indicates the current motion section:

- = 1: Motion in start vector
- = 2: Blending at start vector
- = 3: Transition motion
- = 4: Blending at target vector
- = 5: Motion in target vector

## Command buffering

The motion commands are queued one after the other in the motion queue. This means that the motion is always starting from the latest target position before triggering the FB, or the current position if no motion was active.

To ensure that all commands are processed in the correct order, subsequent MC commands must only be buffered as soon as output "readyForNextCommand" = TRUE.

The remaining path length of the motion sequence is shown at the output "remainingPathLength". It only shows a valid position after all commands are buffered ("readyForNextCommand" = TRUE).

**NOTE**

The output “remainingPathLength” is not affected by internal shortening of the calculated path, e.g., due to subsequent blending or synchronization or desynchronization to or from an OCS. In this case, the output might show a residual path length and is not finishing at 0.

**Conveyor tracking**

It is possible to establish conveyor tracking automatically when executing the MC\_MovePickAndPlace command. To do so, the input “conveyorTracking.enable” must be set to TRUE. The block uses the input values for belt axis, conveyor origin and initial object position and couples the target object coordinate system (Input “coordSystem”) with the referenced belt axis before starting the path motion.

Table 2-8: Parameter of LKinCtrl\_typePnPConveyorTracking

Name	Type	Value	Comment
enable	Bool	FALSE	= TRUE: Set tracking with command execution
conveyorBeltAxis	DB_ANY	0	Conveyor belt axis DB
conveyorBeltOrigin	TO_Struct_Kinematics_Frame	DEF_VAL	Frame to origin of conveyor belt
initialObjectPosition	TO_Struct_Kinematics_Frame	DEF_VAL	Initial object position on conveyor (only x allowed)

**NOTE**

For more information about conveyor tracking and the meaning, usage, and valid values of the conveyor tracking parameter, please refer to the TO Kinematics manual or chapter [3.6](#).

Conveyor tracking is not established if the target coordinate system is the WCS.

If internal tracking is selected (“conveyorTracking.enable” = TRUE), but the target OCS is already in tracking, no new tracking is established internally. However, the input “conveyorTracking.initialObjectPosition” is regarded for internal adaption of the target position, so the user does not need to take care of this.

**NOTICE****Unwanted motion of the kinematics**

It is possible that an externally set MC\_TrackConveyorBelt command is already in the motion queue (“busy” = TRUE) but not finished (“done” = TRUE). In this case, an undefined situation may occur if the internal tracking is also set. This situation cannot be checked by the FB and must be prevented by the user.

Active conveyor tracking of the actual coordinate system or the target coordinate system is regarded in path planning. If necessary, an intermediate point in WCS is calculated and inserted automatically.

**NOTE**

The intermediate point calculation is based on the programmed path dynamics. If these dynamics aren’t reached due to dynamics adaption or limitation of orientation dynamics, the linearity of the resulting path may be affected.

**Blending into command sequence from previous command**

With input “bufferMode” and “transitionParamter”, the user can specify whether the motion sequence should start with or without standstill. The following rules apply:



- No blending selected ("bufferMode" = 1)
  - Command waits until all previous motion is done and the motion queue is empty
  - Command sequence is buffered afterwards
- Blending selected ("bufferMode" = 2 or 5)
  - Command sequence is buffered once the last command of the previous command sequence is active

**NOTE**

By default, blending into the command ("bufferMode" = 2 or 5) is only possible if:

- no start direction is specified
- no motion between two OCS in tracking is programmed.

To enable blending in both cases, the explicit definition of the start position and start coordinate system is necessary ("startCoordSystem" >= 0)

**Dynamics specification**

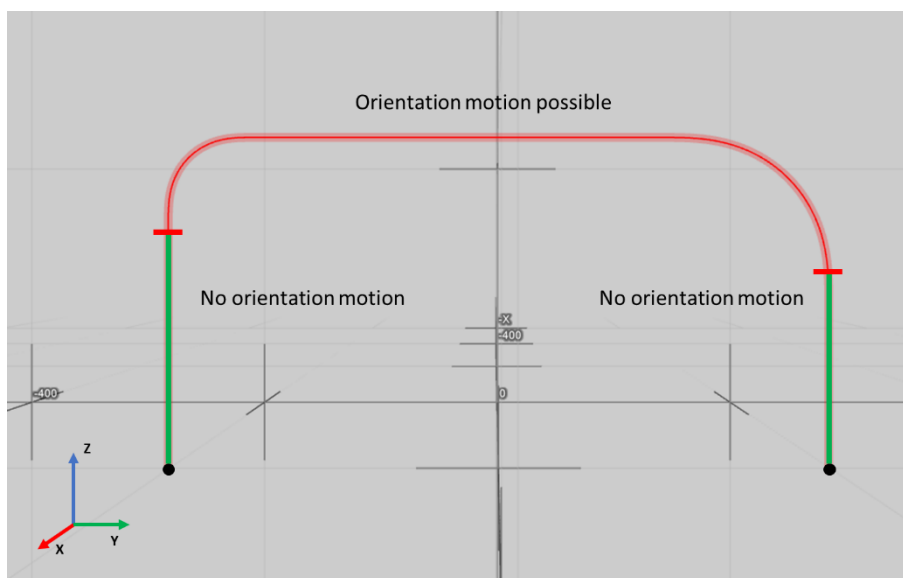
The programmed dynamics of the motion (path and orientation) as well as the dynamics adaption mode are specified in the "dynamics" input. A negative value implies the usage of default values.

The path dynamics scaling factors "startParameter.dynamicsFactor" and "targetParameter.dynamicsFactor" are scaling the programmed velocity along the respective vector.

**Orientation motion**

The orientation of the kinematics is only moving during the transition motion (blending and linear, "commandStep" = 2, 3, 4), not in the linear start or target vector, as shown in [Figure 4-8](#). If a 2D+A or 3D+A Kinematics type is used, the direction of the orientation motion can be forced with input "directionA".

Figure 4-8: Areas of orientation motion

**Start and target vector parametrization**

The vector parametrization is done via input "startParameter" and "targetParameter" with the following variables (See [Figure 4-7](#)):

Table 2-9: Parameter of LKinCtrl\_typePnPParameter

Name	Type	Value	Comment
direction	Array[1..3] of LReal	DEF_VAL	Direction of additional linear motion from start or target position (will be normed internally)
workingHeight	LReal	0.0	= 0: No direction, > 0: Working height in start direction
transitionArea	LReal	0.0	Permissible blending area at end of additional linear motion, < 0: The maximum rounding clearance defined by "<AxesGroup>.transition.FactorBlendingLength" is used, = 0: No blending, > 0: Blending area
dynamicsFactor	dynamicsFactor	100.0	Path velocity scaling factor (Limited between 1..200)

- Direction (always pointing from start / target position) in the used coordinate system (WCS or OCS)
  - "direction[1]": X-component
  - "direction[2]": Y-component
  - "direction[3]": Z-component
  - If all components are = 0: no vector is used
  - The resulting vector is normed to a unit vector
- Working height
  - "workingHeight" sets the length of the vector (relative from start or target position).
  - < 0: not allowed
  - = 0: no vector is used
  - > 0: vector length
- Transition area
  - "transitionArea" sets the length portion of the working height which can be used for blending to or from the transition motion.
  - < 0: the maximum possible blending radius according to <TOKinematics>.transition.factorBlendingLength is set
  - = 0: Standstill at vector end
  - > 0: Blending distance
  - >= "workingHeight": Only blending, no linear vector component

**NOTE**

With FW3.0.x, the following values must be set if an OCS with active conveyor tracking is part of the motion (actual or target position):

startParameter.transitionArea = 0.0 (No blending above start point)

targetParameter.transitionArea = 0.0 (No blending above target point)

dynamics.dynamicAdaption = 0 (No dynamic adaption)

Figure 4-9: Vector parameter

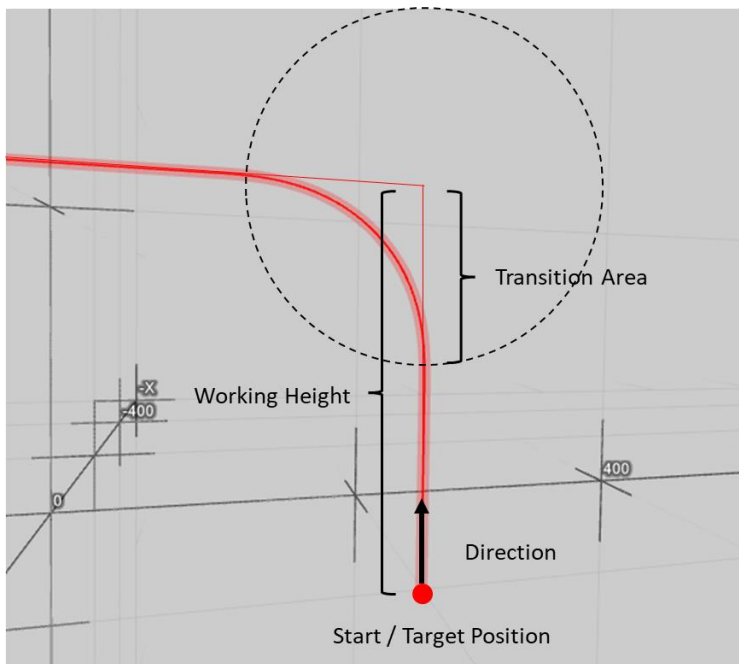


Figure 4-10 shows an example parametrization:

- Start vector (left side):
  - Pointing in positive Z direction
  - Linear motion ( $\text{workingHeight} > \text{transitionArea}$ )
  - Blending active ( $\text{transitionArea} > 0$ )
- Target vector (right side):
  - Pointing equally in positive Z and negative Y direction
  - No blending ( $\text{transitionArea} = 0$ )

Figure 4-10: Start and target parameter example 1

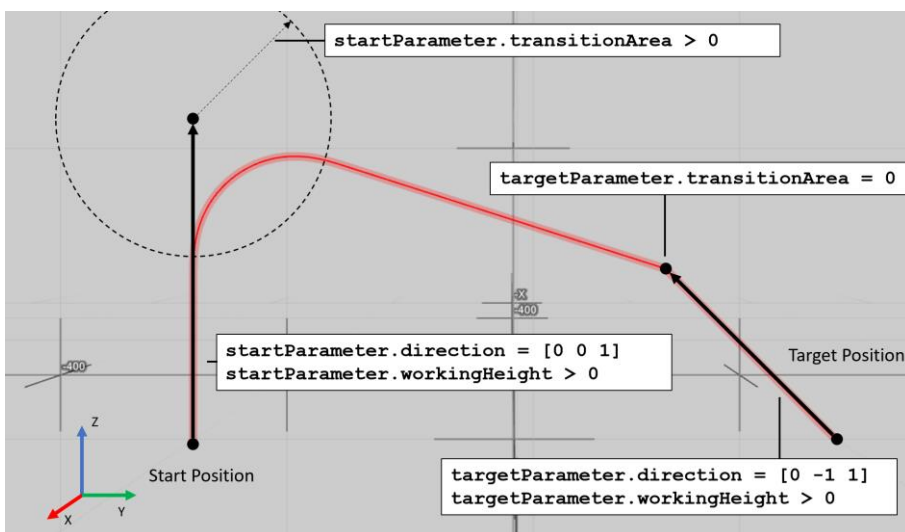
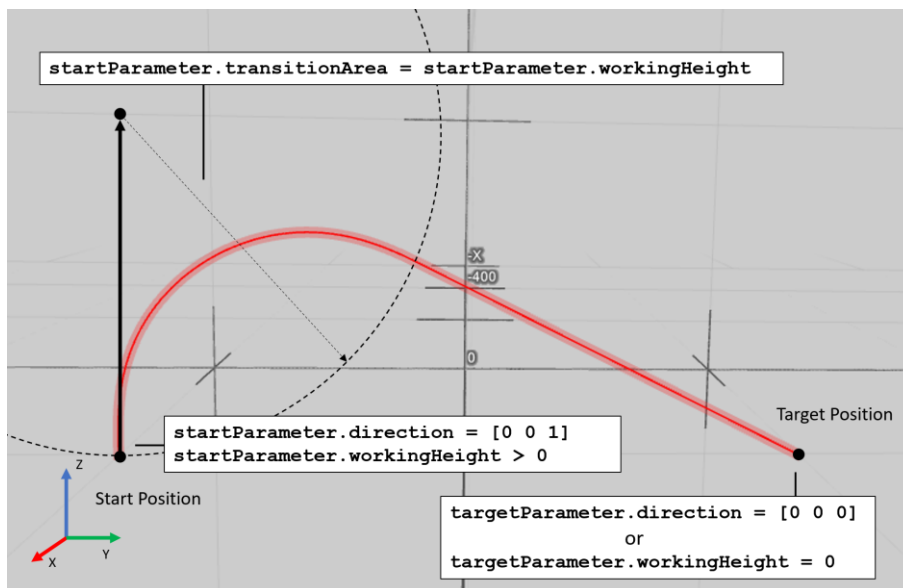


Figure 4-11 shows another example parametrization:

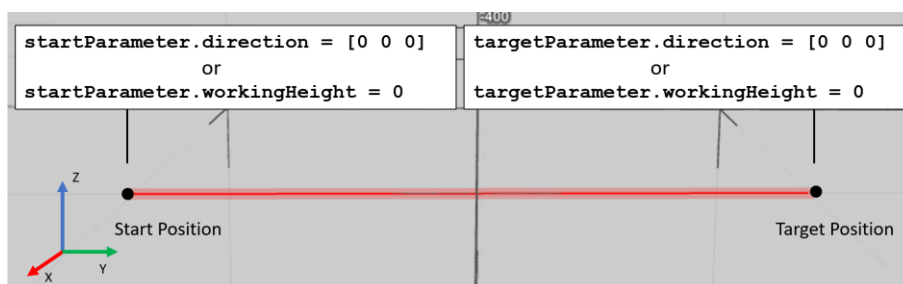
- Start vector (left side):
  - Pointing in positive Z direction
  - No linear motion ( $\text{workingHeight} = \text{transitionArea}$ )
  - Blending active ( $\text{transitionArea} > 0$ )
- Target vector (right side)
  - Not present ( $\text{workingHeight} = 0$  or  $\text{direction} = [0\ 0\ 0]$ , motion ends with transition motion)

Figure 4-11: Start and target parameter example 2



If no vector is specified, a linear motion from start to target results. The FB acts like an MC\_MoveLinearAbsolute command.

Figure 4-12: Start and target parameter example 3



### Start position and coordinate system specification

The explicit specification of the start position and start coordinate system is necessary when blending into command is activated ("bufferMode" > 1) and the motion has a start vector definition or results in a change between two OCS in tracking.

The default value for coordinate system specification can remain ("startCoordSystem" = -1) if above condition does not apply.

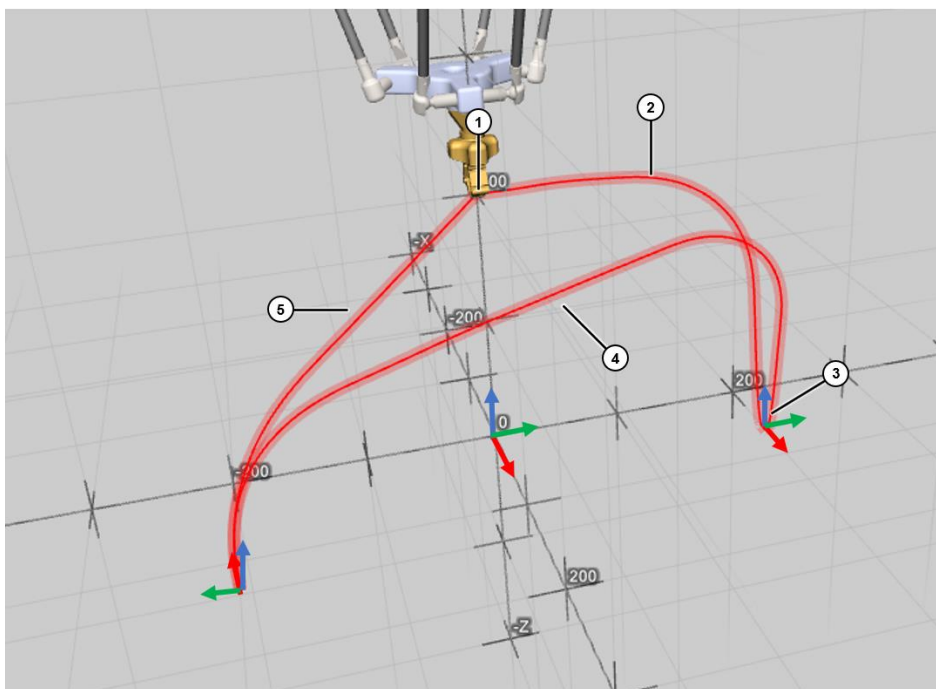
### Example use case

#### NOTE

FW V3.1 was used for creating the following use case.

When using FW V3.0, the parameter "...pickParameter.transitionParameter" and "...placeParameter.transitionParameter" and the default value for dynamics adaption must be set to 0.

Figure 4-13: Example use case



The example use case from [Figure 4-13](#) shows a typical pick and place application with a pick and a place position, both in conveyor tracking.

It is separated into the following sequence:

1. Track OCS1
1. Move from Wait- to Pickposition (OCS1)
2. Track OCS2
3. Move from Pick- to Placeposition (OCS2)
4. Move from Place- to Waitposition (WCS)

Conveyor tracking for OCS1 and OCS2 is implicitly started by the respective motion block according to the configuration in [Figure 4-14](#). The input "conveyorTracking.initialObjectPosition" defines the current object position and needs to be updated each time before execution of the sequence, like shown in [Figure 4-15](#).

Figure 4-14: Conveyor tracking parameter

```

//Pick tracking parameter
//-----
//Enable
"DataControlKinl".pickConveyorTracking.enable := TRUE;
//Conveyor belt
"DataControlKinl".pickConveyorTracking.conveyorBeltAxis := "Conv1_Axis";
//OCS Origin
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.x := 0.0;
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.y := 200.0;
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.z := 0.0;
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.a := 0.0;
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.b := 0.0;
"DataControlKinl".pickConveyorTracking.conveyorBeltOrigin.c := 0.0;

//Place tracking parameter
//-----
//Enable
"DataControlKinl".placeConveyorTracking.enable := TRUE;
//Conveyor belt
"DataControlKinl".placeConveyorTracking.conveyorBeltAxis := "Conv2_Axis";
//OCS Origin
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.x := 200.0;
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.y := -200.0;
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.z := 0.0;
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.a := -180.0;
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.b := 0.0;
"DataControlKinl".placeConveyorTracking.conveyorBeltOrigin.c := 0.0;

```

Figure 4-15: Set initialObjectPosition

```

//Wait for trigger
"InstRTrigExecuteMotion"(CLK:="DataControlKinl".executeMotion);
IF ("InstRTrigExecuteMotion".Q = TRUE) THEN
  //Set measured value to initialObjectPosition with execution
  "DataControlKinl".pickConveyorTracking.initialObjectPosition.x := "DataControlKinl".measuredValuePick;
  "DataControlKinl".placeConveyorTracking.initialObjectPosition.x := "DataControlKinl".measuredValuePlace;
END_IF;

```

All vector parameters are set according to the values shown in [Figure 4-16](#).

Figure 4-16: Pick and place vector parameter

```

//Pick vector parameter
//-----
//Dynamics factor
"DataControlKinl".pickParameter.dynamicsFactor := 50.0;
//Vector direction
"DataControlKinl".pickParameter.direction[1] := 0.0;
"DataControlKinl".pickParameter.direction[2] := 0.0;
"DataControlKinl".pickParameter.direction[3] := 1.0;
//Working height
"DataControlKinl".pickParameter.workingHeight := 200.0;
//Transition area
"DataControlKinl".pickParameter.transitionArea := 100.0;

//Place vector parameter
//-----
//Dynamics factor
"DataControlKinl".placeParameter.dynamicsFactor := 75.0;
//Vector direction
"DataControlKinl".placeParameter.direction[1] := 0.0;
"DataControlKinl".placeParameter.direction[2] := 0.0;
"DataControlKinl".placeParameter.direction[3] := 1.0;
//Working height
"DataControlKinl".placeParameter.workingHeight := 100.0;
//Transition area
"DataControlKinl".placeParameter.transitionArea := 100.0;

```

The pick position is approached and departed with a vertical movement and a working height of 200mm. The permissible blending distance is 100mm. Velocity is reduced to 50%.

Place is approached and departed with a dynamics factor of 75%, also with a vertical vector with 100mm length. However, the vector is completely blended, as "transitionArea" is set same as the working height.

Pick tracking and motion (1, 2) is done with the MC\_MovePickAndPlaceLinear instance shown in

[Figure 4-17](#). The target coordinate system is OCS1 ("coordSystem" = 1). Start parameter are left empty, which means that default values are used. This results in a motion without a defined start direction, it points directly to the target vector end defined in "pickParameter". Default dynamic settings are used, as the input parameter "dynamics" is left empty. Conveyor tracking is automatically done when executing the command, according to the input "conveyorTracking".

Figure 4-17: Instance for motion from wait to pick

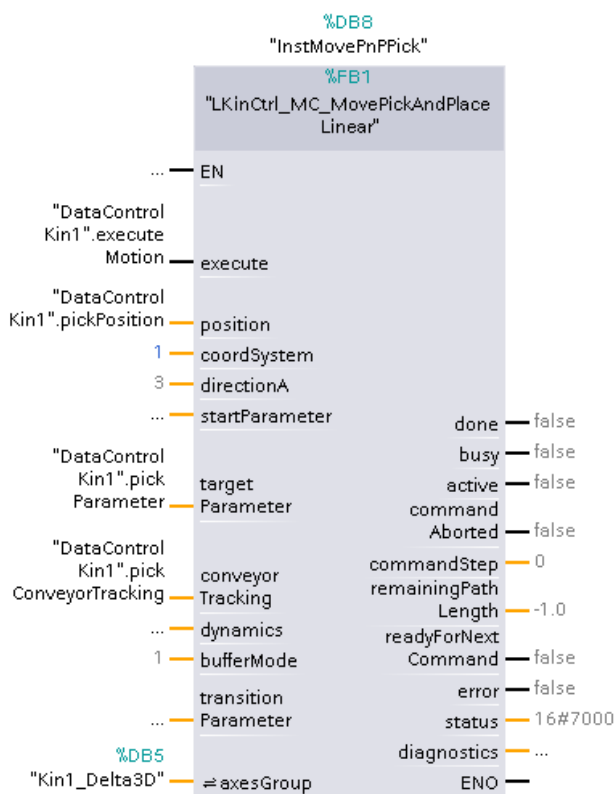
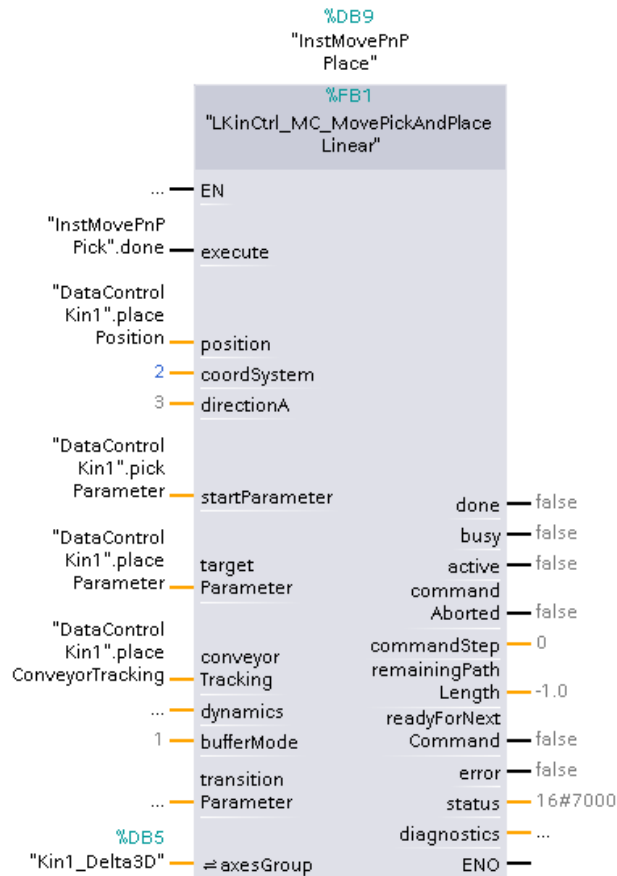


Figure 4-18 shows the instance for tracking and motion from pick to place (3, 4), which is started after the previous command. The target coordinate system is OCS2 ("coordSystem" = 2). Both start and target vector parameter are defined according to Figure 4-16. The tracking of OCS2 is done according to the parameters on input "conveyorTracking".

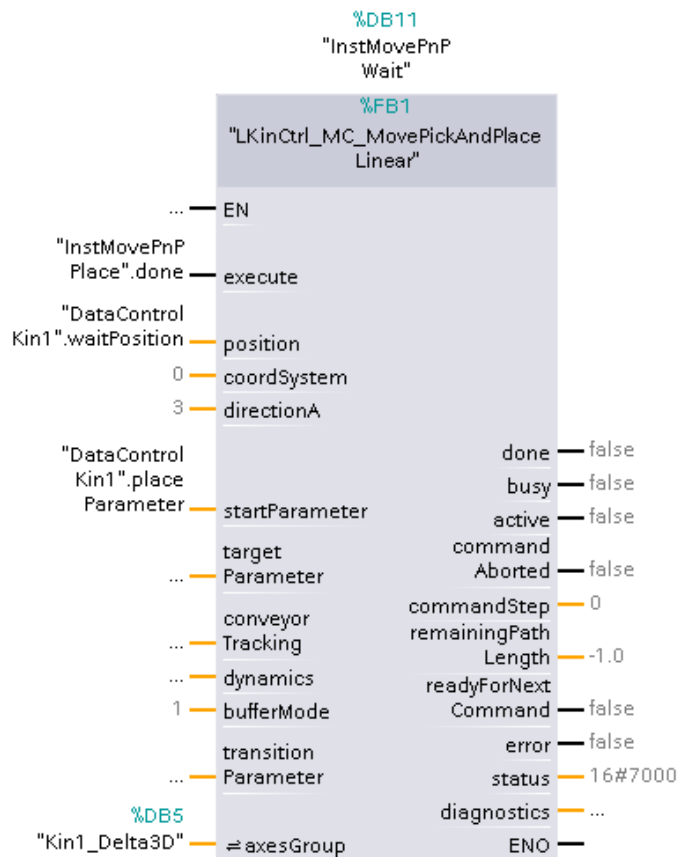
Figure 4-18: Instance for motion from pick to place





The final motion (5) is done with the instance from [Figure 4-19](#). Here, only the start vector is defined, which means that no target vector is present. The movement ends at the wait position in no defined direction. As the target is in WCS, no automatic tracking needs to be performed.

Figure 4-19: Instance for motion from place to wait



## Error handling

Present errors are acknowledged by setting "execute" = FALSE.

Detailed diagnostics are stated in output "diagnostics"

If the block gets stuck in status 16#7000 – 16#8000 it can be aborted by MC\_GroupStop.

For errors on motion commands (status = 16#8701 – 16#8720), the "subFunctionStatus" shows the MC command error. Further information can be found in the TO documentation (<https://support.industry.siemens.com/cs/ww/en/view/109812061>).

For error during command creation (status = 16#8601), [Table 2-10](#) and [Table 2-11](#) are showing more information.

Table 2-10: Meaning of subFunctionStatus for Status = 16#8601

Name	Type	Value	Comment
ERR_GET_TRACKING_INFO	Word	16#9000	Error - Get tracking info Invalid conveyor axis type
ERR_BLENDEDING_NOT_ALLOWED	Word	16#9001	Error - Blending is not allowed Start vector or direct tracking OCS change selected and no start coordinate system defined

Name	Type	Value	Comment
ERR_GET_ORI_DIMENSION	Word	16#9002	Error - Get orientation dimension Unknown kinematics type
ERR_GET_LIN_POINT_DIST	Word	16#9003	Error - Get linear point distance
ERR_CALC_INTERM_POINT	Word	16#9004	Error - Intermediate point calculation See additionalValue2 for more information

Table 2-11: Meaning of AdditionalValue1 for subFunctionStatus = 16#9004 and Status = 16#8601

Name	Type	Value	Comment
ERR_EQUAL_POINTS	Word	16#9000	Error – Start and target point are identical
ERR_INVALID_CS	Word	16#9001	Error - invalid start or target coordSystem
ERR_DYN_PROFILE	Word	16#9002	Error – at positioning profile calculation Check path dynamic settings
ERR_CREATE_ROT_VEC	Word	16#9003	Error - at calculation of rotation vector
ERR_CREATE_VEC_ANGLE	Word	16#9004	Error - at calculation of rotation vector angle
ERR_INVALID_ORI_DIM	Word	16#9005	Error - orientation dimension invalid
ERR_CALC_ORI	Word	16#9006	Error - orientation calculation
ERR_GET_TRACKING_INFO	Word	16#9007	Error - at getting tracking information

## 5. Additional information

### 5.1. Trigger multiple commands per call

It is possible to trigger more than one command per call of FB MovePath. This can help keeping the MotionQueue filled and can eventually result in better path dynamics.

To do so, increase the following PLC tags in the tag table LKinCtrl\_Configuration:

- LKINCTRL\_NO\_OF\_START\_EXECUTE\_CMDS
- LKINCTRL\_NO\_OF\_SEQ\_EXECUTE\_CMDS

#### NOTE

Increasing the number of commands to be triggered can lead to an increased cycle time. To save some cycle time, the repetition is not always done. For example, if the motion queue is already full, the loop is exited early.

### 5.2. Continue at start

Due to an undesired system behavior of TO Kinematics. It can occur that blending between two commands is not correct. This issue is related to an interrupt command being aborted and no continue follows.

To avoid this issue, an interrupt always needs a continue. To ensure that after the abort of the interrupt command a continue follows, the FB MovePath executes a continue command before starting the path motion.

This behavior can be controlled by the internal constant "CONTINUE\_AT\_START".

## 6. Error handling

This chapter shows the error values of the function blocks from this application and explains how to comprehend diagnostics information and their structures for trouble shooting purposes.

### NOTE

If available always check the output “diagnostics” of the respective function block to get the complete error information.

### 6.1. LKinCtrl\_MC\_MovePath

The function block LKinCtrl\_MC\_MovePath reports its status and diagnostics about internal function blocks via two outputs.

The output *status* outputs the following values regarding the state of the LKinCtrl\_MC\_MovePath function block.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	error due to an undefined state in state machine
ERR_EXECUTE_KINMOTIONCMD	Word	16#8201	error due to error in FB LKinCtrl_ExecuteKinMotionCmd
ERR_WRONG_SEQUENCEMODE	Word	16#8203	error due to invalid sequenceMode on input
ERR_PATHDATA_PARAMETERS	Word	16#8204	error due to malconfiguration in motion command parameters
ERR_OFFSET_CALCULATION	Word	16#8205	error due to calculation error in offset compensation
ERR_ERROR_STILL_PENDING	Word	16#8207	error due to previous not acknowledged error (reset MovePath to acknowledge)
ERR_SET_FLAG_CONTROL	Word	16#8208	error due to error in set flag control FB
ERR_VALUE_FLAG_CONTROL	Word	16#8209	error due to error in value flag control FB
ERR_TRIGGER_NOT_POSSIBLE	Word	16#820A	error due to new execute not possible

The output *diagnostics* outputs a structure that contains several information about internal function blocks and states.

Name	Data type	Value	Comment
errorID	Word	16#0	errorID of the FB
pathDataName	String[4]	“	name of PathData an error occurred in
cmdNumber	DInt	0	command number (in PathData) an error occurred in
stateMovePath	DInt	0	state machine status FB
statusSubFunction	Word	16#0	status of LKinCtrl_MC_ExecuteKinMotionCmd
motionFBNumber	DInt	0	motion FB number (internal) an error occurred in
motionFBStatus	DWord	16#0	status of motion FB (internal) an error occurred in
kinematicsStatusWord	DWord	16#0	statusWord TO kinematics

Name	Data type	Value	Comment
kinematicsErrorWord	DWord	16#0	errorWord TO kinematics

This *diagnostics* structure is usually used in combination of the following errors, reporting via the *status* output from the function block LKinCtrl\_MC\_MovePath:

- ERR\_EXECUTE\_KINMOTIONCMD from [LKinCtrl\\_MC\\_ExecuteKinMotionCmd](#)
- ERR\_PATHDATA\_PARAMETERS from [LKinCtrl\\_PreBuffer](#)
- ERR\_OFFSET\_CALCULATION from [LKinCtrl\\_OffsetContour](#)

In these cases the *statusSubFunction* member from the *diagnostics* structure outputs states or error values that are listed in the next chapters accordingly.

### 6.1.1. LKinCtrl\_MC\_ExecuteKinMotionCmd

The function block LKinCtrl\_MC\_ExecuteKinMotionCmd reports its status via outputs that are internally gathered by the function block LKinCtrl\_MC\_MovePath.

The following error values are applicable for the *statusSubFunction* member from the *diagnostics* output of function block LKinCtrl\_MC\_MovePath, if the output *status* equals to the value of state *ERR\_EXECUTE\_KINMOTIONCMD*.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	error due to an undefined state in state machine
ERR_GROUPINTERRUPT_FAILED	Word	16#8242	error MC_GroupInterrupt failed
ERR_GROUPCONTINUE_FAILED	Word	16#8243	error MC_GroupContinue failed
ERR_GROUPSTOP_FAILED	Word	16#8244	error MC_GroupStop failed
ERR_TO_KINEMATIC_ERROR	Word	16#8245	error in TO_Kinematic
ERR_LINEAR_CMD	Word	16#8246	error MC_MoveLinear failed
ERR_CIRCULAR_CMD	Word	16#8247	error MC_MoveCircular failed
ERR_CMD_PARAMETER	Word	16#8248	error command parametrization
ERR_CALC_CIRC_CMD	Word	16#8249	error circular command conversion: arc calculation
ERR_MOVEDIRECT_CMD	Word	16#8250	error MC_MoveDirect failed
ERR_TRACKCONVEYOR_CMD	Word	16#8251	error MC_TrackConveyorBelt failed
ERR_DESYNC_RELATIVE	Word	16#8252	error due to relative command used for desync conveyor
ERR_CONVEYOR_CONFIGURATION	Word	16#8253	error due to invalid conveyor configuration
ERR_PREPARE_COMMAND	Word	16#8254	error FB PrepareCommand failed
ERR_SETOCS_CMD	Word	16#8255	error MC_SetOCSFrame failed
ERR_SETTOOL_CMD	Word	16#8256	error MC_SetTool failed
ERR_DEFINETOOL_CMD	Word	16#8257	error MC_DefineTool failed
ERR_DEFINEWSZONE_CMD	Word	16#8258	error MC_DefineWorkspaceZone failed

Name	Data type	Value	Comment
ERR_ACTIVATEWSZONE_CMD	Word	16#8259	error MC_SetWorkspaceZoneActive failed
ERR_DEACTIVATEWSZONE_CMD	Word	16#825A	error MC_SetWorkspaceZoneInactive failed
ERR_DEFINEKINZONE_CMD	Word	16#825B	error MC_DefineKinematicsZone failed
ERR_ACTIVATEKINZONE_CMD	Word	16#825C	error MC_SetKinematicsZoneActive failed
ERR_DEACTIVATEKINZONE_CMD	Word	16#825D	error MC_SetKinematicsZoneInactive failed
ERR_CIRC_CMD_USER_FRAME	Word	16#825E	error due to invalid circular command in user frame
ERR_LINEARPICKANDPLACE_CMD	Word	16#825F	Error LKinCtrl_MC_MovePickAndPlaceLinear failed

### 6.1.2. LKinCtrl\_PreBuffer

The function block LKinCtrl\_PreBuffer reports its status via outputs that are internally gathered by the function block LKinCtrl\_MC\_MovePath.

The following error values are applicable for the *statusSubFunction* member from the *diagnostics* output of function block LKinCtrl\_MC\_MovePath, if the output *status* equals to the value of state *ERR\_PATHDATA\_PARAMETERS*.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	Error due to an undefined FB state
ERR_OFFSET_COMPENSATION	Word	16#8261	error during offset calculation
ERR_LASTPOS_CALCULATION	Word	16#8262	error during calculation of previous target position
ERR_INVALID_PATHDATA_TYPE	Word	16#8263	error due to an unsupported pathdata type
ERR_CIRC_CMD_USER_FRAME	Word	16#8264	error due to invalid circular command in user frame
ERR_INVALID_POINT	Word	16#8265	error due to invalid point

In case of an error within the PathData (e.g. wrong parameterization of commands) the following error values are applicable for the *statusSubFunction* member from the *diagnostics* output of function block LKinCtrl\_MC\_MovePath. Additionally, the member *cmdNumber* from the *diagnostics* output of function block LKinCtrl\_MC\_MovePath will output the command number of the respective faulty command within the PathData.

Name	Data type	Value	Comment
STATUS_NO_ERROR_IN_COMMAND	Word	16#0000	status no error found command
ERR_PATHDATA_CMDTYPE	Word	16#80A0	error due to wrong cmdType
ERR_PATHDATA_COORDSYSTEM	Word	16#80B1	error due to wrong coordinate system
ERR_PATHDATA_VELOCITY	Word	16#8003	error due to wrong velocity
ERR_PATHDATA_ACCELERATION	Word	16#8004	error due to wrong acceleration
ERR_PATHDATA_DECELERATION	Word	16#8005	error due to wrong deceleration
ERR_PATHDATA_JERK	Word	16#8006	error due to wrong jerk

Name	Data type	Value	Comment
ERR_PATHDATA_ORIENTATIONDIRECTION	Word	16#8007	error due to wrong orientation direction
ERR_PATHDATA_BUFFERMODE	Word	16#80B2	error due to wrong buffer mode
ERR_PATHDATA_DYNAMICADAPTION	Word	16#80B5	error due to wrong dynamic adaption
ERR_PATHDATA_CIRCMODE	Word	16#80B6	error due to wrong circle mode
ERR_PATHDATA_PATHCHOICE	Word	16#80B9	error due to wrong path choice
ERR_PATHDATA_CIRCLEPLANE	Word	16#80BA	error due to wrong circle plane
ERR_PATHDATA_RADIUS	Word	16#80BB	error due to wrong radius
ERR_PATHDATA_ARC	Word	16#80BC	error due to wrong arc
ERR_PATHDATA_SETFLAGS_FLAG	Word	16#80D1	error due to wrong set flag definition
ERR_PATHDATA_SETFLAGS_FLAGMODE	Word	16#80D2	error due to wrong set flag mode definition
ERR_PATHDATA_VALUEFLAGS_FLAG	Word	16#80D3	error due to wrong value flag definition
ERR_PATHDATA_VALUEFLAGS_FLAGMODE	Word	16#80D4	error due to wrong value flag mode definition
ERR_PATHDATA_OFFSET_PARA_NO	Word	16#80A1	error due to wrong offset parameter set number
ERR_PATHDATA_OFFSET_MAINPATHPLANE	Word	16#80A2	error due to wrong main path plane
ERR_PATHDATA_POSITIONMODE	Word	16#80A3	error due to wrong position mode
ERR_PATHDATA_LINK_CONSTELLATION	Word	16#80A4	error due to wrong link constellation
ERR_PATHDATA_CONVEYOR_PARA_NO	Word	16#80A5	error due to wrong conveyor parameter set number
ERR_PATHDATA_TOOLNUMBER	Word	16#80A6	error due to wrong tool number
ERR_PATHDATA_COORDINATES	Word	16#80A7	error due to wrong coordinates
ERR_PATHDATA_ZONENUMBER	Word	16#80A8	error due to wrong zone number
ERR_PATHDATA_ZONECONFIGURATIONINDEX	Word	16#80A9	error due to wrong zone configuration index
ERR_PATHDATA_ZONEDEACTIVATIONMODE	Word	16#80AA	error due to wrong zone de-/activation mode
ERR_PATHDATA_PICKANDPLACE_STARTVECTOR	Word	16#80AB	error due to wrong pick and place start vector parametrization
ERR_PATHDATA_PICKANDPLACE_TARGETVECTOR	Word	16#80AC	error due to wrong pick and place target vector parametrization
ERR_PATHDATA_PICKANDPLACE_BLENDING_NOT_POSSIBLE	Word	16#80AD	error due to invalid buffer mode (blending into command is only possible under certain conditions, see <a href="#">2.1.3</a> )

### 6.1.3. LKinCtrl\_OffsetContour

The function block LKinCtrl\_OffsetContour reports its status via outputs that are internally gathered by LKinCtrl\_MC\_MovePath.

The following error values are applicable for the *statusSubFunction* member from the *diagnostics* output of LKinCtrl\_MC\_MovePath, if the output *status* equals to the value of state *ERR\_OFFSET\_CALCULATION*.

Name	Data type	Value	Comment
ERR_COFF_INVALID_TOOL_LENGTH	Word	16#8301	Error - invalid tool length
ERR_COFF_INVALID_TOOL_RADIUS	Word	16#8302	Error - invalid tool radius
ERR_COFF_INVALID_PATH_PLANE	Word	16#8303	Error - invalid path plane
ERR_COFF_INVALID_COMPENSATION_DIRECTION	Word	16#8304	Error - invalid compensation direction
ERR_COFF_FUNCTION_NOT_SUPPORTED	Word	16#8305	Error - functionality not supported
ERR_COFF_INVALID_COMMAND_TYPE	Word	16#8306	Error - invalid command type
ERR_COFF_INVALID_CIRCLE_MODE	Word	16#8307	Error - invalid circle mode
ERR_COFF_INVALID_PATH_PLANE_OF_CIRCULAR_CMD	Word	16#8308	Error - invalid path plane of a circular command
ERR_COFF_INVALID_PATH_CHOICE_OF_CIRCULAR_CMD	Word	16#8309	Error - invalid path choice of circular command
ERR_COFF_INTERNAL	Word	16#830F	Error - internal
ERR_COFF_VECTOR_LENGTH_ZERO	Word	16#8310	Error - vector length is zero
ERR_COFF_INVALID_ACOS_ARGUMENT	Word	16#8311	Error - invalid ACOS argument
ERR_COFF_LINES_DO_NOT_INTERSECT	Word	16#8312	Error - no solution - lines don't intersect
ERR_COFF_ZERO_VECTOR	Word	16#8313	Error - zero vector
ERR_COFF_CMD_LENGTH_ZERO	Word	16#8314	Error - cmd with length zero not permitted
ERR_COFF_LINE_CIRCLE_DO_NOT_INTERSECT	Word	16#8315	Error - no solution - line and circle don't intersect
ERR_COFF_CIRCLES_DO_NOT_INTERSECT	Word	16#8318	Error - no solution - circles don't intersect
ERR_COFF_ONE_CIRC_CONTAINED_IN_THE_OTHER	Word	16#8319	Error - one circle is contained in the other
ERR_COFF_NO_SOLUTION_FOR_GIVEN_LINEAR_CMD	Word	16#8320	Error - no valid solution with given linear command
ERR_COFF_NO_SOLUTION_FOR_GIVEN_CIRCULAR_CMD	Word	16#8321	Error - no valid solution with given circular command
ERR_COFF_LINEAR_CMD_AND_TOOL_RADIUS_LEAD_TO_REVERSAL	Word	16#8322	Error - linear command in combination with tool radius leads to reversal movement



Name	Data type	Value	Comment
ERR_COFF_CIRCLE_CENTERS_EQUAL	Word	16#8350	Error - circle centers are equal infinite intersections (only internal)

## 6.2. LKinCtrl\_MC\_MovePickAndPlaceLinear

The function block LKinCtrl\_MC\_MovePickAndPlaceLinear reports its status via two outputs. The output *status* reports the status of the function block itself and the output *diagnostics* reports the status of internal sub functions such as motion control instructions as a structure.

The following error values are applicable for the output *status* of the function block.

Name	Data type	Value	Comment
ERR_PAR_START	Word	16#8201	Error - Start parameters invalid
ERR_PAR_TARGET	Word	16#8202	Error - Target parameters invalid
ERR_COORDSYS_TARGET	Word	16#8204	Error - coordSystem is invalid (Not WCS or OCS)
ERR_DIR_A_TARGET	Word	16#8205	Error - DirectionA is invalid
ERR_DYN_FACTOR_START	Word	16#8206	Error - Specification of start dynamics factor
ERR_DYN_FACTOR_TARGET	Word	16#8207	Error - Specification of target dynamics factor
ERR_BUFFERMODE	Word	16#8208	Error - BufferMode specification
ERR_BLENDEDING_NOT_POSSIBLE	Word	16#8209	Error - Blending not allowed
ERR_MOTIONQUEUE_LENGTH	Word	16#820A	Error - Motion Queue length must be at least 4
ERR_TRACKING_PARAMETER	Word	16#820B	Error - Invalid tracking parameter
ERR_COORDSYS_START	Word	16#820C	Error – startCoordSystem is invalid
ERR_UNDEFINED_STATE	Word	16#8600	Error due to an undefined state in state machine
ERR_CREATE_CMD	Word	16#8601	Error during command creation
ERR_INTERRUPT	Word	16#8701	Error during interrupt
ERR_CONTINUE	Word	16#8702	Error during continue
ERR_MOVE_POST_START	Word	16#8711	Error during motion to post start position
ERR_MOVE_INTERMEDIATE	Word	16#8713	Error during motion to intermediate position
ERR_MOVE_PRE_TARGET	Word	16#8714	Error during motion to pre target position
ERR_MOVE_TARGET	Word	16#8716	Error during motion to target position
ERR_TRACK_CONVEYOR	Word	16#8720	Error during track conveyor belt

### 6.3. LKinCtrl\_MC\_JogFrame

The function block LKinCtrl\_MC\_JogFrame reports its status via two outputs. The output *status* reports the status of the function block itself and the output *diagnostics* reports the status of internal sub functions such as motion control instructions as a structure.

The following error values are applicable for the output *status* of the function block.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	Error due to an undefined FB state
ERR_TO_KINEMATICS	Word	16#8001	Error at technology object kinematics
ERR_INVALID_JOG_DIRECTION	Word	16#8007	The specified jog direction is invalid (valid are: 1=x-direction, 2=y-direction, 3=z-direction, 4=a-direction, 5=b-direction, 6=c-direction, 7=Z-axis - about A,, 8=Y-axis - about B, 9=X-axis - about C)
ERR_MOTION_JOB_IN_PROGRESS	Word	16#8090	A motion job is in progress or the kinematics control panel is activated
ERR_TWO_JOG_COMMANDS_SIMULTANEOUSLY	Word	16#8091	Two jog commands were triggered simultaneously
ERR_INVALID_COORDINATE_SYSTEM	Word	16#80B1	The specified coordinate system is invalid (valid values are 0=WCS, 1= OCS1, 2=OCS2, 3=OCS3, 100=MCS, 101=JCS – KinPlus only)
ERR_INVALID_TOOL	Word	16#80CA	The specified tool is invalid (valid values are 1= tool1, 2=tool2, 3=tool3)
ERR_JOG_DIRECTION_NOT_POSSIBLE	Word	16#8100	The specified jog direction is not possible due to reduced degrees of freedom (5D)

### 6.4. Auxiliary blocks

The auxiliary blocks are reporting their status via the *diagnostics* output as a structure. This is described in the following chapters for each function block respectively. The member *status* outputs the state of the auxiliary block. The member *subFunctionSate* outputs the state of internal function blocks such as motion control instructions.

Name	Data type	Value	Comment
status	Word	DEF_VAL	Status of the FB or error identification when error occurred
subfunctionStatus	Word	DEF_VAL	Status or return value of called FBs, FCs and system blocks
axisNumber	Int	DEF_VAL	number of the corresponding axis the error occurred in or a function effecting it
errorDetail	UDInt	DEF_VAL	error detail of corresponding technology object the error occurred in
stateNumber	DInt	DEF_VAL	State in the state machine of the FB when the error occurred

### 6.4.1. LKinCtrl\_MC\_GroupPower

The following error values are applicable for the output *status* of the function block.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	error due to an undefined state in state machine
ERR_AXIS_ERROR	Word	16#8370	error pending at axis
ERR_KIN_ERROR	Word	16#8371	error pending at kinematics
ERR_MC_POWER	Word	16#8372	error during execution of MC_POWER for axis
ERR_AXIS_REF	Word	16#8373	error due to missing axis reference

### 6.4.2. LKinCtrl\_MC\_GroupReset

The following error values are applicable for the output *status* of the function block.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	Error due to an undefined state in state machine
ERR_AXIS_ERROR	Word	16#8360	Error pending at axis
ERR_KIN_ERROR	Word	16#8361	Error pending at kinematics
ERR_MC_RESET_AXES	Word	16#8362	Error during execution of MC_RESET for axis
ERR_MC_RESET_KIN	Word	16#8363	Error during execution of MC_RESET for kinematics
ERR_AXIS_REF	Word	16#8373	error due to missing axis reference

### 6.4.3. LKinCtrl\_MC\_GroupHome

The following error values are applicable for the output *status* of the function block.

Name	Data type	Value	Comment
ERR_UNDEFINED_STATE	Word	16#8600	Error due to an undefined state in state machine
ERR_AXIS_ERROR	Word	16#8350	Error pending at axis
ERR_MC_HOME	Word	16#8351	Error during execution of MC_HOME for axis
ERR_AXIS_REF	Word	16#8373	error due to missing axis reference

## 7. PLC Tags

The following table shows the library internal PLC tag table. As there are no tags included, the table shows library constants only. The use of the single constants is explained in the chapters of their corresponding use case.

### 7.1. LKinCtrl\_Configuration

These tags are meant to configure the application. The values can be changed at will.

Name	Data type	Value	Comment
LKINCTRL_LENGTH_OF_CMDPOINT_NAME	Int	30	length of point name
LKINCTRL_LENGTH_OF_OFFSET_NAME	Int	20	length of offset name
LKINCTRL_LENGTH_OF_PATHDATA_NAME	Int	20	length of PathData name
LKINCTRL_LENGTH_OF_TO_NAME	UInt	30	Length of technology object names
LKINCTRL_NO_OF_CMD_SETFLAGS	Int	3	number of Boolean flags per command
LKINCTRL_NO_OF_CMD_VALUEFLAGS	Int	4	number of LREAL flags per command
LKINCTRL_NO_OF_CONVEYOR	Int	5	number of configurable offsets
LKINCTRL_NO_OF_LAST_SETFLAG	Int	10	number of set flags at MovePath FB
LKINCTRL_NO_OF_LAST_VALUEFLAG	Int	10	number of value flags at MovePath FB
LKINCTRL_NO_OF_OFFSETS	Int	3	number of configurable offsets
LKINCTRL_NO_OF_PATHDATA_ELEMENTS	Int	20	length of LKinCtrl_typePathData
LKINCTRL_NO_OF_PATHS	Int	5	number of paths
LKINCTRL_NO_OF_POINTS_AT_POINT_TABLE	Int	30	number of points at point table.
LKINCTRL_NO_OF_SEQ_EXECUTE_CMDS	Int	1	number of cmds to be buffered after first cycle
LKINCTRL_NO_OF_START_EXECUTE_CMDS	Int	1	number of cmds to be buffered in first cycle
LKINCTRL_NO_OF_ZONES	Int	10	number of zone definitions
LKINCTRL_SPTP_EXECUTION_STATUS_OUTPUT	Int	0	0: Regular time status; 1: Inverted time status; 2: Approximate cartesian distance
LKINCTRL_DISABLE_FLAG_RESET	Bool	False	FALSE: Reset with execute & after done; TRUE: No reset
LKINCTRL_NO_OF_CHANNELS	UInt	0	Number of channels -1. Only relevant for HMI part. Valid values 0..x

## 7.2. LKinCtrl\_Constants

These tags are used for PathData command configuration. The values must not be changed but can be used in programming.

Name	Data type	Value	Comment
LKINCTRL_MC_MOVELINEARABS	Int	1	cmdType for linear absolute command
LKINCTRL_MC_MOVELINEARREL	Int	2	cmdType for linear relative command
LKINCTRL_MC_MOVECIRCULARABS	Int	3	cmdType for circular absolute command
LKINCTRL_MC_MOVECIRCULARREL	Int	4	cmdType for circular relative command
LKINCTRL_MC_MOVEDIRECTABS	Int	5	cmdType for move direct absolute command
LKINCTRL_MC_MOVEDIRECTREL	Int	6	cmdType for move direct relative command
LKINCTRL_MC_TRACKCONVEYORBELT	Int	10	cmdType for track conveyor belt command
LKINCTRL_MC_MOVELINEARPICKANDPLACE	Int	11	cmdType for linear pick and place command
LKINCTRL_MC_SET_OCS_FRAME	Int	20	cmdType for set frame command
LKINCTRL_MC_SET_USER_FRAME	Int	21	cmdType for set user frame
LKINCTRL_MC_SET_TOOL_FRAME	Int	22	cmdType for defining tool frame
LKINCTRL_MC_ACTIVATE_TOOL	Int	23	cmdType for activating a tool
LKINCTRL_MC_DEFINE_WS_ZONE	Int	30	cmdType for defining workspace zone
LKINCTRL_MC_ACTIVATE_WS_ZONE	Int	31	cmdType for activating a workspace zone
LKINCTRL_MC_DEACTIVATE_WS_ZONE	Int	32	cmdType for deactivating workspace zone
LKINCTRL_MC_DEFINE_KIN_ZONE	Int	33	cmdType for defining workspace zone
LKINCTRL_MC_ACTIVATE_KIN_ZONE	Int	34	cmdType for activating a workspace zone
LKINCTRL_MC_DEACTIVATE_KIN_ZONE	Int	35	cmdType for deactivating workspace zone
LKINCTRL_CONFIG_END_OFFSET_COMP	Int	40	cmdType for ending offset compensation
LKINCTRL_CONFIG_START_OFFSET_COMP_LEFT	Int	41	cmdType for starting offset compensation left
LKINCTRL_CONFIG_START_OFFSET_COMP_RIGHT	Int	42	cmdType for starting offset compensation right
LKINCTRL_MC_DESYNC_CONVEYOR	Int	50	cmdType for conveyor desync

Name	Data type	Value	Comment
LKINCTRL_MC_SHIFT_CS_REL	Int	60	cmdType for relative shift of OCS
LKINCTRL_MC_RESET_SHIFT	Int	61	cmdType for resetting relative shift of CS
LKINCTRL_CMD_WAIT_TIME	Int	100	cmdType for wait time command
LKINCTRL_CMD_FLAG_ONLY	Int	0	cmdType for flagOnly command (no motion)
LKINCTRL_PATHDATA_TYPE_REDUCED	USInt	1	constant for PathData type "reduced"
LKINCTRL_PATHDATA_TYPE_NORMAL	USInt	2	constant for PathData type "normal"
LKINCTRL_PATHDATA_TYPE_ADVANCED	USInt	3	constant for PathData type "advanced"
LKINCTRL_FM_DEACTIVATED	USInt	0	flag deactivated
LKINCTRL_FM_SET_BEFORE_AND_NO_RESET	USInt	1	set flag before cmd and no reset
LKINCTRL_FM_SET_BEFORE_AND_RESET_AFTER	USInt	2	set flag before cmd and reset after
LKINCTRL_FM_SET_BEFORE_AND_RESET_AFTER_ONE_CYCLE	USInt	3	set flag before cmd and reset after one cycle
LKINCTRL_FM_SET_BEFORE_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE	USInt	5	set flag before cmd and wait for external reset
LKINCTRL_FM_SET_AFTER_AND_NO_RESET	USInt	11	set flag after cmd and no reset
LKINCTRL_FM_SET_AFTER_AND_RESET_AFTER_ONE_CYCLE	USInt	13	set flag after cmd and reset after one cycle
LKINCTRL_FM_SET_AFTER_AND_NO_RESET_AND_WAIT_FOR_ACKNOWLEDGE	USInt	15	set flag after cmd and wait for external reset
LKINCTRL_FM_SET_AT_REMAINING_DISTANCE	USInt	10	set flag at remaining distance
LKINCTRL_FM_DISTANCE_TOLERANCE	LReal	0.01	tolerance for flagmode 10
LKINCTRL_NO_OF_VALID_BITS	Int	6	number of valid bits in PathDataElement_advanced
LKINCTRL_NO_OF_PATHDATA_PREBUFFER_ELEMENTS	Int	5	number of PreBuffer elements
LKINCTRL_NO_OF_SETFLAGS_BUFFER_ELEMENTS	Int	15	set as double of LKINCTRL_NO_OF_PATHDATA_PREBUFFER_ELEMENTS
LKINCTRL_PI	LReal	3.141592 65358979 32846264	constant value for pi
LKINCTRL_COMP_LEFT	Int	1	tool radius compensation to the left
LKINCTRL_COMP_RIGHT	Int	2	tool radius compensation to the right
LKINCTRL_OUTER_CORNER	Int	1	Outer corner (linear / linear intersection)
LKINCTRL_INNER_CORNER	Int	2	Inner corner (linear / linear intersection)

Name	Data type	Value	Comment
LKINCTRL_CIRCMODE_GCODE	Int	3	circMode used in GCode
LKINCTRL_CS_NO_OF_FIRST_USER_FRAME	Int	11	index of first user frame
LKINCTRL_CS_USER_FRAME_1	Int	11	select first user frame
LKINCTRL_CS_USER_FRAME_2	Int	12	select second user frame
LKINCTRL_CS_USER_FRAME_3	Int	13	select third user frame
LKINCTRL_CS_USER_FRAME_4	Int	14	select fourth user frame
LKINCTRL_CS_USER_FRAME_5	Int	15	select fifth user frame
LKINCTRL_CS_NO_OF_LAST_USER_FRAME	Int	15	index of last user frame
LKINCTRL_FM_NO_SET_AND_WAIT_FOR_FALSE_ONCE	USInt	25	do not set or reset flag but wait for flag to be false once during command
LKINCTRL_FM_NO_SET_AND_WAIT_FOR_TRUE_ONCE	USInt	26	do not set or reset flag but wait for flag to be true once during command
LKINCTRL_FM_RESET_BEFORE	USInt	20	reset flag at start of command
LKINCTRL_FM_RESET_AFTER	USInt	21	reset flag at end of command
LKINCTRL_FM_RESET_AT_REMAINING_DISTANCE	USInt	22	reset flag if remaining distance of current command is reached
LKINCTRL_MEAS_CMD_WO_ACK	USInt	1	continue after measurement command without acknowledge
LKINCTRL_MEAS_CMD_WITH_ACK	USInt	2	continue after measurement command only after acknowledging
LKINCTRL_NO_OF_PATHDATA_ELEMENTS_ADVANCED	Int	20	LEGACY / length of LKinCtrl_typePathData_advanced
LKINCTRL_NO_OF_PATHDATA_ELEMENTS_REDUCED	Int	10	LEGACY / length of LKinCtrl_typePathData_reduced
LKINCTRL_MC_SET_SPTP_DYNAMICS	Int	72	cmdType for set default moveDirect (ptp) dynamics
LKINCTRL_MC_SET_PATH_DYNAMICS	Int	70	cmdType for set default path dynamics
LKINCTRL_MC_SET_ORIENTATION_DYNAMICS	Int	71	cmdType for set default orientation dynamics
LKINCTRL_LAST_INDEX_INFO_BUFFER	Int	20	Last entry in buffered commands output array
LKINCTRL_FIRST_INDEX_INFO_BUFFER	Int	-10	First entry in buffered commands output array

### 7.3. MC\_Constants

These tags are used for configuration of MC commands. The values must not be changed but can be used in programming.

Name	Data type	Value	Comment
BM_BUFFERED	Int	1	Buffer mode 1: Standstill between motions
BM_BLENDING_LOW	Int	2	Buffer mode 2: Blending with lower velocity
BM_BLENDING_HIGH	Int	5	Buffer mode 5: Blending with higher velocity
DA_DEFAULT	Int	-1	Dynamic adaption -1: Use default
DA_DEACTIVATED	Int	0	Dynamic adaption 0: Deactivated
DA_WITH_VARIABLE_LIMITS	Int	1	Dynamic adaption 1: With segmentation of path
DA_WITH_CONSTANT_LIMITS	Int	2	Dynamic adaption 2: No segmentation of path
SM_ACTIVE_DYNAMICS	Int	0	Stop mode 0: Stop with active dynamics
SM_MAX_DYNAMICS	Int	1	Stop mode 1: Stop with maximum dynamics
PC_POSITIVE	Int	0	Path choice 0: Shorter positive circle segment
PC_LONG_RUN_POSITIVE	Int	2	Path choice 2: Longer positive circle segment
PC_LONG_RUN_NEGATIVE	Int	3	Path choice 3: Longer negative circle segment
CP_XZ	Int	0	Circle plane 0: Circle in X-Z plane
CP_YZ	Int	1	Circle plane 1: Circle in Y-Z plane
CP_XY	Int	2	Circle plane 2: Circle in X-Y plane
CM_BORDER	Int	0	Circle mode 0: "EndPoint" + "AuxPoint" parameter defines a point on circular path
CM_CENTER	Int	1	Circle mode 1: "Arc" + "AuxPoint" defines circle center
CM_RADIUS	Int	2	Circle mode 2: "Radius" + "EndPoint" parameters define the circle segment
IM_ACTIVE_DYNAMICS	Int	0	Interrupt mode 0: Interrupt with active dynamics
IM_MAX_DYNAMICS	Int	1	Interrupt mode 1: Interrupt with maximum dynamics
DIRA_POSITIVE	Int	1	Orientation direction 1: Positive direction
DIRA_NEGATIVE	Int	2	Orientation direction 2: Negative direction



Name	Data type	Value	Comment
DIRA_SHORTEST	Int	3	Orientation direction 3: Shortest direction
CS_WCS	Int	0	Coordinate system 0: World coordinate system (WCS)
CS_OCS1	Int	1	Coordinate system 1: Object coordinate system 1 (OCS[1])
CS_OCS2	Int	2	Coordinate system 2: Object coordinate system 2 (OCS[2])
CS_OCS3	Int	3	Coordinate system 3: Object coordinate system 3 (OCS[3])
PC_NEGATIVE	Int	1	Path choice 1: Shorter negative circle segment
CS_MCS	Int	100	Coordinate system 100: Machine coordinate system (axis values)
CS_JCS	Int	101	Coordinate system 101: Joint coordinate system (joint values)
LC_DEFAULT	DWord	16#FFFFFF	LinkConstellation: Default to keep link constellation of kinematics
PM_ABS_ABS	DInt	1	PositionMode 1: Interpretation of cartesian orientation as absolute (only effective for absolute MoveDirect commands)
PM_ABS_REL	DInt	2	PositionMode 2: Interpretation of cartesian orientation as relative (only effective for absolute MoveDirect commands)
CS_FCS	DInt	0	Coordinate system 0: Flange coordinate system (FCS); Only valid for kinematics zone commands
CS_TCS	DInt	1	Coordinate system 1: Tool coordinate system (TCS); Only valid for kinematics zone commands
ZONE_DEACTIVATION_MODE_SPECIFIC	DInt	0	Zone deactivation mode 0: Specific zone index
ZONE_DEACTIVATION_MODE_ALL	DInt	1	Zone deactivation mode 1: All zones
ZONE_DEACTIVATION_MODE_BLOCKED	DInt	2	Zone deactivation mode 2: All blocked zones
ZONE_DEACTIVATION_MODE_SIGNAL	DInt	3	Zone deactivation mode 3: All signal zones
ZONE_DEACTIVATION_MODE_WORKSPACE	DInt	4	Zone deactivation mode 4: Currently active workspace zone

## 8. Appendix

### 8.1. Service and support

#### SiePortal

The integrated platform for product selection, purchasing and support - and connection of Industry Mall and Online support. The SiePortal home page replaces the previous home pages of the Industry Mall and the Online Support Portal (SIOS) and combines them.

- **Products & Services**  
In Products & Services, you can find all our offerings as previously available in Mall Catalog.
- **Support**  
In Support, you can find all information helpful for resolving technical issues with our products.
- **mySieportal**  
mySiePortal collects all your personal data and processes, from your account to current orders, service requests and more. You can only see the full range of functions here after you have logged in.

You can access SiePortal via this address: [sieportal.siemens.com](https://sieportal.siemens.com)

#### Technical Support

The Technical Support of Siemens Industry provides you fast and competent support regarding all technical queries with numerous tailor-made offers – ranging from basic support to individual support contracts.

Please send queries to Technical Support via Web form: [support.industry.siemens.com/cs/my/src](https://support.industry.siemens.com/cs/my/src)

#### SITRAIN – Digital Industry Academy

We support you with our globally available training courses for industry with practical experience, innovative learning methods and a concept that's tailored to the customer's specific needs.

For more information on our offered trainings and courses, as well as their locations and dates, refer to our web page: [siemens.com/sitrain](https://siemens.com/sitrain)

#### Industry Online Support app

You will receive optimum support wherever you are with the "Industry Online Support" app. The app is available for iOS and Android:



## 8.2. Application support

Siemens AG  
 Digital Industries Division  
 Factory Automation  
 Production Machines  
 DI FA PMA APC  
 Frauenaauracher Str. 80  
 91056 Erlangen, Germany

mailto: [tech.team.motioncontrol@siemens.com](mailto:tech.team.motioncontrol@siemens.com)

## 8.3. Links and literature

### No. Topic

11\	Siemens Industry Online Support <a href="https://support.industry.siemens.com">https://support.industry.siemens.com</a>
12\	Link to this entry page of this application example <a href="https://support.industry.siemens.com/cs/ww/en/view/109755891">https://support.industry.siemens.com/cs/ww/en/view/109755891</a>
13\	Link to TO Kinematics manual <a href="https://support.industry.siemens.com/cs/ww/en/view/109817886">https://support.industry.siemens.com/cs/ww/en/view/109817886</a>

## 8.4. Change documentation

Version	Date	Modifications
V1.0.0	04/2018	First version
V1.0.1	04/2018	New UDT added for JogFrame No changes for MC_MovePath functionality
V1.0.2	08/2018	Updates for JogFrame New additional function blocks for TO Kinematics errorWord & warningWord evaluation
V1.0.7	12/2018	BugFixes active signal MC_MovePath
V1.0.13	01/2019	Fixes interrupt / continue triggering MC_MovePath Additional error information MC_JogFrame
V2.0	04/2019	Version Update: Flag functionalities Set / reset / set in remainingDistance Wait times Different PathData Types Path generation from points Deactivation of path commands Radius and tool compensation calculation Memory and runtime optimized
V2.1	05/2019	Subversion update: Adaptions for use in LKinLang

Version	Date	Modifications
V2.2	02/2020	Subversion update: valueFlags added FlagOnly commands added
V2.3	03/2020	Subversion update: Added Editor faceplate Separate constants for flag arrays Added valueFlags to PathDescription
V2.4	06/2020	Subversion update: Handle preprocessing stop for LKinLang
V3.0	08/2020	Version update: Added sPTP commands Added ConveyorTracking commands
V3.0.2	10/2020	Updated error constants
V3.1.0	08/2021	Added new commands Updated editor
V3.1.3	11/2021	Updated flagMode
V3.2.0	02/2022	Fixed wrong naming of FlagMode
V3.3.0	08/2022	Added orientation dynamics to PathData New FlagModes Updated editor
V4.0.0	01/2023	Added point reference to PathData Support MotionControl V7 Updated appendix
V4.0.1	02/2023	Updated PLC Tags
V4.1.0	01/2024	Added Pick-and-Place-Block, Relative Shift and Teach Faceplate
V4.1.5	05/2024	Added new error 16#8100 to FB JogFrame
V4.2.0	08/2024	Added description for measurement command
V5.0.0	11/2024	Moved example project into separate documentation Removed description for FC LKinCtrl_PathSelect and type LKinCtrl_typePathDescription Updated FB interfaces Update for Error handling chapter
V5.1.0	05/2025	Added pathData command overview Update for MovePickAndPlaceLinear command
V5.2.0	09/2025	Added new commands Added JCS error description for JogFrame Added Caution note to programming & supported command